IBM

DB2 사용자 가이드

for Linux, UNIX and Windows

Published by IBM 정보관리사업부

IBM Software Group
Information Management DB2 FTSS
Information Management Marketing

Information Management software

DB2 사용자 가이드

for Unix, Linux and Windows

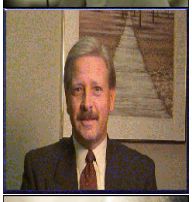
Document History

Version 1.0 2006-04-28
Version 2.0 2009-09-30

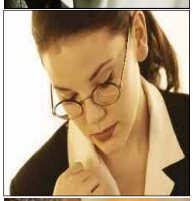
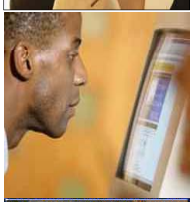
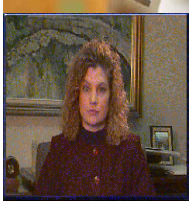
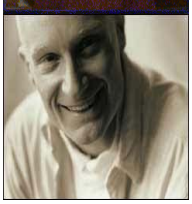
IBM Software Group
Information Management DB2 FTSS

Contents

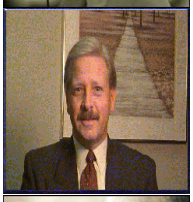
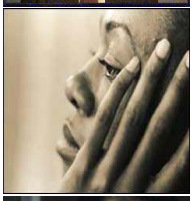


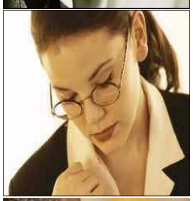
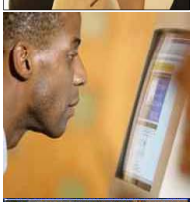
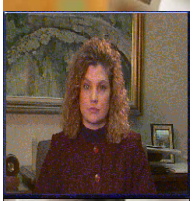
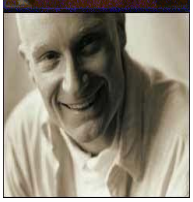




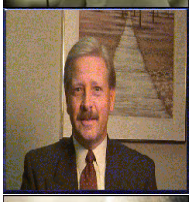


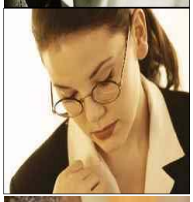
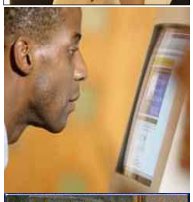






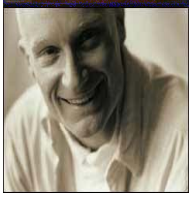
Contents



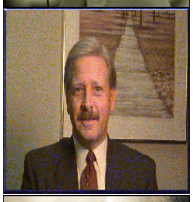
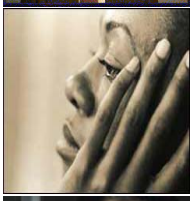


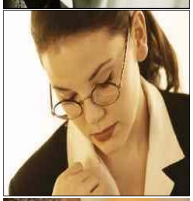
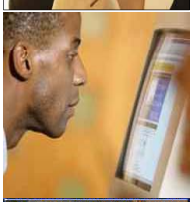
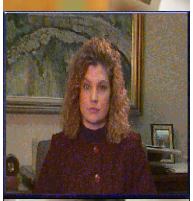
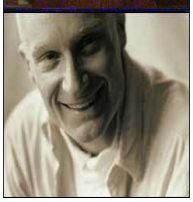




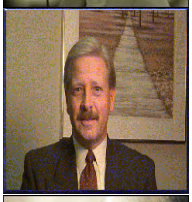


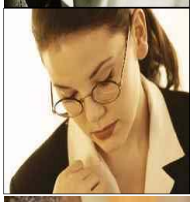
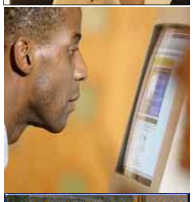






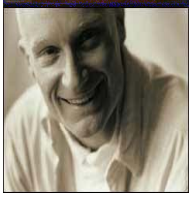
Contents

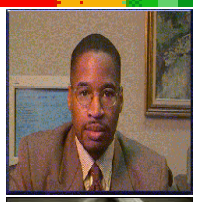


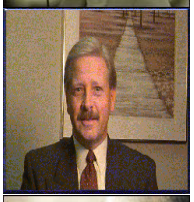




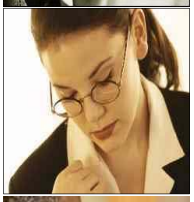
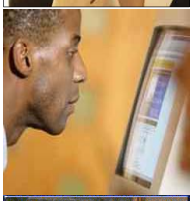



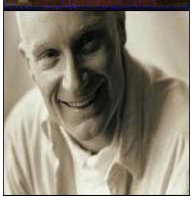




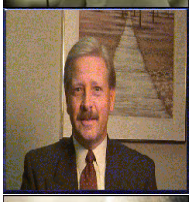


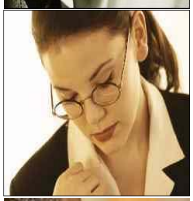


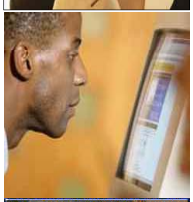



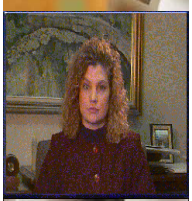


Contents

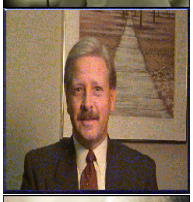




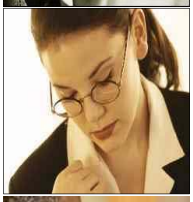
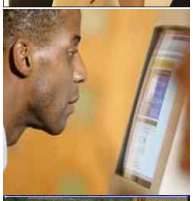


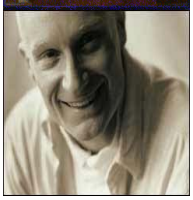










UNIT 01

제품소개

DB2 for LUW 는 Linux, Unix, Windows 에서 사용되는 RDBMS 입니다. 서버에 DB2 데이터베이스 서버를 구성하려면 DB2 ESE 를 설치 합니다. 클라이언트 용도인 경우에는 최소한 Data Server Runtime Client 제품을 설치 합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 서버 제품 군
- 클라이언트 제품 군
- 지원되는 프로토콜
- 유용한 URL

TOPIC 0101 서버 제품 군

Point

DB2는 Express Edition, Workgroup Server Edition, Enterprise Server Edition 등의 서버 제품 군이 존재 합니다.
호스트 시스템을 위한 DB2 iSeries와 DB2 zSeries도 있습니다.

Tip

다양한 플랫폼 별로 지원되는 모든 DB2 제품들을 DB2 Family 라고 합니다.

Tip

DB2 Family 간에는 DRDA라는 IBM의 전용 데이터 프로토콜이 사용됩니다. 클라이언트의 액세스 요청은 DRDA 프로토콜을 이용하여 서버에 전달되고, 서버는 DRDA 프로토콜을 이용하여 결과를 반환합니다.

1. 모든 서버 제품 군은 90%이상 동일한 코드를 기반 하였으므로 애플리케이션 코드의 변경 없이 보다 큰 서버 제품 군으로 또는 다중 프로세서 서버 클러스터로 확장할 수 있습니다.

제 품	용 도	지원 OS	CPU 메모리 제한	DataSize 제한	설 명
EXP	개발자 및 솔루션업체	Windows Linux	2 CPU 4GB 까지	무제한	DB2 Express의 약자. 무료로 개발/구축/배포할 수 있음.
WSE	중소형 기업	Windows Linux AIX HP-UX Solaris Linux/390	4 CPU 16GB 까지	무제한	Workgroup Server Edition의 약자. High Availability 기능이 포함 됨.
ESE	대용량 처리	Windows Linux AIX HP-UX Solaris Linux/390	무제한	무제한	Enterprise Server Edition 의 약자. High Availability 기능 포함, Workload Management, Table Partitioning Feature 지원

2. DB2 ESE에서 지원하는 DPF(Database Partitioning Feature) 기능

DPF는 다중 데이터베이스 파티션 기능을 이용하여 병렬 데이터베이스를 구축하는 기능입니다. 즉 여러 시스템에 파티션을 생성하여 데이터베이스를 분리하지만 사용자에게는 논리적인 단일 데이터베이스로 보임으로써 확장성 및 병렬성, 고성능을 제공합니다. DPF는 DB2 9.5부터 IBM Infosphere Warehouse Edition에 포함되었습니다.

3. AS/400과 OS/390 에서 지원되는 DB2 제품 군을 호스트 DB2라고 합니다.

제 품	설 명
DB2 iSeries	OS/400 에서 지원됩니다.
DB2 zSeries	OS/390 에서 지원됩니다.

TOPIC 0102 클라이언트 제품 군

Point

서버에 설치된 WSE, ESE with DPF 등의 서버 제품 군에 접속을 원하는 클라이언트에 설치하는 제품 군으로 Data Server Runtime Client, Data Server Client가 있습니다. 호스트 DB2에 접속하려면 DB2 Connect 제품이 필요합니다.

Tip

DRDA 프로토콜을 이용한 통신에서 데이터를 요청하는 프로그램을 DRDA Application Requester 라고 합니다.

Tip

DRDA 프로토콜을 이용한 통신에서 데이터를 제공하는 서버를 DRDA Application Server 라고 합니다.

Tip

DB2 서버 제품은 클라이언트 모듈을 포함하고 있습니다.

Tip

DB2 9.1이전까지는 Runtime Client와 Runtime Client Lite 가 따로 존재 하였으나 DB2 9.1부터 두 제품이 결합되었으며, DB2 9.5부터 Data Server Runtime Client로 명칭이 변경 되었습니다.

Tip

DB2 9.1 이전까지는 Application Development Client 와 Administration Client 가 따로 존재 하였으나 DB2 9.1부터 두 제품이 결합되었으며, DB2 9.5부터 Data Server Client로 명칭이 변경 되었습니다.

Tip

DB2 Connect Enterprise Edition은 DB2 gateway라고도 합니다.

1 클라이언트의 요청은 서버의 엔진에 포함된 통신 리스너 프로세스에 의해 처리됩니다.

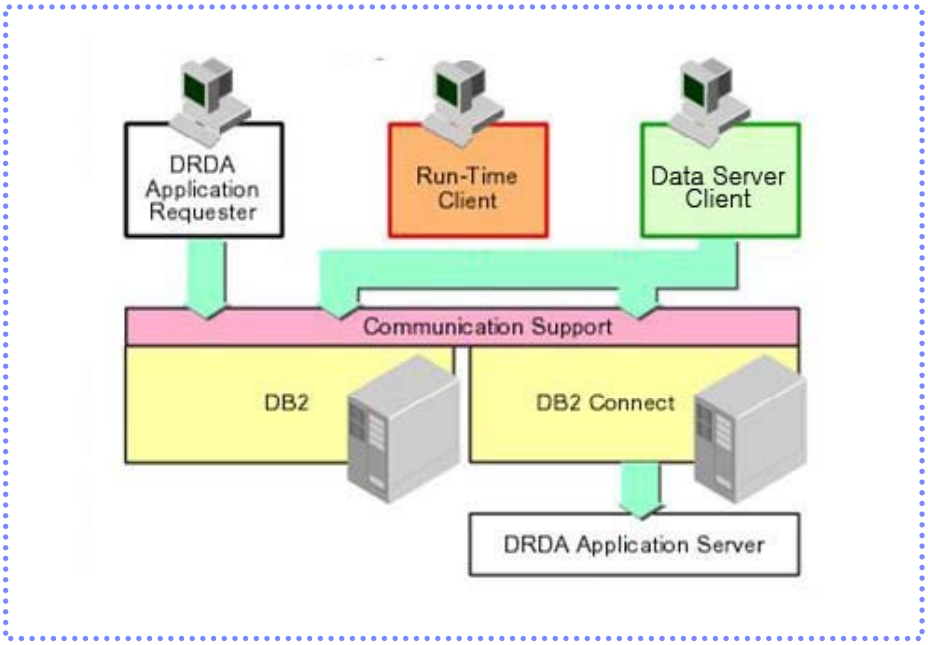


Figure 0102A ••• 클라이언트 제품 군과 DB2 Connect

2 DB2 클라이언트 제품을 이용하여 DB2 서버에 액세스할 수 있습니다.

제 품	제 품
IBM Data Server Runtime Client	ODBC, JDBC 등을 포함한 실행 환경을 제공하는 최소한의 클라이언트 모듈입니다.
IBM Data Server Client	Run-Time Client 의 기능을 포함하며, 다양한 GUI 도구들을 이용한 관리 작업을 가능하게 하는 클라이언트 모듈과 ESQL 등의 개발이 가능하도록 하는 프리 컴파일러, 헤더파일, 라이브러리 등을 제공하는 클라이언트 모듈입니다.

http://www-01.ibm.com/software/data/db2/9/download.html 에서 DB2 Client 제품을 다운로드 받아 설치할 수 있습니다.

3 DB2 Connect를 이용하면 OS/390, AS/400 등의 호스트에 있는 데이터베이스를 액세스할 수 있습니다.

제 품	제 품
DB2 Connect Personal Edition	Linux 와 Windows 플랫폼에서 지원되는 단일 사용자용 호스트 데이터베이스 액세스 제품입니다.
DB2 Connect Enterprise Edition	Windows, AIX, HP-UX, Solaris, Linux, Linux/390 플랫폼에서 지원되는 다중 사용자용 호스트 데이터베이스 액세스 제품입니다.

TOPIC 0103 지원되는 프로토콜

Point

LUW환경에서 DB2 클라이언트와 서버의 통신을 위해 TCP/IP, Named Pipe, SSL 프로토콜을 지원합니다. 호스트 DB2 서버에 접속할 때는 TCP/IP 프로토콜을 이용합니다.

Tip

TCP/IP 프로토콜은 설정이 간편하여 가장 많이 사용됩니다.

1 DB2 서버와 클라이언트는 TCP/IP, Named Pipe, SSL 등의 통신 프로토콜을 사용합니다.

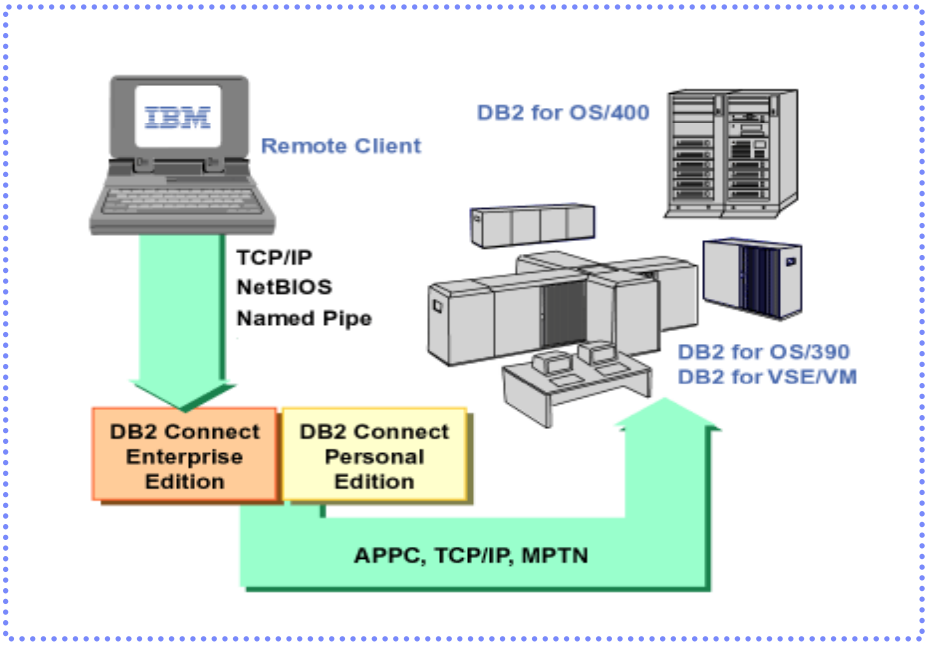


Figure 0103A ··· DB2가 지원하는 프로토콜

Tip

NetBIOS 및 SNA 프로토콜은 DB2® 버전 9.1 이후 더 이상 지원되지 않습니다.

2 데이터베이스 서버에 설치된 WSE, ESE, ESE with DPF 등의 서버 제품과 클라이언트에 설치된 Data Server Clinet 등의 클라이언트 제품간의 물리적인 통신은 TCP/IP, Named Pipe, SSL 프로토콜을 사용합니다.

3 호스트 데이터베이스 서버에 설치된 DB2/390, DB2/400 등의 서버 제품과 클라이언트에 설치된 DB2 Connect Personal Edition, DB2 Connect Enterprise Edition 등의 클라이언트 제품간의 물리적인 통신은 TCP/IP 프로토콜을 사용합니다.

Tip

OS/390과 AIX간의 프로토콜과 AIX와 Windows간의 프로토콜은 다른 종류로 설정할 수 있습니다.

4 Windows에서 AIX에 설치된 DB2 Connect를 게이트웨이로 OS/390의 데이터베이스를 액세스할 수 있습니다. OS/390과 AIX간의 프로토콜을 설정하여 AIX에서 OS/390 의 접속을 확인하고, AIX와 Windows간의 프로토콜 설정을 하도록 합니다.

5 TCP/IP 프로토콜을 사용할 때는 각 서버간의 물리적인 네트웍 설정이 완료된 상태에서 각 서버의 IP주소와 통신 포트 번호만 DB2에 설정하면 됩니다.

TOPIC 0104 유용한 URL

Point

DB2에 대한 다양한 정보를 얻을 수 있는 URL 입니다.

1 온라인 정보 센터

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp 에서 DB2 에서 제공되는 명령어와 SQL문에 대한 종합적인 정보를 확인할 수 있습니다.

2 온라인 제품 매뉴얼

http://www-01.ibm.com/support/docview.wss?rs=71&uid=swg27009474 에서 영문 PDF 제품 매뉴얼을 다운로드할 수 있습니다.

3 SQL Cookbook

http://mysite.verizon.net/Graeme_Birchall/id1.html URL 에서 DB2에서 제공되는 각종 SQL 함수에 대한 설명을 예제를 이용하여 정리한 자료를 다운로드할 수 있습니다.

4 Technical Library

http://www-128.ibm.com/developerworks/views/db2/libraryview.jsp 에서 DB2에 대한 다양한 TIP 들을 검색할 수 있습니다.

5 Redbooks

www.redbooks.ibm.com 에서 DB2와 관련된 기술 서적을 찾아볼 수 있습니다.

Tip

별도의 FTP 프로그램을 이용할 때는 IBM의 소프트웨어 다운로드 사이트인 ftp.software.ibm.com 를 사용할 수 있습니다.

6 Fixpack과 Client 모듈 다운로드

http://www-01.ibm.com/software/data/db2/support/db2_9/download.htm 에서 플랫폼과 버전별로 DB2 의 Fixpack과 클라이언트 모듈을 다운로드 할 수 있습니다.

7 Fixpack별 APAR 리스트

http://www-01.ibm.com/software/data/db2/9/download.html 에서 DB2의 FP별로 상세한 APAR 정보를 확인할 수 있습니다.

8 사용자 게시판

http://kdug.kr/ 에서 DB2와 관련된 정보를 얻을 수 있으며, 게시판을 통해 다른 사용자들과 다양한 정보를 공유할 수 있습니다.

9 IBM 홈페이지

http://www-01.ibm.com/software/kr/data/db2/index.html 에서 DB2 제품에 관한 최신 정보를 접할 수 있습니다.

UNIT 02

서버 설치

DB2 제품과 FixPack은 플랫폼과 버전별로 제공되며,각각 db2_install 과 installFixPack 명령어를 이용하여 사용자가 간단하게 설치할 수 있도록 되어 있습니다. 플랫폼에 적합한 기본적인 서버 제품을 설치하고, 최신의 FixPack 을 적용하도록 합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- CDROM 장치 정의
- CDROM 파일 시스템 정의 및 마운트
- 설치용 파일 복사
- 시스템 및 소프트웨어 설치 요구사항
- 기본 설치
- 한글 메시지 파일 설치
- FixPack 다운로드
- FixPack 적용
- 라이센스 등록
- DB2 9.7 업그레이드

TOPIC 02 01 CDROM 장치 정의

Point

설치용 CD를 이용하려면, mkdev 명령어 또는 smitty cdrom 명령어를 이용하여 CDROM 장치를 정의합니다. ftp를 이용하여 설치 이미지를 다른 서버에서 전송 받는다면 이 단계는 필요하지 않습니다.

Tip

lsdev 명령어의 SMIT 경로는 smitty cdrom -> 정의된 모든 CDROM 드라이브 리스트 입니다.

Tip

mkdev 명령어의 SMIT 경로는 smitty cdrom -> CD ROM 드라이브 표시 입니다.

1 root 사용자로 로그인하여 CDROM 장치가 정의되어 있는지 확인합니다.

```
$ login root
$ lsdev -Cc cdrom
```

2 지원되는 CDROM 장치를 확인합니다.

```
$ lsdev -Pc cdrom
cdrom cdrom1     scsi CD-ROM 드라이브
cdrom oscd       scsi 기타 SCSI CD-ROM 드라이브
cdrom scsd       scsi 기타 SCSI CD-ROM 드라이브
```

3 mkdev 명령어를 이용하여 CDROM 장치를 정의합니다.

```
$ mkdev -c cdrom -t 'scsd' -s 'scsi' -p 'scsi0' -w '1,0'
```

4 CDROM 장치가 정의되어 장치명이 cd0 인 것을 확인합니다.

```
$ lsdev -Cc cdrom
cd0 사용 가능 40-60-00-1,0 16 비트 SCSI멀티미디어CDROM
```

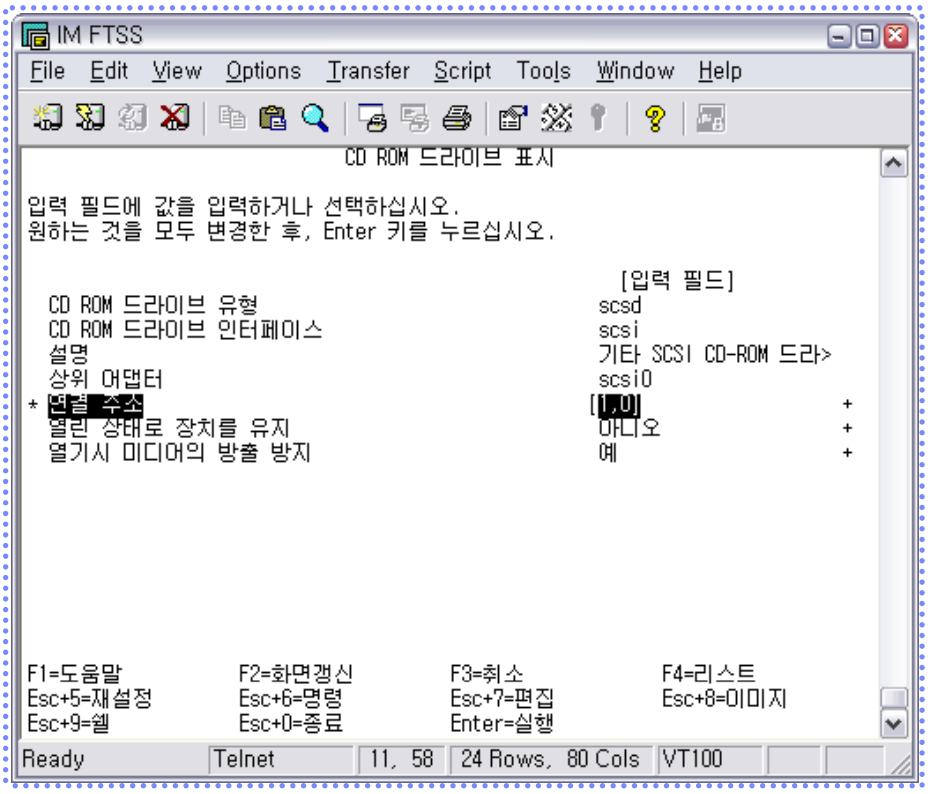


Figure 0201A SMIT을 이용한 CDROM 장치 정의

TOPIC 02-02 CDROM 파일 시스템 정의 및 마운트

Point

CDROM 파일시스템은 crfs 명령어 또는 smitty crfs 명령어로 정의합니다. 정의된 CDROM 파일시스템은 mount 명령어 또는 smitty mount 명령어를 이용하여 마운트해서 사용합니다.

Tip

crfs 명령어의 SMIT 경로는 smitty crfs -> CDROM 파일시스템 추가 입니다.

Tip

mount 명령어의 SMIT 경로는 smitty mount -> 파일시스템 마운트 입니다.

1 root 사용자로 로그인하여 lsfs 명령어로 CDROM 파일시스템이 정의되어 있는지 확인합니다.

```
$ login root
$ lsfs | grep cdrfs
```

2 crfs 명령어를 이용하여 CDROM 파일시스템을 정의합니다.

```
$ crfs -v cdrfs -p ro -d'cd0' -m'/cdrom' –A"
```

3 CDROM 파일시스템이 성공적으로 정의된 것을 확인합니다.

```
$ lsfs | grep cdrfs
/dev/cd0      --      /cdrom          cdrfs --    ro
```

4 CDROM 드라이브에 설치용 CD를 넣고, 마운트합니다.

```
$ mount /cdrom
```

5 CDROM 파일시스템이 성공적으로 마운트된 것을 확인합니다.

```
$ df
```

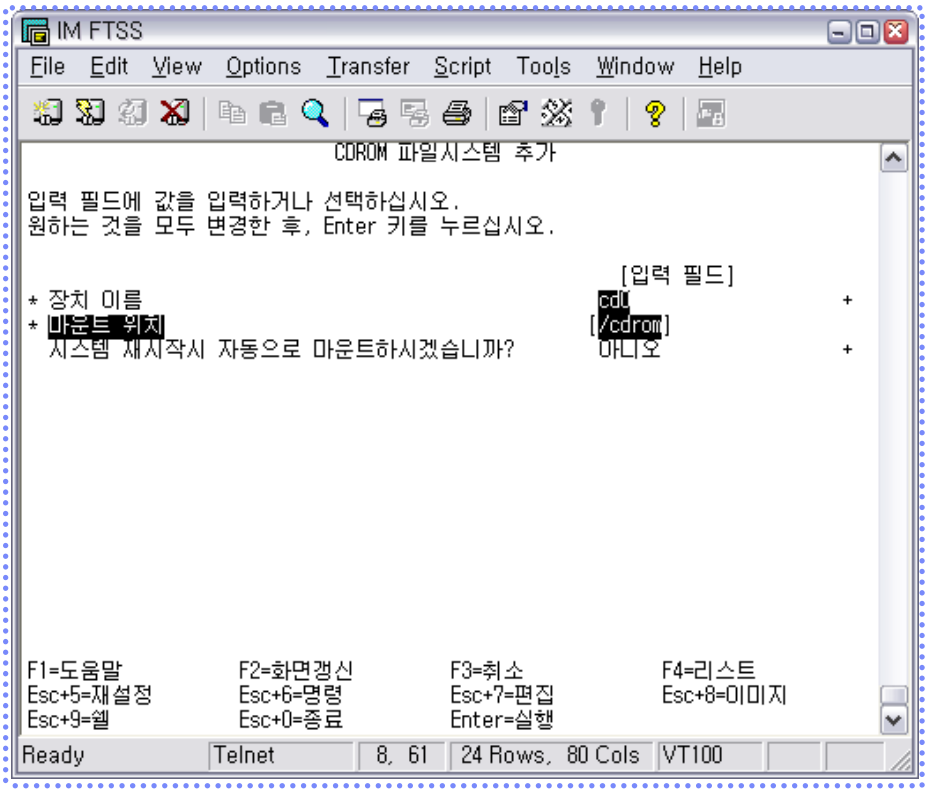


Figure 0202A SMIT을 이용한 CDROM 파일시스템 정의

TOPIC 02 03 설치용 파일 복사

Point

설치용 CD의 설치 모듈은 압축되어 있으므로 임시 디렉토리로 복사하여 gzip 명령어와 tar 명령어로 압축을 해제합니다. 임시 디렉토리는 충분히 커야 합니다.

Tip

<이미지명.gz> 이미지와 압축해제 된 이미지 <이미지명.tar> 의 크기를 수용 할 수 있는 파일시스템 디렉토리가 충분히 커야 합니다.

1 root 사용자로 로그인하여 임의의 임시 디렉토리로 이동합니다.

```
$ login root
$ mkdir product
$ cd /product
```

2 설치용 CD에 있는 압축 파일을 임시 디렉토리로 복사합니다.

```
$ cp /cdrom/<이미지명.gz> .
```

3 gzip과 tar 명령어로 압축을 해제합니다.

```
$ gzip -d <이미지명.gz>
$ tar -xvf <이미지명.tar>
```

4 압축이 해제되면 server 디렉토리에 설치에 필요한 파일들이 저장됩니다. 압축 해제 작업이 완료된 후에는 tar 파일을 삭제할 수 있습니다.

```
$ ls server
$ rm *.tar
```

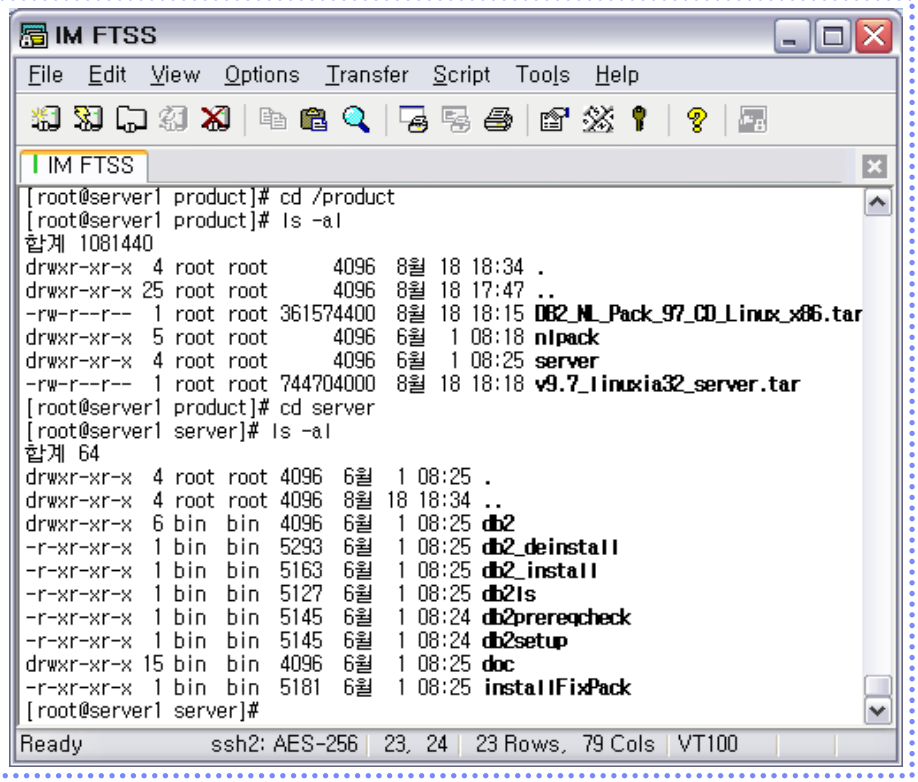


Figure 0203A ··· 설치용 파일 복사 및 압축 해제

TOPIC 02 04 시스템 및 소프트웨어 설치 요구사항

Point

DB2를 설치하기 전에 반드시 시스템의 하드웨어 및 소프트웨어 요구사항을 확인합니다.

Tip

- 아래 URL을 확인하여 DB2 제품의 설치가 가능한지 확인합니다.
- http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.qb.server.doc/doc/r0008857.html

1 시스템 요구사항을 확인 합니다.

시스템 리소스	요구 사항
디스크	제품 설치 시 필요한 디스크 공간은 설치 유형 및 파일 시스템 유형에 따라 다르지만, 대략 DB2 서버 제품 군 설치 시 1GB의 여유공간이 주어지며 데이터베이스에 대한 공간은 별도로 필요합니다. Linux/ Unix 운영체제의 경우 /tmp 에 2GB이상 여유공간 권장
메모리	DB2 데이터베이스 시스템은 최소한 256MB이 필요합니다. DB2 및 DB2 GUI 도구만 실행하는 시스템의 경우 최소 512MB 이 필요합니다. 그러나 성능을 위해서는 최소 1GB이상을 권장합니다

2 지원 Platform 및 소프트웨어 설치 요구사항을 확인 합니다.

Tip

- 설치파일이 있는 디렉토리에 db2prereqcheck를 실행하면 OS커널, 라이브러리 버전을 검사합니다. 사전 요구 환경이 준비 되어있지 않으면 필요한 커널 레벨과 라이브러리 버전을 출력해 줍니다.

운영체제	설치 요구 사항	하드웨어
AIX	AIX 버전 5.3 -.64비트 AIX 커널이 필수임 -.TL9 SP2 -.최소 C++ runtime level xlC.rte 9.0.0.8 및 xlC.aix50.rte 9.0.0.8 AIX 버전 6.12 -.64비트 AIX 커널이 필수임 -.TL 2 -.최소 C++ runtime level xlC.rte 9.0.0.8 및 xlC.aix61.rte 9.0.0.8	eServer™ pSeries ® IBM ® System p ™ IBM System p5 ™
HP-UX	11iv2(11.23.0505) -.2005년 5월 출시된 Base Quality(QPKBASE) -.2005년 5월 출시된 Applications Quality(QPKAPPS) 번들 HP-UX 11iv3(11.31)	Itanium® 기반 HP Integrity Series Systems
LINUX	RHEL(Red Hat Enterprise Linux) 5 Update 2 SLES(SUSE Linux Enterprise Server) 10 SP 2 SLES(SUSE Linux Enterprise Server) 11 Ubuntu 8.0.4.1	X86 32비트 Intel 및 AMD 프로세서 X64, POWER ® eServer System z ®
Solaris	Solaris 9 -.64비트 커널 -.패치 111711-12 및 111712-12 -.원시 디바이스가 사용되는 경우, 패치 122300-11 -.64비트 Fujitsu PRIMEPOWER 및 패치 912041-01에 대한 수정사항을 가져오기 위한 Solaris 9 커널 갱신 패치 112233-01 이상 Solaris 10 갱신 5 -.64비트 커널 -.원시 디바이스가 사용되는 경우, 패치	UltraSPARC 또는 SPARC64 프로세서

TOPIC 02 05 기본 설치

Point

설치용 파일 셋을 준비하고 설치 요구사항 확인 후, db2_install 명령어를 이용하여 설치를 시작합니다. DB2 ESE를 설치하려면, DB2ESE 라는 키워드를 입력합니다.

Tip

- DB2 9.5이후 root 사용자가 아닌 일반 사용자를 이용하여 db2 설치가 가능합니다.
- 관련 사항은 URL 참조
- http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.qb.server.doc/doc/c0050562.html

Tip

- DB2 9.1 부터 db2_install 명령어에 –b 옵션으로 설치 경로를 따로 지정할 수 있습니다.
- 설치 경로를 따로 지정하지 않으면 /opt/ibm/db2/V9.7 이 기본 경로가 됩니다.

Tip

- db2_install –p ESE –b <설치 경로>명령을 실행해도 됩니다.

1 root 사용자로 로그인하여 설치 파일 셋이 저장된 임시 디렉토리로 이동합니다.

```
$ login root
$ cd /product/server
```

2 db2_install 명령을 이용하여 설치를 시작합니다. ESE를 설치하려면 ESE를 입력합니다.

```
[root@localhost server]# ./db2_install
Default directory for installation of products - /opt/ibm/db2/V9.7
***********************************************************
Do you want to choose a different directory to install [yes/no] ?
no
Specify one of the following keywords to install DB2 products.
  ESE
  CONSV
  WSE
  EXP
  PE
  CLIENT
  RTCL
Enter "help" to redisplay product names.
Enter "quit" to exit.
***********************************************************
ESE
```

3 설치가 완료되면 /opt/ibm/db2/V9.7/install 이동하여 db2ls로 확인합니다.

```
[root@localhost install]# ./db2ls -q -a
```

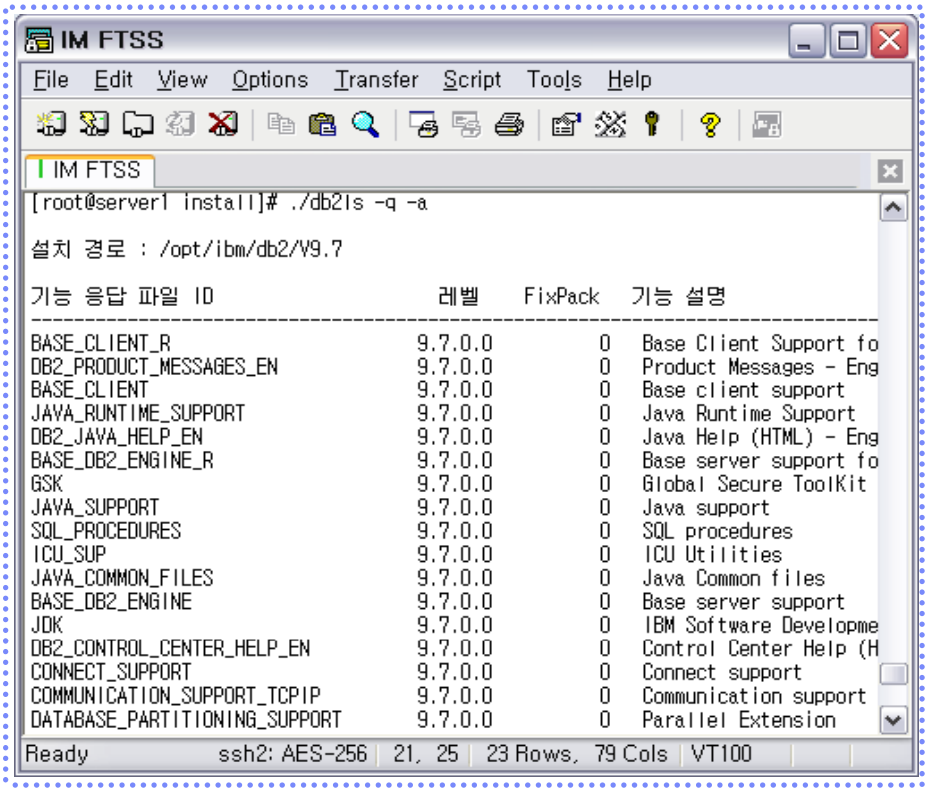


Figure 0205A • • • db2_install 명령어를 이용한 설치

TOPIC 02 06 한글 메시지 파일 설치

Point

db2_install 명령어는 기본적으로 영문용 메시지 파일만 설치하므로, 한글 메시지 등을 추가로 설치하려면 nlpack을 설치합니다. 한글 메시지 파일을 설치하면, OS의 LANG 변수가 한글 로케일인 ko_KR인 경우에 도움말이 한글로 표시됩니다.

Tip

DB2 설치 마법사는 그래픽 설치 프로그램입니다. 사용자의 시스템에서 DB2 설치 마법사를 실행하려면 그래픽 사용자 인터페이스를 렌더링할 수 있는 XWindows 소프트웨어가 있어야 합니다.

Tip

db2_install –L KR 옵션 사용전 ese 이미지와 nlpack 이미지는 같은 디렉토리에 놓거나 –c 옵션에 nlpack 경로를 지정합니다.

Tip

OS의 LANG 변수가 ko_KR로 설정되었을 때만 도움말이 한글로 표시됩니다. LANG의 값을 en_US로 설정하면 다시 영문 메시지가 표시됩니다.

1 root 사용자로 로그인하여 nlpack 설치 파일 셋이 있는 임시 디렉토리로 이동합니다.

```
$ login root
$ cd /product/nlpack
```

2 메시지 파일이 있는 디렉토리에서 db2setup을 입력합니다.

```
$ ./db2setup
```

3 /product 경로에 nlpack 이미지 압축을 풀고 기본설치에서 사용하는 명령어 db2_install 에서 –L KR 옵션을 이용하여 기본설치 진행할 때 한글메시지를 같이 설치할 수 도 있습니다.

```
$ db2_install –L KR
```

4 db2ls 명령어로 설치 확인을 합니다.

```
[root@localhost]# cd /opt/ibm/db2/V9.7/install
[root@localhost install]# ./db2ls -q –a | grep –i KR
```

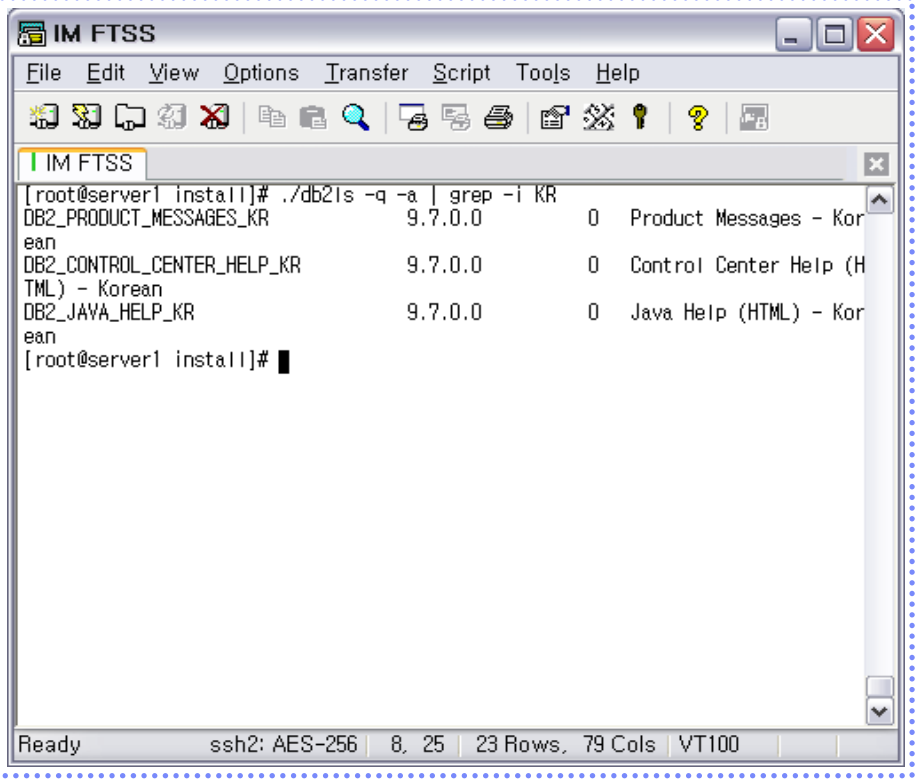


Figure 0206A ••• 한글 메시지 파일 설치

TOPIC 02 07 FixPack 다운로드

Point

IBM의 FTP 사이트에서 원하는 FixPack을 다운로드 받아 서버의 임시 디렉토리에 압축 해제하여 준비합니다. FTP 사이트는 ftp://ftp.software.ibm.com 입니다.

1 IBM의 FTP 사이트인 ftp://ftp.software.ibm.com 에 접속합니다.

(사용자 ID : anonymous, 비밀번호 : e-mail ID)

Figure 0207A ··· IBM FTP 사이트 접속

2 예를 들어, AIX V5용 DB2 9.5를 위한 FixPack 4을 다운로드 하려면 DB2 FixPack 디렉토리인 /ps/products/db2/fixes2/english-us/db2aix5v95/FixPack/FP4_U825478

으로 이동하여 v9.5fp4_aix_server.tar.gz 를 다운로드 합니다.

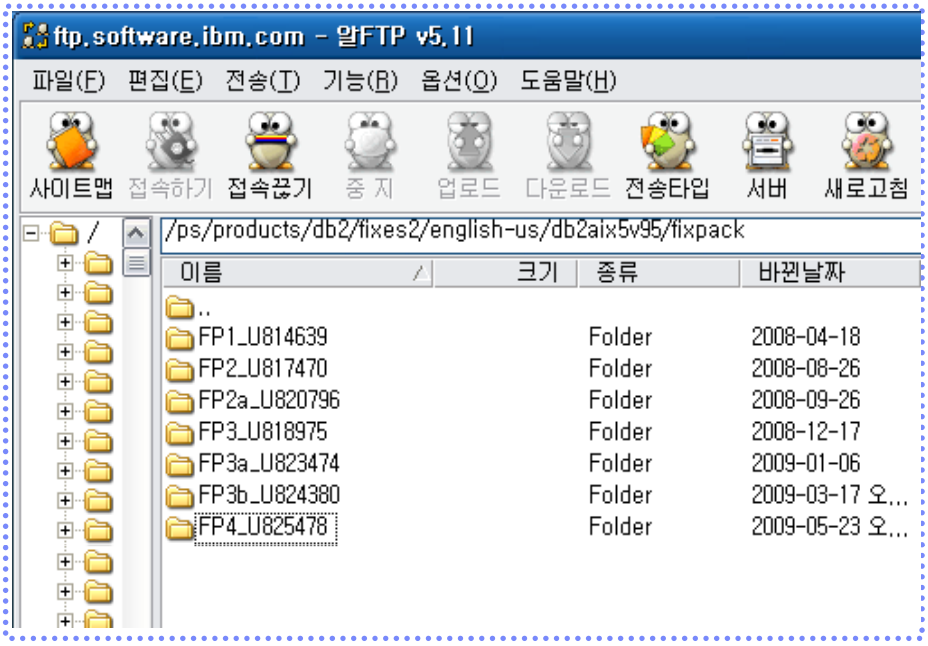


Figure 0207B ··· AIX5용 DB2 V9.5 FP4 다운로드

Tip

최신 FP에 대한 정보는 아래 URL에서 확인합니다.

http://www-01.ibm.com/support/docview.wss?rs=71&uid=swg21293566

TOPIC 02 08 FixPack 적용

Point

installFixPack 명령어에서 –b 옵션을 이용하여 FixPack을 적용합니다.

Tip

FixPack을 적용하기 전에 DB2 엔진을 중지합니다.

Tip

인스턴스가 존재하는 상태에서 FixPack적용 후에는 db2iupdt를 실행합니다.

Tip

FixPack을 제거하려면

./installFixPack -f level -b <db2 엔진경로>를 이용하여 이전 FixPack 이미지로Overwrite하여 설치합니다.

1 root 사용자로 로그인하여 압축된 FixPack 파일이 저장된 임시 디렉토리로 이동합니다.

```
$ login root
$ cd /product
```

2 gzip과 tar 명령어로 압축을 해제합니다.

```
$ gzip v9.5fp4_aix_server.tar.gz
$ tar -xf v9.5fp4_aix_server.tar
```

3 InstallFixPack 명령을 이용하여 FixPack을 적용합니다. –b 옵션을 이용하여 해당 DB2 엔진 경로에 대해서 FP를 적용할 수 있습니다. 즉, 이는 DB2 제품에 대해 서로 다른 FixPack가 존재할 수 있음을 나타냅니다.

```
$ ./installFixPack –b <DB2 엔진 설치 경로 : /opt/ibm/db2/V9.7>
```

4 적용이 완료되면, db2ls 로 FixPack설치를 확인 합니다.

```
$./db2ls –q -a
```

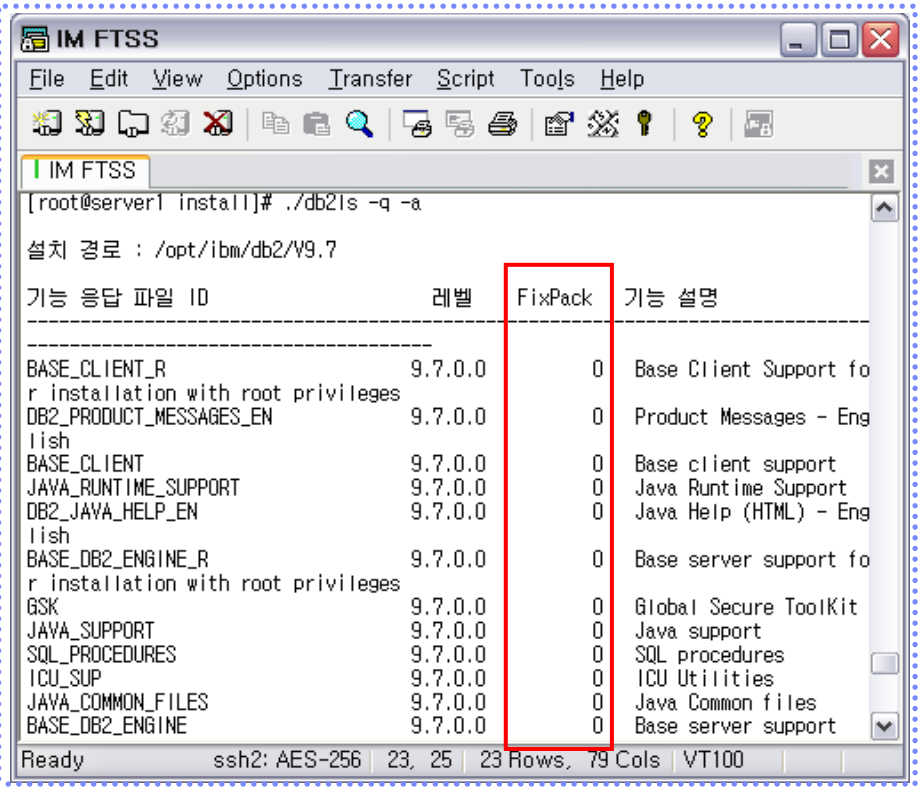


Figure 0208A ··· installFixPack 명령어로 FP 적용

TOPIC 02 09 라이센스 등록

Point

설치가 완료되고, 인스턴스를 생성한 후에는 db2licm 명령어를 이용하여 DB2 9.7 라이센스 파일을 적용해야 합니다. DB2 Enterprise Server Edition의 제품 키워드는 DB2ESE 입니다.

Tip

라이센스 파일 적용은 인스턴스 생성 후에 하며, 한 번만 적용하면 됩니다.

Tip

라이센스 파일을 적용하지 않으면 시험 사용 기간인 90일 동안 사용이 가능합니다.

1 root 사용자로 로그인하여 db2licm 명령어가 있는 디렉토리로 이동합니다.

```
$ login root
$ cd /opt/ibm/db2/V9.7/adm
```

3 라이센스 파일은 설치용 임시 디렉토리에 있습니다.

```
$ cd /opt/ibm/db2/V9.7/adm
$ ./db2licm -a /product/server/db2/license/db2ese.lic
```

4 적용된 라이센스 정보를 확인 후 적용합니다.

```
$ ./db2licm –l
$ ./db2licm –a ./db2ese.lic
```

5 DB2 9.7 라이센스를 제거 하려면 db2licm –r 옵션을 사용합니다.

```
$ ./db2licm –r db2ese
$ ./db2licm -l
```

5 라이센스 파일이 적절하지 않으면, 인스턴스를 시작할 때 경고가 기록됩니다.

```
$ cat <인스턴스 홈 디렉토리>/sqllib/db2dump/db2diag.log
```

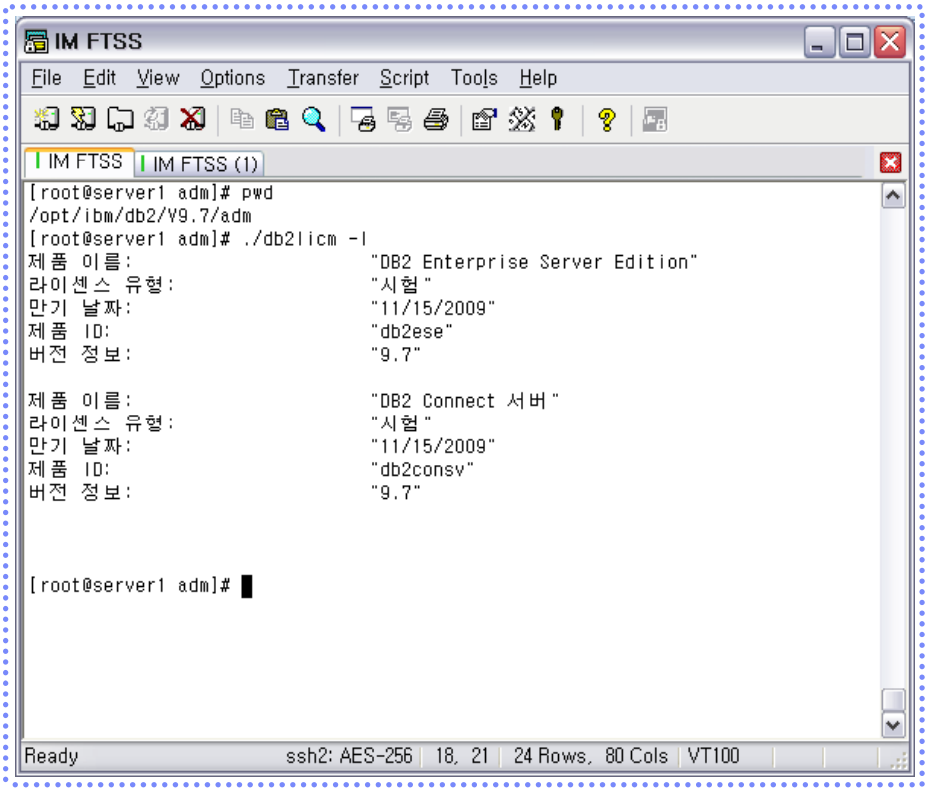


Figure 0209A ••• 라이센스 등록

TOPIC 02-10 DB2 9.7 업그레이드

Point

기존에 사용하고 있는 DB2 8.2, 9.1, 9.5를 DB2 9.7로 업그레이드가 가능합니다.

Tip

DB2 9.5, 9.1 및 DB2 버전 8에서부터 업그레이드가 가능합니다. DB2 버전 7이하 제품들은 중간 경유가 필요 합니다. (V7 -> V8.2-> V9.7)

1 DB2 9.7 사본을 설치합니다.

```
[root@localhost server]# ./db2_install
```

2 인스턴스를 업그레이드 합니다. (root 계정으로 수행)

```
[root@server1 instance]./db2iupgrade db2inst1
db2ckupgrade가 완료되었습니다. 데이터베이스를 업그레이드 할 수 있습니다.
```

3 데이터베이스를 업그레이드 합니다. (기존 사용 계정으로 수행)

Tip

업그레이드 작업시 루트 접근 권한이 있어야 합니다.

```
[db2inst1@server1 ~]$ db2 upgrade db sample
DB20000I UPGRADE DATABASE 명령이 완료되었습니다.
```

4 데이터베이스에 연결해서 업그레이드가 잘 성공적인지 확인합니다.

Tip

업그레이드 작업을 시작하기 전에 반드시 백업 이미지를 받으십시오.

```
[db2inst1@server1 ~]$ id
Uid=998(db2inst1) gid=998(db2gp) groups=998(db2gp)
[db2inst1@server1 ~]$ db2level
DB21085I  Instance “db2inst1" uses “32" 비트와 레벨 ID “08010107“의 DB2
코드 릴리스 “SQL09070”을(를) 사용합니다.
정보용 토큰은 "DB2 v9.7.0.0", "s090521", “LINUXIA3297“ 및 Fix Pack
"0"입니다.
제품이 "/opt/IBM/db2/V9.7“에 설치되었습니다.

[db2inst1@server1 ~]$ db2 connect to sample

  데이터베이스 연결 정보

데이터베이스 서버        = DB2/LINUX 9.7.0
SQL 권한 부여 ID         = DB2INST1
로컬 데이터베이스 별명   = SAMPLE
```

Tip

업그레이드 작업 전후로 dbm 및 db cfg의 값의 변화유무 확인을 포함한 기타 확인 작업이 필요 합니다. 자세한 사항은 아래 URL을 참고 하십시오.
http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.qb.upgrade.doc/doc/c0023662.html

5 업그레이드 이후 작업

- DB2 환경변수 재 조정 (디폴트 값 변경 확인)
- 통계 정보 갱신 (인덱스 통계 정보 갱신)
- 이벤트 모니터 재 작성
- HADR 복제 초기화
- JAVA 외부 루틴 매개변수 설정
- 업그레이드된 데이터베이스에 패키지 리바인드 작업
- 버전 별 신기능 적용 (Large Tablespace, Partitioning, Compression등)

UNIT 03

명령행 처리기

DB2가 제공하는 명령어와 SQL문은 DB2 명령창을 이용하여 대화식 또는 비대화식 모드로 실행합니다. DB2 명령어, SQLCODE, SQLSTATE에 대한 온라인 도움말을 이용하여 편리하게 DB2에 접근할 수 있으며, 다양한 명령 옵션과 파일을 이용한 처리를 지원합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- Windows용 DB2 명령행 처리기
- Windows용 DB2 명령창
- UNIX용 터미널 세션
- CLPPLUS
- DB2 명령어
- DB2 명령어 옵션
- 입력 파일을 이용한 처리 방법
- DB2 온라인 도움말

TOPIC 03 01 Windows용 DB2 명령행 처리기

Point

DB2 명령행 처리기는『db2=> 』라는 전용 프롬프트에서 텍스트 기반으로 DB2 명령어와 SQL문을 실행하고, !(느낌표)와 함께 OS명령어를 실행합니다.『시작 → 모든 프로그램 → IBM DB2 → 명령행 도구 → 명령행 처리기』메뉴를 선택합니다.

1. DB2 명령행 처리기 세션은 다음의 방법으로 실행합니다.

   시작 → 모든 프로그램 → IBM DB2 → 명령행 도구 → 명령행 처리기

2. 세션이 시작되면 db2=> 라는 전용 프롬프트가 표시됩니다.

```
db2=>
```

3. DB2 전용 프롬프트에서 DB2 명령어를 실행합니다. DB2 엔진은 기동되어 있어야 합니다.

```
db2=> <DB2 명령어>
```

Tip

SQL문을 실행하려면, connect 명령어를 이용하여 데이터베이스에 접속해야 합니다.

4. DB2 전용 프롬프트에서 SQL문을 실행합니다.

```
db2=> connect <데이터베이스명>
db2=> <SQL문>
```

5. DB2 전용 환경이므로 OS 명령어는 ! (느낌표 부호)와 함께 입력합니다.

```
db2=> !<OS 명령어>
```

6. 세션을 완전히 종료하려면, terminate 명령어로 DB2 명령행 처리기를 종료합니다.

```
db2=> terminate
```

Tip

이러한 방식을 '대화식 모드'라고 합니다. 내부적으로 db2bp.exe와 db2.exe가 실행됩니다.

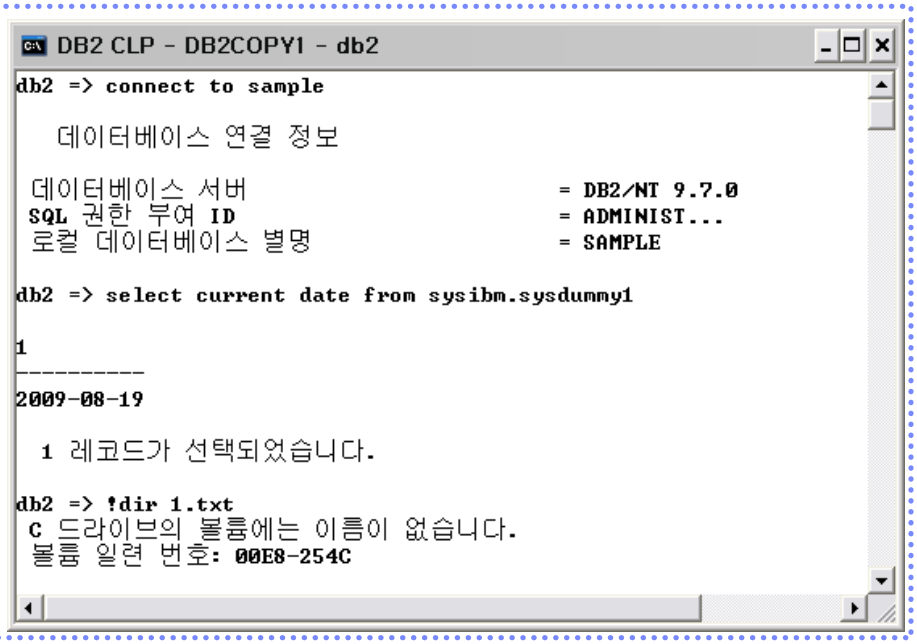


Figure 0301A ··· Windows의 DB2 명령행 처리기 세션

TOPIC 03 02 Windows용 DB2 명령창

Point

DB2 명령창은 Windows의 OS 명령창에서 텍스트 기반으로 DB2 명령어와 SQL문을 실행할 수 있는 환경입니다. db2라는 명령어를 이용하여 실행하며, 『시작 → 모든 프로그램 → IBM DB2 → 명령행 도구 → 명령창』 메뉴로 선택합니다.

Tip

Windows의 OS 명령창에는 DB2를 위한 환경이 설정되지 않았으므로 DB2 명령어를 실행할 수 없습니다.

1 DB2 명령창(CLP)세션은 다음의 두 가지 방법으로 실행합니다.

```
시작 → 모든 프로그램 → IBM DB2 → 명령행 도구 → 명령창
시작 → 실행 → db2cmd → 엔터키
```

2 DB2 명령어를 실행하려면, db2 라는 명령어를 이용합니다.

```
C:\> db2start
C:\> db2 <DB2 명령어>
```

Tip

SQL문을 실행하려면, connect 명령어를 이용하여 데이터베이스에 접속해야 합니다.

3 SQL문을 실행하려면, db2 라는 명령어를 이용합니다.

```
C:\> db2 connect <데이터베이스명>
C:\> db2 "<SQL문>"
```

4 MS 프롬프트와 동일한 환경을 제공하므로 OS 명령어도 실행할 수 있습니다.

```
C:\> <OS 명령어>
```

5 세션을 완전히 종료하려면, exit 명령어로 DB2 명령창을 종료합니다.

```
C:\> exit
```

Tip

이러한 방식을 '비대화식 모드'라고 합니다. db2 는 명령어로서 내부적으로 db2bp.exe가 실행됩니다.

Tip

SQL문은 "(쌍따옴표)로 묶는 것이 좋습니다.

Tip

'비대화식 모드'에서는 DB2 명령어 또는 SQL문을 실행할 때, | (파이프)와 >(리다이렉션) 등을 함께 이용할 수 있습니다.

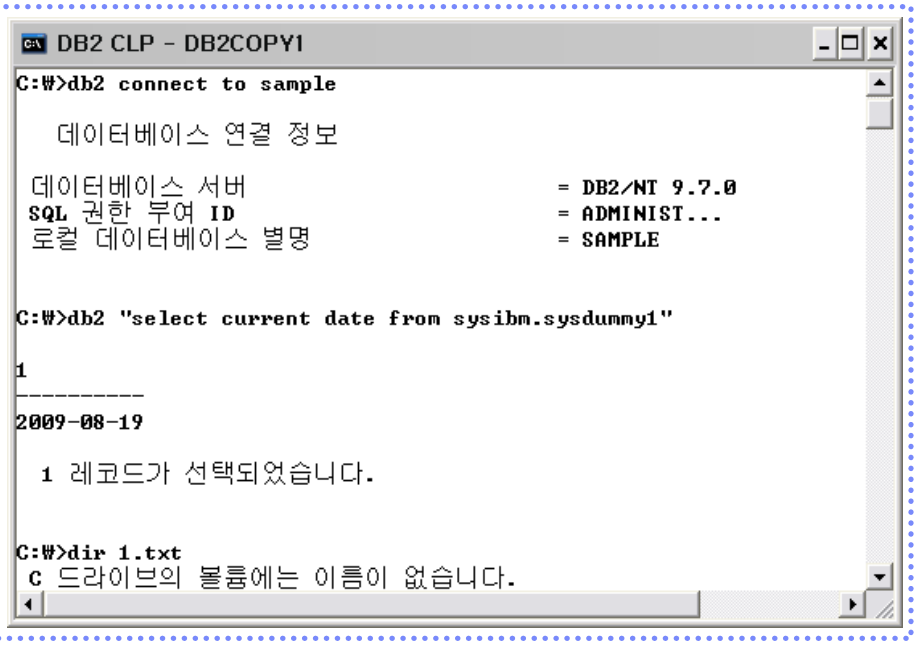


Figure 0302A Windows의 DB2 명령창 세션

TOPIC 03 03 UNIX용 터미널 세션

Point

UNIX 서버에서는 telnet 프로그램을 이용하여 터미널 세션을 열고, db2 라는 명령어로 DB2 세션을 실행합니다.

1 telnet 프로그램을 이용하여 서버의 OS 사용자 계정으로 로그인합니다.

```
C:\> telnet <원격 UNIX 서버의 IP 주소>
```

2 DB2 명령어와 SQL문을 사용하려면 DB2를 위한 환경을 설정해야 합니다.

```
$ . <DB2 인스턴스 사용자의 홈디렉토리>/sqllib/db2profile
$ echo $DB2INSTANCE
```

Tip

db2profile을 실행하는 명령을 사용자의 .profile에 포함시키면 로그인시에 자동으로 DB2를 위한 환경이 설정되므로 편리합니다.

3 Windows의 명령행 처리기과 동일한 대화식 방식으로 DB2 세션을 실행할 수 있습니다. quit 명령어를 이용하면 비대화식 모드로 전환합니다.

```
db2=> <db2 명령어>
db2=> <SQL문>
db2=> !<OS 명령어>
db2=> quit
```

Tip

대화식 또는 비대화식 모드에서 설정된 데이터베이스 접속은 실행 모드가 전환되어도 유지됩니다.

4 Windows의 명령창과 동일한 비대화식 방식으로 DB2 세션을 실행할 수 있습니다. db2 명령어와 엔터키를 입력하면 대화식 모드로 전환됩니다.

```
$ db2 <DB2 명령어>
$ db2 <SQL문>
$ <OS 명령어>
$ db2 <엔터키>
```

Tip

terminate 명령어는 데이터베이스 접속을 해제하고 DB2 세션을 종료시킵니다. quit 명령어는 실행 모드만 전환하므로 구별해야 합니다.

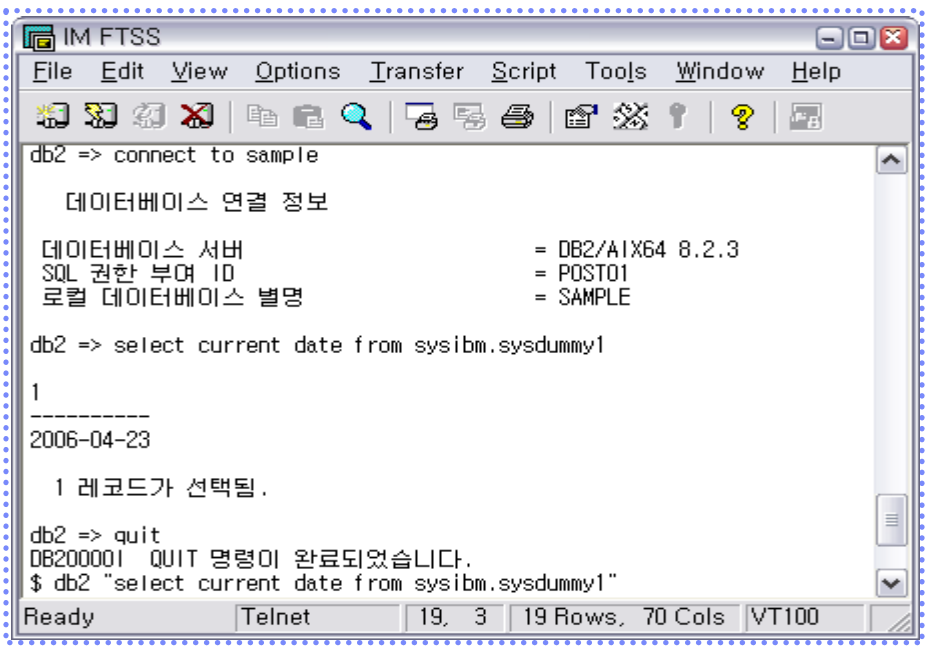


Figure 0303A ··· UNIX의 DB2 세션

TOPIC 03 04 CLPPLUS

Point

CLPPlus는 SQL문 및 데이터베이스 명령에 대해 사용하기 쉬운 새로운 대화식 명령행 처리기입니다. 이 환경은 SQL문 및 스크립트의 동적 작성, 편집 및 실행을 위해 제공됩니다

Tip

이 프로세서에서는 SQL*Plus 명령행 처리기에 호환 가능한 기능을 제공합니다 .

Tip

CLPPlus 인터페이스 환경에서 show all 명령어로 현재 구성값을 확인할 수 있습니다.

Tip

CPLPlus 의 SET 옵션 URL:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.swg.im.dbclient.clpplus.doc/doc/r0054132.html

Tip

CLPPlus 명령어 URL:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.swg.im.dbclient.clpplus.doc/doc/r0053830.html

1 CLPPlus 명령어 형식은 아래와 같습니다.

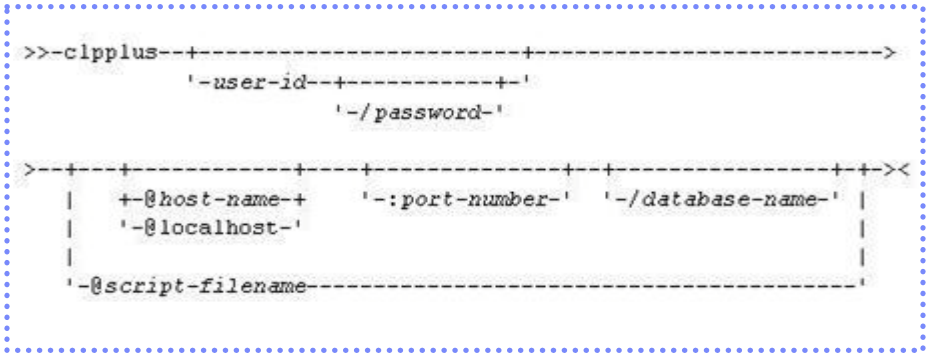
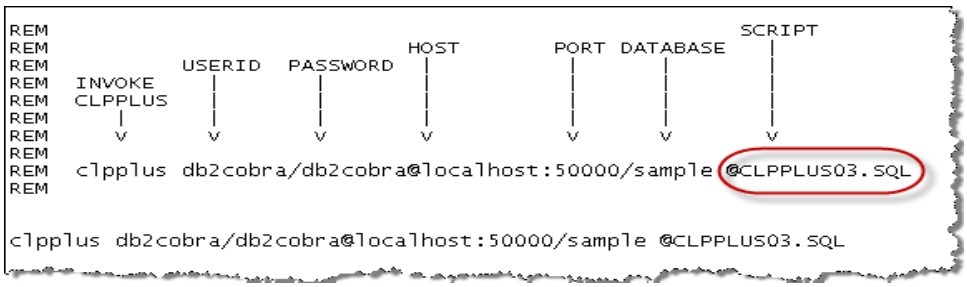


Figure 0304A • • • CLPPLUS 명령어에 대한 구문 도움말



2 명령창에 아래와 같은 명령어로 UID, PWD, Port, DB Name을 입력하여 database에 접속할 수 있습니다.

```
$ clpplus db2inst1/db2inst1@remote_host:50000/sample
```

3 CLPPlus 를 이용하여 사용자 스크립트를 실행할 수 도 있습니다.

```
$ cat > dept_query.sql
SET PAGESIZE 9999
SET ECHO ON
SELECT * FROM DEPT;
EXIT
```

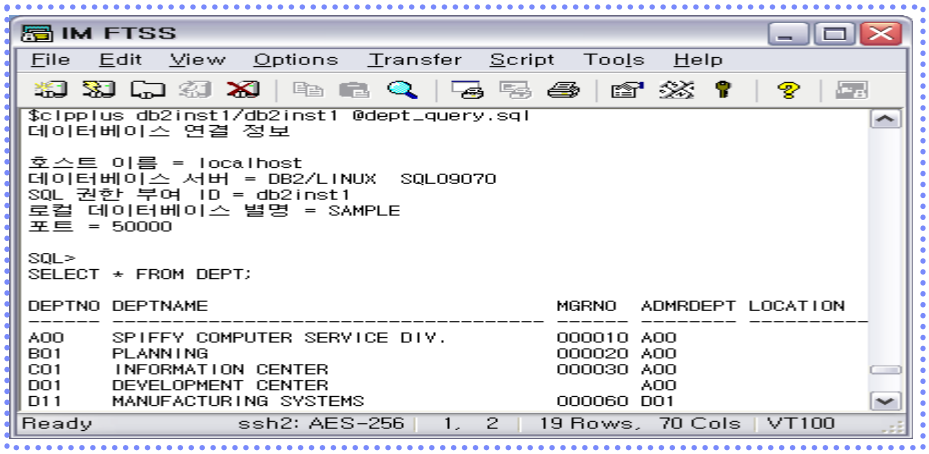


Figure 0304B • • • CLPPlus 사용 예제

TOPIC 03 05 DB2 명령어

Point

<DB2 명령어> 또는 <SQL문>을 실행시키기 위해 기본적으로 제공되는 프로그램입니다. db2라는 명령어를 실행하면 DB2 세션이 시작됩니다. CLP 세션이라고도 합니다.

Tip

db2 명령어를 실행하면 DB2 세션이 시작되고, 백그라운드 프로세스인 db2bp가 생성됩니다.

1. 비대화식 모드에서 db2 라는 명령어로 <DB2 명령어> 또는 <SQL문>을 실행합니다.

```
$ db2 <DB2 명령어>
$ db2 <SQL문>
$ <OS 명령어>
$ db2 <엔터키>
```

2. db2 명령어는 한 개 이상의 옵션을 지원합니다. – (마이너스)를 이용하여 옵션을 ON 시키고, +(플러스)를 이용하여 옵션을 OFF 시킵니다.

```
$ db2  –<옵션> <DB2 명령어>
$ db2  +<옵션> <DB2 명령어>
```

3. 제공되는 옵션의 목록과 현재 상태값은 다음과 같이 확인할 수 있습니다.

```
$ db2 list command options
```

4. ?(물음표)를 이용하여 온라인 도움말 기능을 이용할 수 있습니다.

```
$ db2 ? <DB2 명령어>
```

5. 특수 문자가 사용되는 경우에는 “ (쌍따옴표 부호)가 필요할 수도 있습니다.

```
$ db2 “SQL문”
```

Tip

quit 명령어는 ‘대화식 모드’에서 ‘비대화식 모드’로 실행 모드만 전환합니다. 내부적으로는 ‘비대화식 모드’에서 생성된 db2bp 프로세스를 계속 사용하므로, 새로운 DB2 세션을 시용하려면 terminate 명령을 입력하고 다시 db2를 입력합니다.

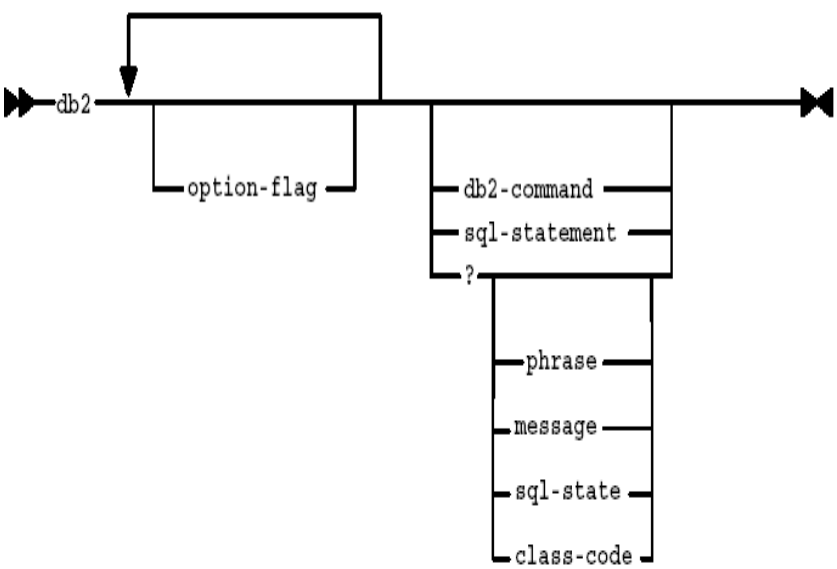


Figure 0305A db2 명령어

TOPIC 03 06 DB2 명령어 옵션

Point

대화식 모드의 DB2 세션은 update command options 명령어로 옵션을 변경하고, 비대화식 모드에서는 –(마이너스)와 +(플러스)를 이용하여 변경합니다. DB2OPTIONS 레지스터리 변수에 지정된 옵션은 모든 세션에 자동으로 적용됩니다.

Tip

자동 커미트 옵션이 기본적으로 설정되어 있으므로, SQL문은 즉시 커미트됩니다. 트랜잭션 처리를 원하면 +c 옵션을 이용합니다.

1 제공되는 옵션의 목록과 현재 설정 값은 다음과 같이 확인할 수 있습니다.

```
$ db2 list command options
```

2 대화식 모드에서 옵션의 상태 전환은 다음 명령어를 이용하여 제어합니다. 대화식 모드를 종료하면 옵션은 기본값으로 복귀합니다.

```
db2=> UPDATE COMMAND OPTIONS USING <옵션> ON
db2=> <DB2 명령어>
db2=> UPDATE COMMAND OPTIONS USING <옵션> OFF
db2=> <SQL문>
```

3 비대화식 모드에서는 – (마이너스)를 이용하여 옵션을 ON 시키고, +(플러스)를 이용하여 옵션을 OFF 시킵니다. 옵션은 실행 당시에만 유효합니다.

```
$ db2  –<옵션> <DB2 명령어>
$ db2  +<옵션> <DB2 명령어>
```

Tip

레지스터리 변수에서 지정한 옵션은 레지스터리 변수를 제거할 때까지 유지됩니다.

4 DB2 레지스터리 변수를 이용하여 옵션을 모든 대화식 또는 비대화식 세션에 자동으로 적용되도록 설정할 수 있습니다.

```
$ db2set DB2OPTIONS="-a +c"
$ db2 <DB2 명령어>
$ db2 <SQL문>
```

Tip

대화식 또는 비대화식 모드에서 지정한 옵션은 DB2OPTIONS 레지스터리 변수보다 우선적으로 적용됩니다.

```
옵션   설명                                  현재 설정값
------ ------------------------------------- ------------
  -a   SQLCA 표시                            OFF
  -c   자동 커미트                           ON
  -e   SQLCODE/SQLSTATE 표시                 OFF
  -f   입력 파일에서 읽기                    OFF
  -l   실행기록 파일에 명령 로그             OFF
  -n   줄 바꾸기 문자 제거                   OFF
  -o   출력 표시                             ON
  -p   대화식 입력 프롬프트 표시             ON
  -r   보고서 파일에 출력 저장               OFF
  -s   명령 오류시 실행 중지                 OFF
  -t   명령문 종료 문자 설정                 OFF
  -v   현재 명령 반향 출력                   OFF
  -w   FETCH/SELECT 경고 메시지 표시         ON
  -x   컬럼 헤더 인쇄 안함                   OFF
  -z   출력 파일에 모든 출력 저장            OFF
```

Figure 0306A • • • db2 명령어의 옵션

TOPIC 03 07 입력 파일을 이용한 처리 방법

Point

DB2 세션에서 한 개 이상의 명령문 또는 SQL문을 묶어서 실행하려면 비대화식 모드에서 입력 파일을 이용합니다. –svtf 옵션을 사용하는 것이 일반적입니다.

Tip

자동 커미트 옵션이 기본적으로 설정되어 있으므로, SQL문은 즉시 커미트됩니다.

Tip

COMMIT 또는 ROLLBACK 등의 SQL문을 이용하여 트랜잭션 처리를 하려면 +c 옵션을 이용합니다.

1 입력 파일은 OS가 제공하는 에디터를 이용하여 작성합니다.

```
$ vi  <파일명>
```

2 각 명령문은 기본적으로 한 줄 단위로 구별되므로, 한 명령문을 한 줄 이상에 걸쳐 표현하려면 ; (세미콜론 부호) 등의 명령문 구분자가 필요하며, 반드시 –t 옵션을 함께 사용해야 합니다. 주석문은 – – (대쉬 부호 2개)를 이용합니다.

```
– – <주석문>을 입력합니다.
create table t1
    (c1          int,
     c2          char(10));
Insert into t1 values (1,'A'),(2,'B');
select * from org where deptnumb > 10;
-- commit;
```

3 입력 파일명은 –f (file) 옵션을 이용하여 지정하며, ; (세미콜론 부호) 를 명령문 구분자로 사용하면 –t 옵션을 함께 사용해야 합니다. –s (stop)는 입력 파일의 실행 도중에 오류가 발생하면 실행을 중단하게 합니다. –v (verify) 옵션은 실행한 명령어와 결과를 함께 보여주게 합니다.

```
$ db2 –svtf xx.db2
```

4 옵션은 개별적으로 지정할 수도 있습니다.

```
$ db2 –s  -v  -t  -f xx.db2
```

Edit create.tab

```
-- comment:   db2 -svtf create.tab


connect to sample;

create table tab3
   (name varchar(20) not null,
    phone char(40),
    salary dec(7,2));

select * from tab3;

commit work;

connect reset;
```

db2 -svtf create.tab

Figure 0307A ••• 입력 파일을 이용할 때의 옵션

TOPIC 03 08 DB2 온라인 도움말

Point

DB2 세션에서 ?(물음표)를 사용하여, DB2 명령어에 대한 구문과 SQLCODE, SQLSTATE 값에 대한 텍스트 기반의 도움말을 확인할 수 있습니다.

Tip

SQL문에 대한 도움말은 제공하지 않습니다. 자세한 도움말은 온라인 정보 센터와 제품 설명서를 참조하십시오.

Tip

?(물음표)를 특수 문자로 인식하는 OS 에서는 비대화식 모드에서 "(쌍따옴표)와 함께 사용합니다.

1. DB2 세션에서 제공하는 모든 명령어의 목록은 다음과 같이 확인합니다.

```
$ db2 ?
```

2. 특정한 DB2 명령어에 대한 구문과 옵션은 다음과 같이 확인합니다.

```
$ db2 ? <DB2 명령어>
```

3. SQLCODE에 대한 설명은 다음과 같이 확인할 수 있습니다. SQLCODE는 SQLnnnn 의 형식으로 표시되며, nnnn은 숫자입니다.

```
$ db2 ? SQLnnnn
```

4. SQLSTATE에 대한 설명은 다음과 같이 확인할 수 있습니다. SQLSTATE는 nnnnn 형식으로 표시되는 5자리의 숫자입니다.

```
$ db2 "? nnnnn"
```

5. 대화식 모드에서도 도움말을 확인할 수 있습니다.

```
db2=> ?
db2=> ? <DB2 명령어>
db2=> ? SQLnnnn
db2=> ? nnnnn
```

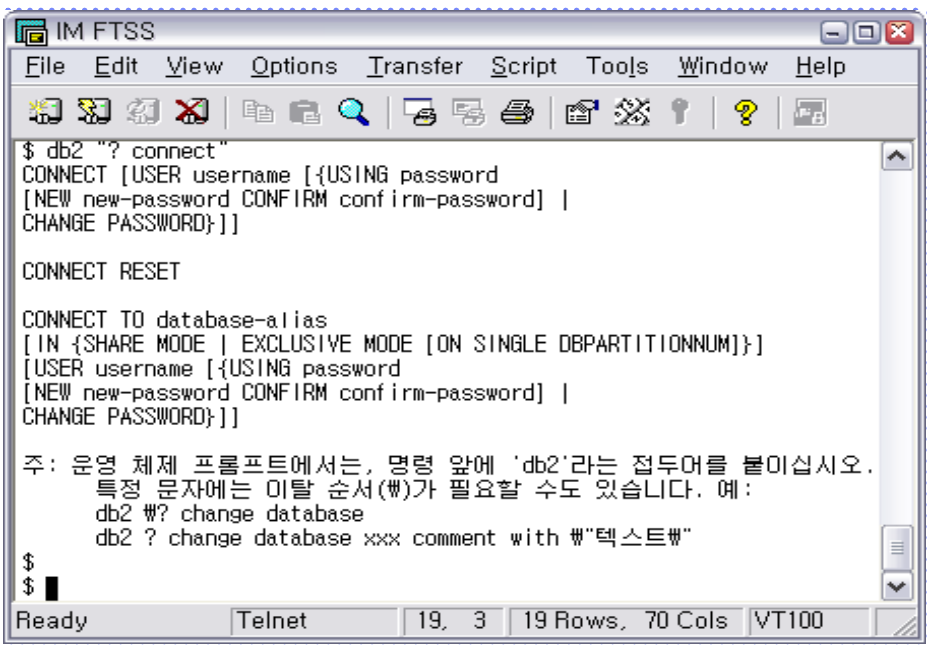


Figure 0308A ··· connect 명령어에 대한 구문 도움말

UNIT 04

인스턴스

DB2 엔진의 기능을 사용할 수 있는 논리적인 환경을 DB2 인스턴스라고 합니다. 한 서버에서 독립적인 환경을 가지는 한 개 이상의 인스턴스를 운영할 수 있습니다. Linux 및 Unix서버에서 인스턴스는 특정한 사용자 계정과 연관되며, db2icrt 명령어를 이용하여 root 사용자가 생성합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 인스턴스 개요
- 인스턴스 생성
- 인스턴스 갱신
- 인스턴스 제거
- 인스턴스 기동
- 인스턴스 중지
- 인스턴스 강제 중지
- 인스턴스 구성 파일
- 인스턴스 지정 방법
- DB2 레지스터리 변수
- DB2 관리 서버
- 원격 클라이언트 지원
- DPF 환경 설정

TOPIC 0401 인스턴스 개요

Point

DB2 엔진의 기능을 사용할 수 있는 논리적인 환경을 DB2 인스턴스라고 합니다. 한 서버에 여러 개의 인스턴스를 생성하여 독립적인 DB2 엔진으로 운영할 수 있습니다. LINUX/UNIX에서 인스턴스는 OS 사용자 계정과 연관됩니다.

Tip

데이터베이스 관리자를 간단히 DBM 이라고 합니다.

Tip

인스턴스 수준에서 설정하는 환경 변수는 db2 registry (db2set) 정보와 dbm cfg 입니다.

Tip

한 서버에 여러 개의 인스턴스를 생성하는 경우에도 DB2 제품은 한번만 설치 합니다.

1 인스턴스는 DB2 엔진의 기능을 사용할 수 있는 논리적인 환경으로서 DB2 데이터베이스 관리자라고 합니다.

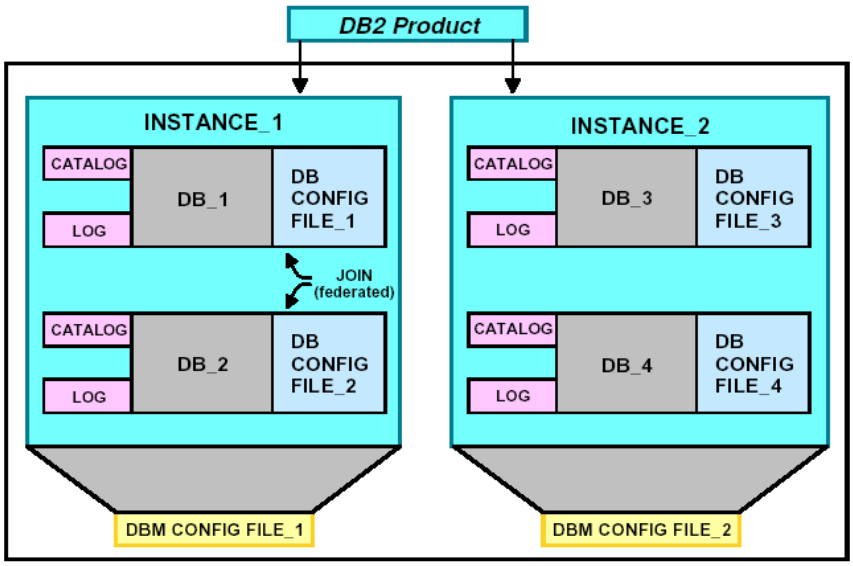


Figure 0401A DB2 인스턴스

2 DB2 엔진의 기능을 사용하고, 데이터베이스를 구축하려면 DB2 서버 제품을 설치한 후에 DB2 인스턴스를 생성해야 합니다. UNIX에서 인스턴스는 OS 사용자 계정과 연관됩니다. Windows 에서는 인스턴스와 OS의 사용자 계정이 연관되지 않습니다.

3 한 서버에 한 개 이상의 인스턴스를 생성할 수 있으며, 각 인스턴스는 독립적인 DB2 엔진으로 운영되므로 개발용 인스턴스와 운영용 인스턴스를 따로 생성할 수 있습니다.

4 각 인스턴스는 고유의 환경을 구성하는 인스턴스 구성 파일을 가지게 됩니다. 인스턴스 구성 파일에는 다양한 인스턴스 구성 변수들이 있습니다.

5 동일한 서버에 존재하는 DB2 인스턴스를 지역 인스턴스라고 하고, 원격 서버에 존재하는 DB2 인스턴스를 원격 인스턴스라고 합니다.

6 클라이언트와 통신을 위한 프로토콜 및 서비스 포트의 지정은 인스턴스 수준에서 정의됩니다.

TOPIC 04-02 인스턴스 생성

Point

root 사용자로 로그인하여 db2icrt 명령어를 이용하여 한 개 이상의 인스턴스를 생성합니다. 인스턴스 사용자로 사용될 OS 사용자 계정과 그룹이 필요합니다.

Tip

인스턴스 사용자 계정의 일차 그룹은 고유한 그룹명으로 지정하는 것을 권장합니다.

1 Linux 및 UNIX에서 인스턴스를 생성하려면, OS 그룹과 사용자 계정이 필요합니다.

```
$ login root
$ mkgroup <일차 그룹명>
$ mkuser pgrp=<일차 그룹명> <인스턴스 사용자명>
$ passwd <인스턴스 사용자명>
```

Tip

<fenced 사용자명>은 <인스턴스명> 과 동일하게 설정 가능하나, 안전한 저장 프로시저의 사용을 위해 별도의 Fenced 사용자를 정의하여 사용하기를 권장합니다.

2 DB2 제품이 설치된 디렉토리에 있는 db2icrt 명령어를 이용하여 생성합니다.

```
$ cd /opt/ibm/db2/V9.7/instance
$ ./db2icrt  -u <fenced 사용자명>  <인스턴스명>
```

3 인스턴스 사용자의 홈디렉토리에는 sqllib 라는 서브 디렉토리가 생성되어 DB2 9.7 엔진에서 제공하는 실행 모듈들이 링크됩니다.

```
$ ls <인스턴스 사용자의 홈디렉토리>
```

4 /etc/services 파일에 인스턴스와 관련된 4개의 포트가 추가됩니다.

```
$ grep <인스턴스명> /etc/services
DB2_<인스턴스명>60000/tcp
DB2_<인스턴스명>_1              60001/tcp
DB2_<인스턴스명>_2              60002/tcp
DB2_<인스턴스명>_END            60003/tcp
```

Tip

DB2 인스턴스와 관련된 TCP/IP 포트의 번호는 변경할 수 있습니다.

Tip

Windows 에서는 설치 시에 DB2 라는 이름의 인스턴스가 자동으로 생성됩니다. 인스턴스를 추가로 생성하는 것이 가능하지만, 기본 인스턴스인 DB2를 사용하는 것을 권장합니다.

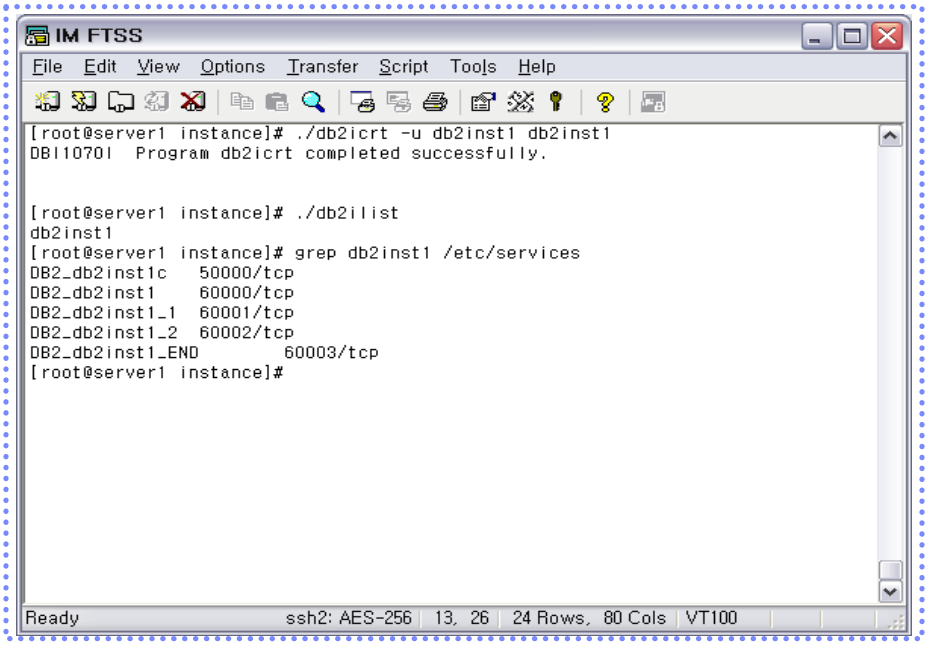


Figure 0402A • • • db2icrt 명령어

TOPIC 04-03 인스턴스 갱신

Point

root 사용자가 db2iupt 명령어를 이용하여 기존 인스턴스의 환경을 갱신합니다. FixPack을 적용한 후에 인스턴스의 라이브러리 링크를 갱신하고자 할 때 사용합니다.

Tip

기존 인스턴스에서 사용자가 구성한 환경은 유지됩니다.

Tip

기존 인스턴스에서 생성한 데이터베이스도 유효합니다.

1. 인스턴스의 환경을 갱신하기 위해서는 먼저 인스턴스를 중지해야 합니다.

```
$ login <인스턴스 사용자명>
$ db2stop force
```

2. root 사용자가 db2iupdt 명령어를 이용하여 인스턴스의 환경을 갱신합니다.

```
$ login root
$ cd /opt/ibm/db2/V9.7/instance
$ ./db2iupdt <인스턴스명>
$ ./db2ilist
```

3. 복수개의 코드 레벨(DB2 엔진)이 존재 할 경우, 한 경로의 상위 코드 레벨에서 다른 경로에 설치된 낮은 코드 레벨로 인스턴스를 이동합니다

```
$ login root
$ cd /opt/ibm/db2/V9.7/instance
$ ./db2iupdt  -D db2inst1
```

4. 인스턴스 사용자가 db2start 명령어로 인스턴스를 재 기동하고, DB2 제품 정보를 확인합니다.

```
$ login <인스턴스 사용자명>
$ db2start
$ db2level
```

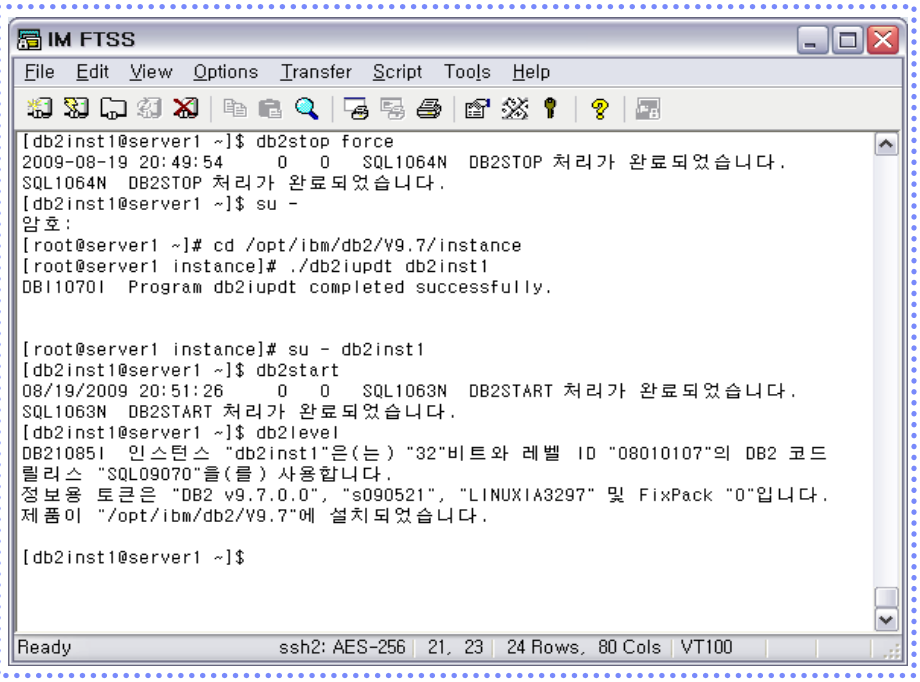


Figure 0403A db2iupdt 명령어

TOPIC 0404 인스턴스 제거

Point

root 사용자가 db2idrop 명령어를 이용하여 불필요한 인스턴스를 제거합니다. 인스턴스를 제거하면, DB2 엔진 기능을 사용할 수 없으며, 인스턴스 사용자는 DB2와는 무관한 일반 사용자가 됩니다.

1 인스턴스를 제거하기 위해서는 먼저 인스턴스를 중지해야 합니다.

```
$ login <인스턴스 사용자명>
$ db2stop force
```

Tip

인스턴스를 제거해도 데이터베이스 설치 파일들은 제거되지 않으며, 사용했던 데이터베이스도 그대로 유지됩니다. 따라서 인스턴스를 다시 생성하여 데이터베이스를 재 카탈로그하면 기존의 데이터베이스를 다시 사용할 수 있습니다.

2 root 사용자가 db2idrop 명령어를 이용하여 인스턴스를 제거합니다.

```
$ login root
$ cd /opt/ibm/db2/V9.7/instance
$ ./db2idrop <인스턴스명>
$ ./db2ilist | grep <인스턴스명>
```

3 인스턴스 사용자의 홈 디렉토리에 존재하던 sqllib 라는 서브 디렉토리가 제거된 것을 확인할 수 있습니다.

```
$ ls <인스턴스 사용자의 홈디렉토리>/sqllib
```

Tip

/etc/services 파일에 추가된 인스턴스와 관련된 TCP/IP 포트 정보는 설치 후에도 제거되지 않습니다. 인스턴스를 제거한 후에 사용자가 직접 제거하십시오.

4 /etc/services 파일에 추가된 포트는 자동으로 제거되지 않습니다.

```
$ grep <인스턴스명> /etc/services
DB2_<인스턴스명>60000/tcp
DB2_<인스턴스명>_1              60001/tcp
DB2_<인스턴스명>_2              60002/tcp
DB2_<인스턴스명>_END            60003/tcp
```

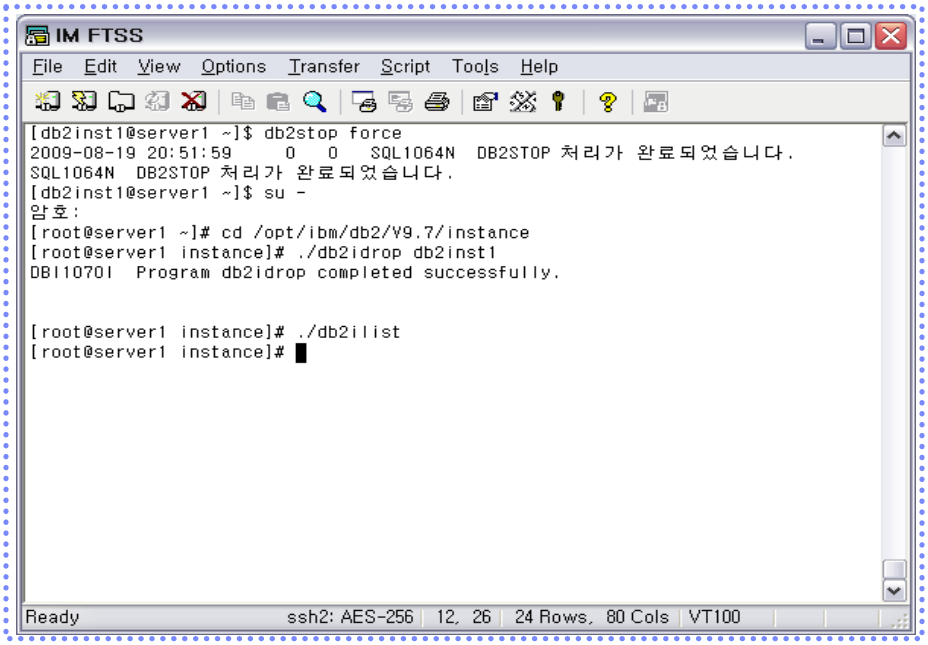


Figure 0404A db2idrop 명령어

TOPIC 04-05 인스턴스 시작

Point

인스턴스 사용자는 db2start 명령어를 이용하여 인스턴스를 기동합니다. 인스턴스가 성공적으로 기동되면, db2sysc를 비롯한 여러 개의 프로세스 및 스레드가 생성되고, 인스턴스 수준의 공유 메모리가 할당됩니다.

Tip

단순히 db2 또는 db2bp라는 이름을 가진 프로세스는 DB2 세션을 위한 프로세스입니다. db2 라는 글자로 시작하지만, DB2 엔진과는 무관합니다.

Tip

DB2 9.5부터는 프로세스 방식이 아닌 스레드 방식을 이용합니다.

db2pd -edu 로 DB2 엔진과 관련된 스레드를 확인할 수 있습니다.

1. 인스턴스 사용자로 로그인합니다.

```
$ login <인스턴스 사용자명>
```

2. 인스턴스 기능을 시작하려면 db2start 명령어를 이용합니다.

```
$ db2start
```

3. db2sysc 를 비롯한 여러 개의 프로세스가 생성됩니다. DB2 엔진과 관련된 프로세스의 이름은 대부분 db2 라는 글자로 시작됩니다. 프로세스 이름의 마지막 부분에 나오는 숫자는 데이터베이스 파티션 번호입니다. DPF인 경우에는 동일한 엔진 프로세스가 데이터베이스 파티션별로 생성됩니다.

```
$ ps –ef | grep <인스턴스 사용자명>
$ db2pd -edu
```

4. 인스턴스 공유 메모리가 할당된 것을 확인할 수 있습니다.

```
$ db2mtrk –i
인스턴스용 메모리
   monh       other
   16.0K      1.3M
```

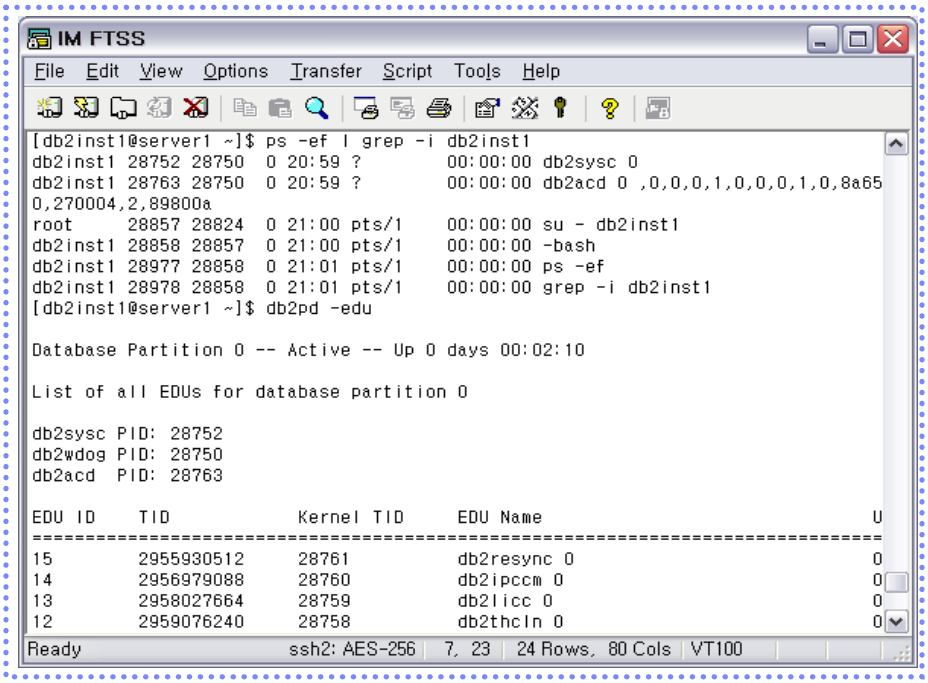


Figure 0405A ··· db2start 명령어

TOPIC 04-06 인스턴스 중지

Point

인스턴스 사용자는 db2stop 명령어를 이용하여 인스턴스를 중지시킬 수 있습니다. 먼저 데이터베이스에 접속된 모든 응용프로그램들은 중지되어야 합니다. 인스턴스와 관련된 모든 프로세스가 제거되고, 인스턴스 수준의 공유 메모리도 해제됩니다.

Tip

데이터베이스에 접속된 응용프로그램이 존재하면 SQL1025N 오류가 반환되고, 인스턴스는 중지되지 않습니다.

Tip

db2stop 명령어가 성공적으로 완료된 이후에도 인스턴스 사용자와 관련된 DB2 프로세스가 비정상적으로 남아있다면, 인스턴스 강제 중지를 실행합니다. (db2_kill 명령어)

1. 인스턴스 중지는 인스턴스 사용자가 담당합니다.

```
$ login <인스턴스 사용자명>
```

2. 인스턴스를 중지하기 전에 데이터베이스에 접속된 응용프로그램을 모두 중지시켜야 합니다. db2stop 명령어에서 force 옵션을 이용하면, 접속된 응용프로그램을 모두 강제로 중지하고 인스턴스를 중지하게 합니다.

```
$ db2stop force
```

3. 인스턴스가 중지되면, 인스턴스 사용자의 DB2 프로세스는 모두 제거됩니다.

```
$ ps –ef | grep <인스턴스 사용자명>
$ db2pd -edu
```

4. db2bp 라는 이름의 프로세스는 db2stop 명령어를 실행해도 제거되지 않습니다. db2bp는 DB2 세션용 프로그램을 위한 별도의 프로세스이므로 kill 명령어로 제거해도 무방합니다.

```
$ kill -9 <db2bp 프로세스의 pid>
```

5. 인스턴스 공유 메모리가 해제된 것을 확인할 수 있습니다.

```
$ db2mtrk –i
인스턴스가 시작되지 않음.
```

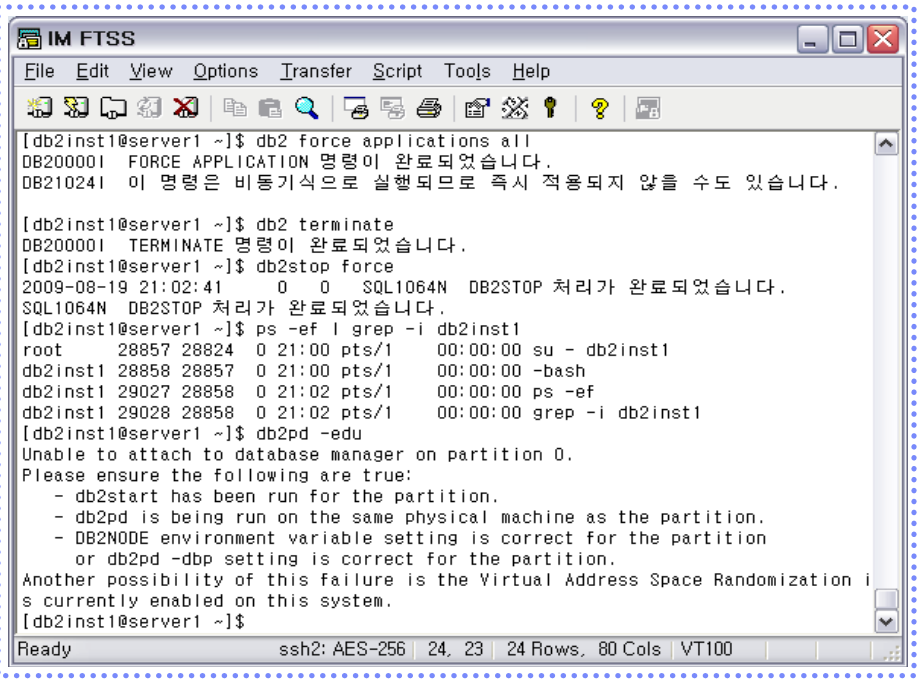


Figure 0406A • db2stop 명령어

TOPIC 0407 인스턴스 강제 중지

Point

db2stop 명령어를 이용하여 인스턴스를 정상적으로 중지할 수 없으면, db2_kill 명령어를 이용하여 강제로 중지할 수 있습니다. Ipclean 명령어와 ipcs 명령어를 이용하여 DB2가 사용한 리소스를 명시적으로 정리하도록 합니다.

Tip

db2_kill 은 비상시에 사용하는 명령어로 정상적인 경우에 사용하는 것은 권장하지 않습니다. 정상적으로 인스턴스를 중지할 수 있다면 db2stop 명령어를 사용합니다.

Tip

Ipclean 명령어를 이용한 후에도 DB2가 사용한 리소스가 남아있는 경우에는 ipcrm 명령어를 이용하여 제거합니다.

Tip

db2_kill 명령어를 실행한 후에도 DB2와 관련된 프로세스가 남아있는 경우에는 kill 명령어로 제거합니다.

1. 인스턴스 중지는 인스턴스 사용자가 담당합니다.

```
$ login <인스턴스 사용자명>
```

2. 인스턴스를 중지하기 전에 데이터베이스에 접속된 응용프로그램을 모두 중지시켜야 합니다.

```
$ db2 force applications all
```

3. db2_kill 명령을 사용하여 인스턴스를 강제로 중지시킵니다. 인스턴스를 비정상적으로 중지한 경우에는 ipclean 명령어를 이용하여 DB2가 사용하던 리소스를 정리하도록 합니다.

```
$ db2_kill
$ ipclean –a
$ ipcs –a | grep <인스턴스 사용자명>
```

4. 인스턴스 사용자의 DB2 프로세스가 모두 제거된 것을 확인합니다.

```
$ ps –ef | grep <인스턴스 사용자명>
$ db2pd -edu
```

5. db2start 명령어를 이용하여 인스턴스를 재 시작합니다.

```
$ db2start
```

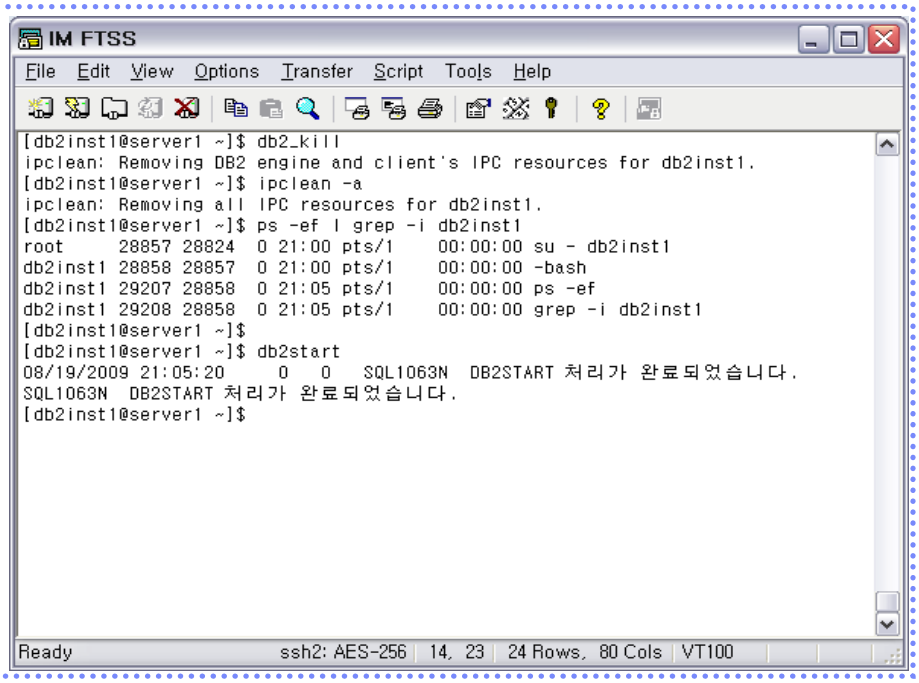


Figure 0407A db2_kill 명령어

TOPIC 0408 인스턴스 구성 파일

Point

인스턴스 구성 파일에 저장되는 인스턴스 구성 변수를 이용하여 인스턴스 별로 고유한 환경을 구성합니다. 인스턴스 구성 파일은 get dbm cfg, update dbm cfg, reset dbm cfg 명령어를 이용하여 인스턴스 사용자가 관리합니다.

Tip

인스턴스 구성 파일을 DBM 구성 파일이라고 하고, 인스턴스 구성 변수는 DBM 구성 변수라고도 합니다.

Tip

show detail 옵션은 attach 명령어로 인스턴스에 접속해야 사용할 수 있습니다.

Tip

값을 변경하는 즉시 반영되는 인스턴스 구성 변수를 '온라인 구성 가능한 인스턴스 구성 변수' 라고 합니다.

1 인스턴스 구성 파일은 인스턴스 사용자가 관리합니다.

```
$ login <인스턴스 사용자명>
```

2 get dbm cfg 명령어를 이용하여 인스턴스 구성 파일을 확인합니다. show detail 옵션을 이용하면 설정값과 현재값을 확인할 수 있습니다.

```
$ db2 attach to <인스턴스명>
$ db2 get dbm cfg show detail | grep <구성변수명>
$ db2 detach
```

3 인스턴스를 재 기동하지 않고 즉시 반영되는 구성 변수의 값을 변경할 때는 attach 명령어로 인스턴스에 접속하여 update dbm cfg 명령어를 사용합니다.

```
$ db2 attach to <인스턴스명>
$ db2 update dbm cfg using <구성변수명>  <값>
$ db2 detach
```

4 reset dbm cfg 명령어는 모든 인스턴스 구성 변수를 초기값으로 변경합니다. 일부 인스턴스 구성 변수는 인스턴스를 재 기동해야 변경한 값이 반영됩니다.

```
$ db2 reset dbm cfg
$ db2stop force
$ db2start
```

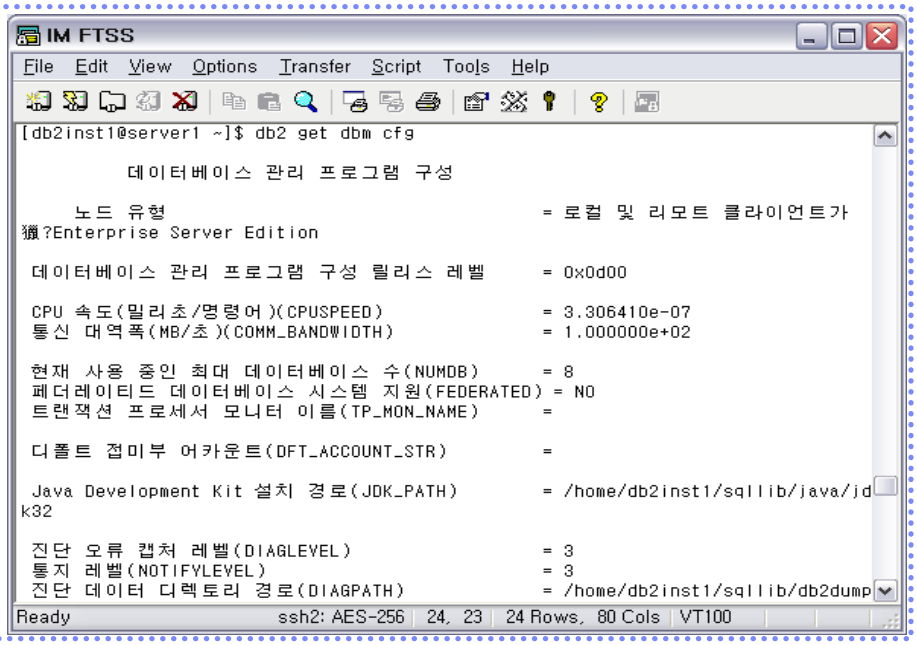


Figure 0408A ••• 인스턴스 구성 변수

TOPIC 0409 인스턴스 지정 방법

Point

DB2 사용자는 DB2INSTANCE라는 환경 변수를 이용하여 원하는 인스턴스를 선택할 수 있습니다. DB2 명령을 실행하여 인스턴스의 서비스를 받으려면 PATH 등의 기본적인 환경 변수의 설정이 필요합니다.

Tip

Windows 에서 인스턴스를 지정할 때에는 제어판 또는 DB2 명령 창을 이용하여 DB2INSTANCE 환경 변수를 설정합니다.

1. DB2 사용자로 로그인합니다.

```
$ login <DB2 사용자명>
```

2. DB2INSTANCE 라는 OS 환경 변수 명을 이용하여 인스턴스 명을 지정합니다.

```
$ export DB2INSTANCE=<인스턴스명>
$ echo $DB2INSTANCE
```

3. DB2 명령을 실행하려면 PATH 등의 기본적인 환경 변수 설정이 필요합니다.

```
$ export PATH=$PATH:<인스턴스 사용자 홈디렉토리>/sqllib/bin
```

4. DB2INSTANCE, PATH 등의 환경 변수를 간단하게 설정하려면, 인스턴스 사용자의 홈 디렉토리에 생성된 sqllib 라는 서브 디렉토리에 있는 db2profile 을 실행하면 됩니다.

```
$ . <인스턴스 사용자 홈디렉토리>/sqllib/db2profile
$ echo $DB2INSTANCE
```

5. 다른 인스턴스를 지정하려면, 원하는 인스턴스 사용자의 db2profile을 선택하여 실행합니다.

```
$ . <원하는 인스턴스 사용자 홈디렉토리>/sqllib/db2profile
$ echo $DB2INSTANCE
```

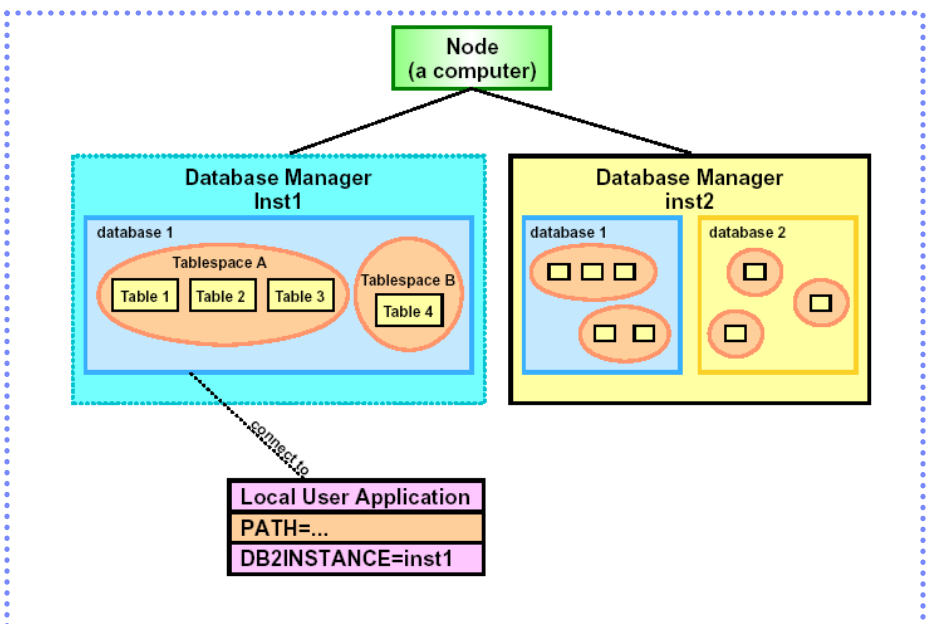


Figure 0409A DB2INSTANCE 환경 변수

TOPIC 04-10 DB2 레지스터리 변수

Point

DB2 인스턴스에만 적용되는 시스템 환경 변수를 DB2 레지스터리 변수라고 합니다. DB2 레지스터리 변수는 인스턴스 사용자가 db2set 명령어를 이용하여 관리하며, OS 환경 변수가 아니므로 시스템의 재기동이 필요하지 않습니다.

1. DB2 레지스터리 변수는 인스턴스 사용자가 관리합니다.

```
$ login <인스턴스 사용자>
```

2. db2set 명령어의 기본 형식은 db2set <레지스터리 변수명>=<값> 입니다.

```
$ db2set <레지스터리변수>=<값>
```

3. 레지스터리 변수의 설정을 취소하려면 <값> 항목을 입력하지 않으면 됩니다.

```
$ db2set <레지스터리변수>=
```

4. -i 옵션을 사용하면 특정한 인스턴스에만 변수가 적용되게 합니다. -g 옵션은 현재의 서버에 존재하는 모든 인스턴스에 대해 변수를 적용합니다.

```
$ db2set -i <인스턴스명> <레지스터리변수>=<값>
$ db2set -g <레지스터리변수>=<값>
$ db2set -all
```

5. 대부분의 레지스터리 변수는 설정 후에 인스턴스의 재기동이 필요합니다.

```
$ db2stop force
$ db2start
```

Tip

별도의 옵션이 없으면, 현재 세션이 지정한 인스턴스에만 적용됩니다.

Tip

레지스터리 변수의 값을 설정할 때는 = (등호) 전후에 공백 문자가 없도록 주의합니다.

Tip

지원되는 모든 레지스터리 변수는 db2set -lr 명령어로 확인 가능하며 그에 대한 자세한 설명은 정보센터를 참조하십시오.

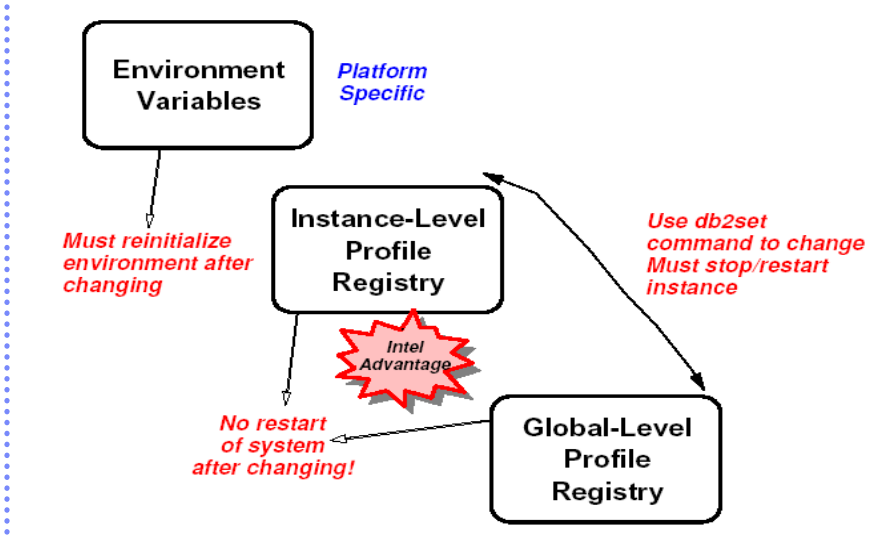


Figure 0410A DB2 레지스터리 변수

TOPIC 04-11 DB2 관리 서버

Point

제어 센터 등의 GUI 도구를 이용하여 DB2 를 운영할 때만 필요한 특수 인스턴스로서 한 서버에 한 개만 존재할 수 있습니다. dascrt, daslist, dasupdt, dasdrop 등의 명령어로 관리하며, db2admin 명령어로 시작되고 중지됩니다.

Tip

DB2 9.7부터 DAS가 Deprecate되었으며, Optim Database Administrator로 기능 이전되었습니다.

1 UNIX에서 DAS 인스턴스를 생성하려면 OS 그룹과 사용자 계정이 필요합니다.

```
$ login root
$ mkgroup <일차 그룹명>
$ mkuser pgrp=<일차 그룹명> <DAS용 인스턴스 사용자명>
$ passwd <DAS용 인스턴스 사용자명>
```

2 DB2 제품이 설치된 디렉토리에 있는 dascrt 명령어를 이용하여 생성합니다.

```
$ cd /opt/ibm/db2/V9.7/instance
$ ./dascrt  <DAS용 인스턴스명>
$ ./daslist
```

3 DAS용 인스턴스 사용자는 db2admin 명령어로 DAS를 시작하거나 중지합니다.

```
$ login <DAS 인스턴스명>
$ db2admin start
$ db2admin stop
```

Tip

DAS용 인스턴스 사용자 계정의 일차 그룹은 고유한 그룹명으로 지정하는 것이 권장됩니다.

4 DAS를 제거하려면 dasdrop 명령어를 이용합니다.

```
$ ./dasdrop dasusr1
```

Tip

DAS는 생성시에 자동으로 시작됩니다.

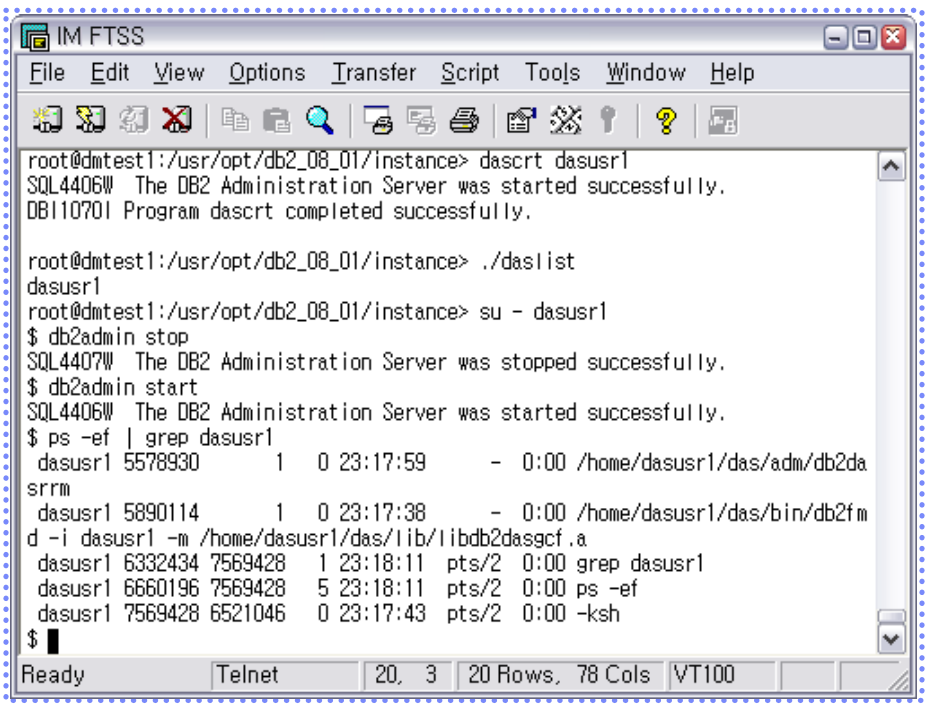


Figure 0411A DAS 생성과 기동

Tip

FixPack 적용 후에는 dasupdt 명령어를 이용하여 DAS를 갱신합니다.

Tip

Windows 에서는 설치 시에 DB2DAS00 라는 이름의 DAS용 인스턴스가 자동으로 생성됩니다.

TOPIC 04-12 원격 클라이언트 지원

Point

DB2 서버 제품은 2개의 연속적인 TCP/IP 서비스 포트를 이용하여 UNIX, Windows 등의 원격 클라이언트 제품으로부터의 접속을 지원합니다.

Tip

포트 번호는 50000번 이상이 권장되며, 서비스 명은 임의로 정의합니다.

1 원격 클라이언트와 통신하려면 /etc/services 파일에 연속된 번호를 가진 2개의 TCP/IP 포트를 추가합니다.

```
$ login root
$ grep <인스턴스명> /etc/services
<인스턴스명>c                    50000/tcp
<인스턴스명>i                    50001/tcp
```

2 DB2 레지스터리 변수인 DB2COMM에 TCPIP를 지정합니다.

```
$ login <인스턴스 사용자>
$ db2set DB2COMM=TCPIP
```

Tip

SVCENAME 구성 변수에 <서비스명> 대신에 <포트 번호>를 바로 사용할 수도 있지만, /etc/services 파일을 사용하는 것을 권장합니다.

3 /etc/services 파일에서 지정한 첫 번째 포트의 서비스 명을 DBM 구성 변수인 SVCENAME 에 지정하고, 인스턴스를 재 기동합니다.

```
$ db2 update dbm cfg using SVCENAME <서비스 명>
$ db2stop force
$ db2start
```

4 TCP/IP용 통신 리스너 프로세스인 db2tcpcm이 생성된 것을 확인합니다.

```
$ db2pd -edu | grep -I db2tcpcm
```

Tip

Personal Edition은 클라이언트를 지원하지 않습니다.

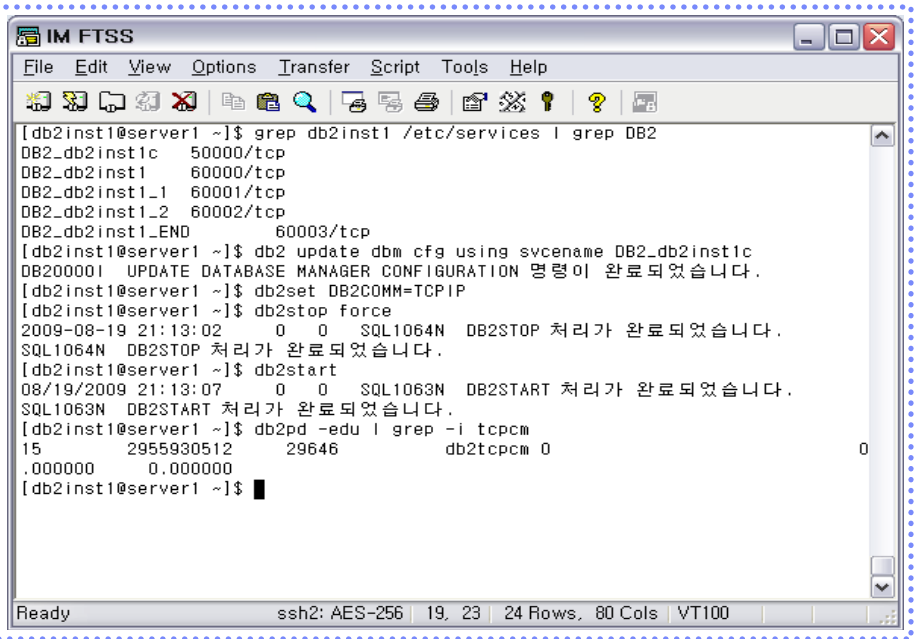


Figure 0412A ••• 원격 클라이언트를 위한 TCP/IP 포트 설정

TOPIC 04-13 DPF 환경 설정

Point

DB2 인스턴스는 기본적으로 한 개의 데이터베이스 파티션으로 구성되어 있습니다. DPF 옵션을 이용한 다중 데이터베이스 파티션을 구성하면, 병렬 데이터베이스를 구성할 수 있습니다.

1. 서버의 호스트 명을 확인합니다. 호스트 명은 hostname 명령어의 실행 결과로 확인하며, /etc/hosts 파일의 정의 내용과 반드시 동일해야 합니다.

```
$ login <인스턴스 사용자명>
$ hostname
$ cat /etc/hosts
```

2. DPF는 rsh을 이용하므로 .rhosts 파일이 적절하게 생성되어야 합니다.

```
$ vi  rhosts
```

Tip

db2nodes.cfg 파일은 인스턴스가 중지된 상태에서만 수정이 가능합니다.

3. 인스턴스를 중지한 후에 db2nodes.cfg 파일을 수정합니다.

```
$ db2stop force
$ cd <인스턴스 사용자의 홈디렉토리>/sqllib
$ vi db2nodes.cfg
```

4. 인스턴스를 재 기동하면, 각 파티션별로 인스턴스용 프로세스가 생성됩니다. 프로세스 이름 뒤에 표시된 번호는 데이터베이스 파티션 번호입니다.

```
$ db2start
$ ps –ef | grep <인스턴스명> | grep db2sysc
```

Tip

기본적으로 한 개의 물리적인 서버에 4개의 논리적인 데이터베이스 파티션을 지원합니다. 특정 서버의 논리적 파티션의 최대 개수가 4개를 초과하면 /etc/services 파일에 정의된 TCP/IP 포트의 설정을 변경해야 합니다.

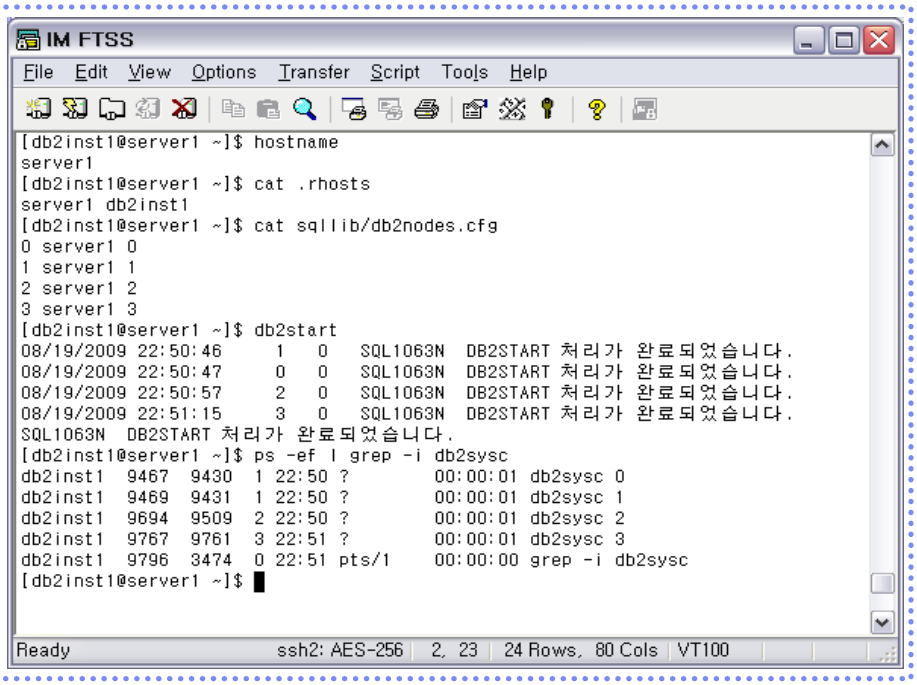


Figure 0413A 다중 데이터베이스 파티션 구성

UNIT 05

데이터베이스

하나의 인스턴스에는 독립적인 환경을 가지는 한 개 이상의 데이터베이스를 생성할 수 있습니다. 데이터베이스는 시스템 카탈로그와 로그 파일을 가지며, 테이블을 비롯한 여러 오브젝트들을 생성하여 사용하게 됩니다. CREATE DB 명령어로 생성하고, ACTIVATE DB 명령어로 기동시킵니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 데이터베이스
- 데이터베이스 생성과 제거
- 시스템 카탈로그
- 데이터베이스 구성 파일
- 데이터베이스 기동과 중지
- 데이터베이스 접속과 해제
- 데이터베이스에 접속된 응용프로그램 목록
- 응용프로그램 강제 종료
- 원격 노드 등록
- 지역 노드 등록
- 원격 데이터베이스 등록
- 시스템 데이터베이스 목록
- 지역 데이터베이스 목록

TOPIC 05 01 데이터베이스

Point

한 인스턴스에서 한 개 이상의 데이터베이스를 생성할 수 있습니다. 각 데이터베이스별로 데이터베이스 구성 파일, 기본 테이블스페이스, 시스템 카탈로그 테이블, 데이터베이스 로그 파일 등이 생성됩니다.

Tip

제품을 설치하면 SAMPLE이라는 이름의 기본 데이터베이스가 제공됩니다. db2sampl 이라는 명령어로 생성할 수 있습니다.

1 한 인스턴스에서 한 개 이상의 데이터베이스를 생성할 수 있습니다.

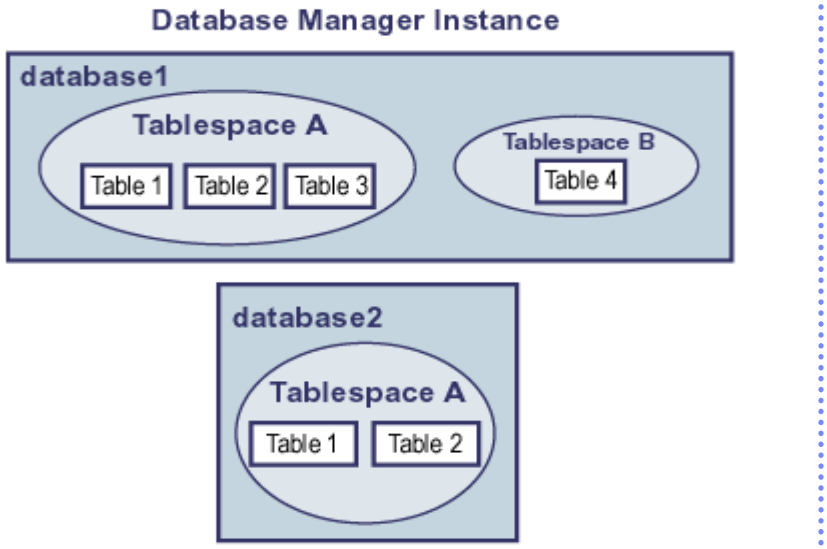


Figure 0501A ··· 동일한 인스턴스에 생성된 두 개의 데이터베이스

2 데이터베이스 구성 파일을 이용하여 개별적인 환경을 구성할 수 있습니다. 데이터베이스 구성 파일에는 다양한 데이터베이스 구성 변수가 저장됩니다.

3 3개의 기본 테이블스페이스인 SYSCATSPACE, TEMPSPACE1, USERSPACE1을 가지고 있으며, 사용자가 다양한 유형의 테이블스페이스를 추가할 수 있습니다.

4 SYSCATSPACE 테이블스페이스에는 데이터베이스의 모든 오브젝트에 대한 정보를 저장하고 있는 메타 테이블인 시스템 카탈로그 테이블이 생성됩니다.

5 데이터베이스의 데이터에 대한 변경 사항은 데이터베이스 트랜잭션 로그 파일에 기록됩니다. 초기에는 3개의 기본 로그 파일과 2개의 보조 로그 파일이 생성되며, 데이터베이스 구성 파일을 이용하여 로그 파일의 개수와 크기를 조절할 수 있습니다.

6 테이블스페이스, 테이블, 인덱스 등의 다양한 오브젝트를 생성하고, 사용자의 데이터를 테이블에 저장하여 SQL문으로 액세스합니다.

Tip

이종DBMS(오라클,MS-SQL)의 테이블과 JOIN 하려면 IFS라는 제품을 추가적으로 설치해야 합니다.

7 서로 다른 데이터베이스에 속한 테이블들은 한 개의 SQL문으로 JOIN 하려면 FEDERATED 기능을 이용합니다. DB2 제품군에 대한 FEDERATED 기능은 기본적으로 제공됩니다.

TOPIC 05 02 데이터베이스 생성과 제거

Point

인스턴스 사용자가 CREATE DB 명령어로 데이터베이스를 생성합니다. ON 옵션으로 데이터베이스와 관련된 파일들이 저장되는 디렉토리를 지정하며, CODESET 옵션으로 코드 페이지를 결정합니다. DROP DB 명령어로 제거합니다.

1 create db 명령어를 이용하여 데이터베이스를 생성합니다.

```
$ login <인스턴스 사용자명>
$ db2 CREATE DB <데이터베이스명>
$ ls –lia <인스턴스 사용자의 홈디렉토리>/$DB2INSTANCE/NODE*
```

Tip

Windows 에서는 ON 옵션에서 디렉토리명 대신에 드라이브명을 지정합니다.

2 기본 디렉토리가 아닌 다른 디렉토리에 데이터베이스 파일을 생성하려면 create db 명령어에서 ON 옵션을 사용합니다.

```
$ db2 CREATE DB <데이터베이스명> ON <디렉토리명>
$ ls –lia <디렉토리명>/$DB2INSTANCE/NODE*
```

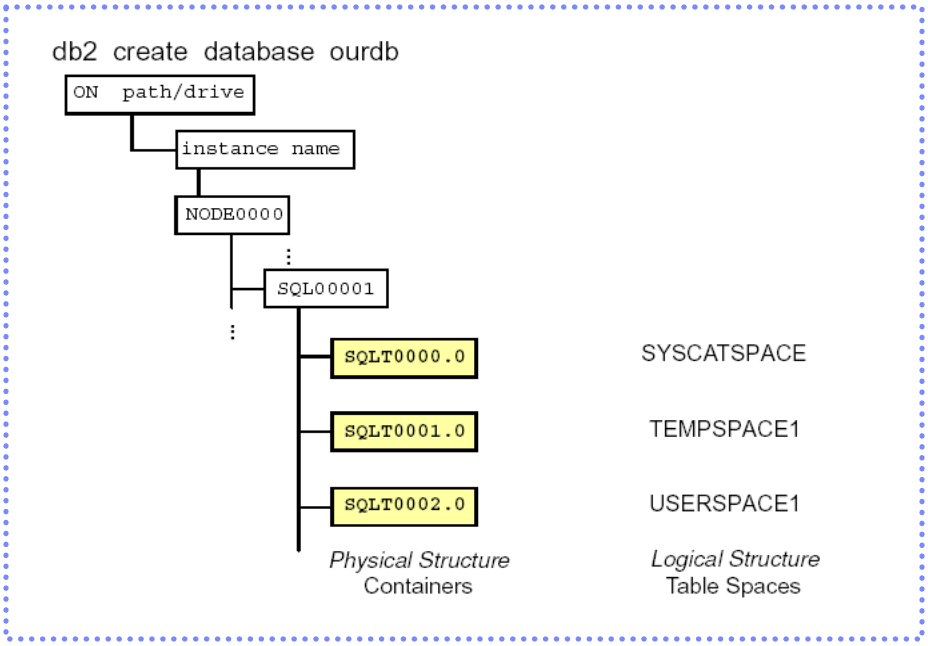


Figure 0502A • • • 데이터베이스와 관련된 서브디렉토리

Tip

데이터베이스와 세션의 코드 페이지가 호환되지 않으면, SQL0332N 오류가 반환되어 데이터베이스에 접속할 수 없습니다. 세션의 코드 페이지를 변경하거나, DB2 레지스터리 변수인 DB2CODEPAGE 를 이용합니다.

3 데이터베이스는 생성시에 사용하는 코드 페이지가 결정되고 변경할 수 없습니다. CODESET과 TERRITORY 옵션을 이용하여 데이터베이스를 위한 코드페이지를 지정할 수 있습니다. 기본값은 현재 세션에 적용된 코드 페이지로 결정됩니다.

```
$ db2 CREATE DB <데이터베이스명> USING CODESET <코드셋>
   TERRITORY <국가코드>
$ db2 GET DB CFG FOR <데이터베이스명> | grep <국가코드>
```

4 데이터베이스가 중지된 상태에서 drop db 명령어를 이용하여 제거합니다.

```
$ db2 drop db <데이터베이스명>
```

TOPIC 05 03 시스템 카탈로그

Point

SYSCATSPACE 테이블스페이스에 기본적인 시스템 카탈로그 테이블이 생성되어 데이터베이스에 대한 메타데이터를 저장합니다. 카탈로그 테이블은 SYSIBM, SYSIBMADM, SYSPUBLIC, SYSCAT, SYSSTAT 라는 스키마명을 가집니다.

Tip

카탈로그에 저장된 데이터베이스 오브젝트인 테이블, 인덱스 등의 이름은 대문자로 저장됩니다.

1. <DB2 사용자>로 로그인합니다.

```
$ login <DB2 사용자명>
```

2. 데이터베이스에 접속합니다.

```
$ db2 CONNECT TO <데이터베이스명>
```

3. 시스템 카탈로그 테이블의 목록을 확인합니다. 카탈로그는 SYSIBM, SYSCAT, SYSSTAT 라는 스키마명을 가집니다.

```
$ db2 LIST TABLES FOR SYSTEM
```

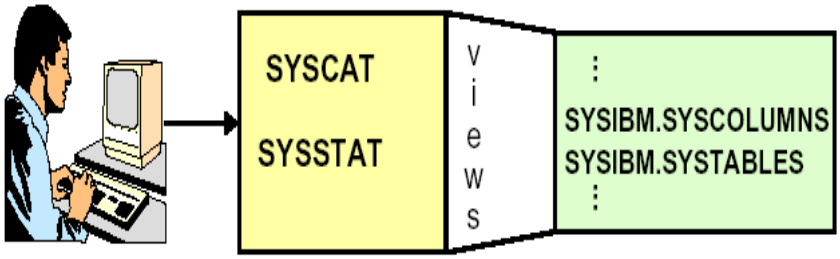


Figure 0503A ••• 시스템 카탈로그

Tip

카탈로그에 저장된 데이터베이스 오브젝트인 테이블, 인덱스 등의 이름은 대문자로 저장됩니다.

4. 시스템 카탈로그 테이블의 정보를 확인합니다.

```
$ db2 “select * from <시스템 카탈로그 테이블>”
```

```
$ db2 “select * from sysibm.systables where name = 'EMPLOYEE' ”
$ db2 “SELECT * FROM SYSIBM.SYSTABLES WHERE NAME = 'ORG' ”
```

Figure 0503B ••• 시스템 카탈로그 테이블의 정보 조회

Tip

시스템 카탈로그 테이블보다 시스템 카탈로그 뷰를 액세스하는 것이 보다 편리합니다.

Tip

시스템 카탈로그 테이블과 뷰는 기본적으로 조회만 가능하고, SYSSTAT 스키마를 가진 시스템 카탈로그 뷰는 UPDATE도 허용됩니다. (SQL튜닝 목적)

5. 시스템 카탈로그 뷰의 정보를 확인합니다.

```
$ db2 “select * from <시스템 카탈로그 뷰>”
```

```
$ db2 “select * from syscat.tables where tanname = 'EMPLOYEE' ”
$ db2 “SELECT * FROM SYSCAT.TABLES WHERE TABNAME = 'ORG' ”
```

Figure 0503C ••• 시스템 카탈로그 뷰의 정보 조회

TOPIC 05 04 데이터베이스 구성 파일

Point

데이터베이스를 생성하면 데이터베이스 구성 파일이 생성됩니다. 다양한 데이터베이스 구성 변수를 이용하여 고유한 환경을 구성할 수 있으며, 인스턴스 사용자가 GET DB CFG, UPDATE DB CFG, RESET DB CFG 명령어를 이용하여 관리합니다.

Tip

get db cfg 명령어의 show detail 옵션을 사용하려면 connect 명령어를 이용하여 데이터베이스에 접속해야 합니다.

1 <인스턴스 사용자>는 데이터베이스에 접속하여 get db cfg 명령어로 데이터베이스 구성 변수의 설정값을 확인합니다.

```
$ login <인스턴스 사용자명>
$ db2 connect to <데이터베이스명>
$ db2 get db cfg for <데이터베이스명> show detail
$ db2 connect reset
```

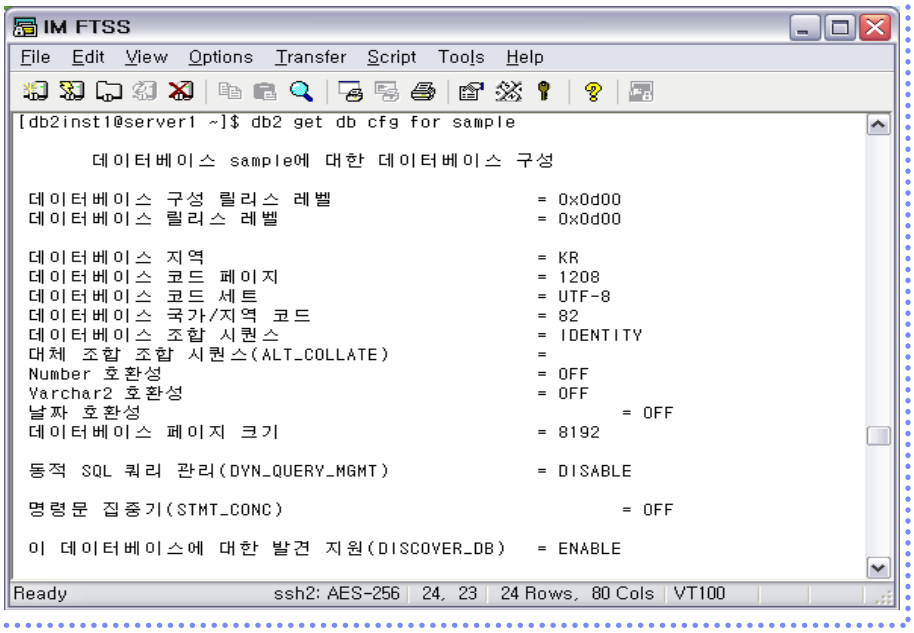


Figure 0504A • • • GET DB CFG 명령어

Tip

값을 변경하는 즉시 반영되는 데이터베이스 구성 변수를 '온라인 구성 가능한 데이터베이스 구성 변수' 라고 합니다.

2 update db cfg 명령어로 데이터베이스 구성 변수를 변경합니다.

```
$ db2 connect to <데이터베이스명>
$ db2 update db cfg for <데이터베이스명> using <구성변수명> <값>
$ db2 connect reset
```

3 reset db cfg 명령어는 데이터베이스의 모든 구성 변수를 초기화 상태로 변경합니다. 일부 데이터베이스 구성 변수는 해당 데이터베이스에 접속된 모든 응용프로그램을 종료한 후에 데이터베이스를 재기동해야 반영됩니다.

```
$ db2 reset db cfg for <데이터베이스명>
$ db2 force applications all
$ db2stop force
$ db2start
$ db2 connect to <데이터베이스명>
$ db2 get db cfg for <데이터베이스명> show detail
```

TOPIC 05-05 데이터베이스 기동과 중지

Point

ACTIVATE DB 명령어를 이용하여 데이터베이스를 기동하면, 데이터베이스와 관련된 프로세스들이 생성되고, 버퍼풀을 포함한 공유 메모리가 할당됩니다. DEACTIVATE DB 명령어로 데이터베이스를 중지한 후에는 데이터베이스를 액세스할 수 없습니다.

Tip

데이터베이스의 활성화는 인스턴스가 시작된 상태에서 가능합니다.

Tip

활성화된 데이터베이스를 deactivate db 명령어로 비활성화하고, activate db 명령어로 다시 활성화시키는 것을 데이터베이스의 재활성화라고 합니다.

Tip

데이터베이스가 활성화되지 않은 상태에서 activate db 명령어 대신에 connect to 명령어를 실행하면 자동으로 데이터베이스가 활성화되고 첫 번째 접속이 완료됩니다.

Tip

activate database 명령어를 실행하지 않은 상태로 데이터베이스가 기동된 경우에는 마지막 응용프로그램이 접속을 해제하는 순간 자동으로 데이터베이스가 중지됩니다.

1 <인스턴스 사용자>로 로그인하여 데이터베이스를 활성화시킵니다.

```
$ login <인스턴스 사용자명>
$ db2 activate db <데이터베이스명>
```

2 연관된 프로세스의 목록과 메모리를 확인합니다.

```
$ ps –ef | grep <인스턴스명>
$ db2mtrk -d
메모리 추적 설정: 21:53:53에서 2009/07/29
데이터베이스용 메모리: SAMPLE

  utilh      pckcacheh   other      catcacheh   bph (1)     bph (S32K)
  64.0K      192.0K      128.0K     64.0K       8.2M        832.0K

  bph (S16K)  bph (S8K)   bph (S4K)   shsorth     lockh       dbh
  576.0K      448.0K      384.0K      0           16.7M       20.9M
```

3 데이터베이스를 기동하기 위해 activate database 명령어를 사용했다면, deactivate database 명령어를 사용하여 중지시킵니다.

```
$ db2 deactivate db <데이터베이스명>
$ db2mtrk -d
활성화된 데이터베이스가 없음
```

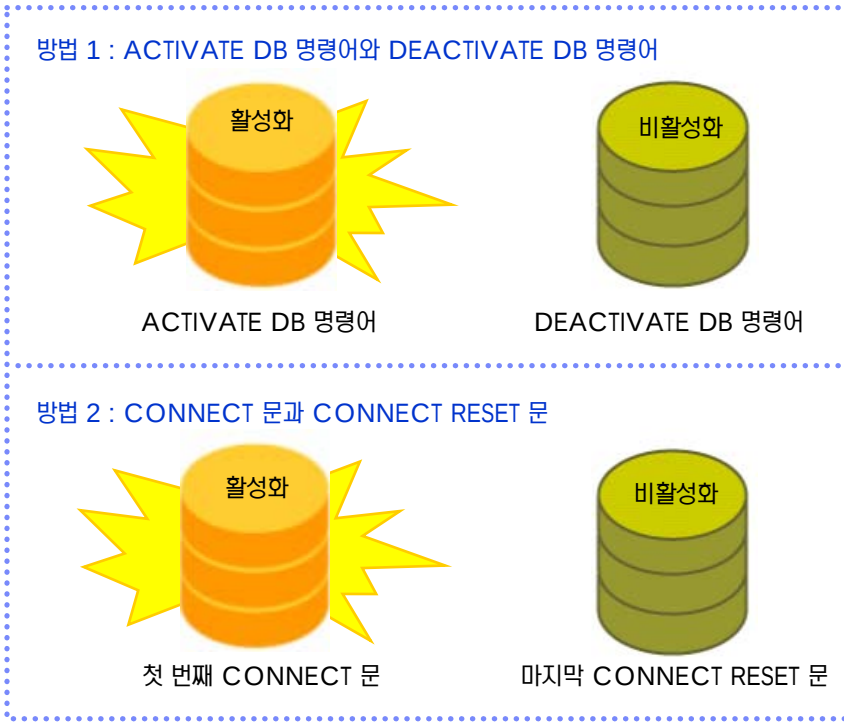


Figure 0505A... 데이터베이스의 활성화와 비활성화

TOPIC 05 06 데이터베이스 접속과 해제

Point

DB2 명령어와 SQL문을 이용하여 데이터베이스에 대한 액세스를 시작하려면 활성화된 데이터베이스에 대한 접속이 필요합니다. CONNECT 명령어를 이용하여 데이터베이스에 접속하고, CONNECT RESET 명령어를 이용하여 접속을 해제합니다.

Tip

activate database 를 사용하여 데이터베이스를 활성화한 후에 connect 명령어로 데이터베이스에 접속하는 것을 권장합니다.

Tip

데이터베이스에 접속할 때는 <데이터베이스 별명>을 이용합니다.

1 connect 문을 이용하여 데이터베이스에 접속한 후에 액세스가 가능합니다.

```
$ login <DB2 사용자명>
$ db2 connect to <데이터베이스명>
$ db2 “select * from syscat.tables”
```

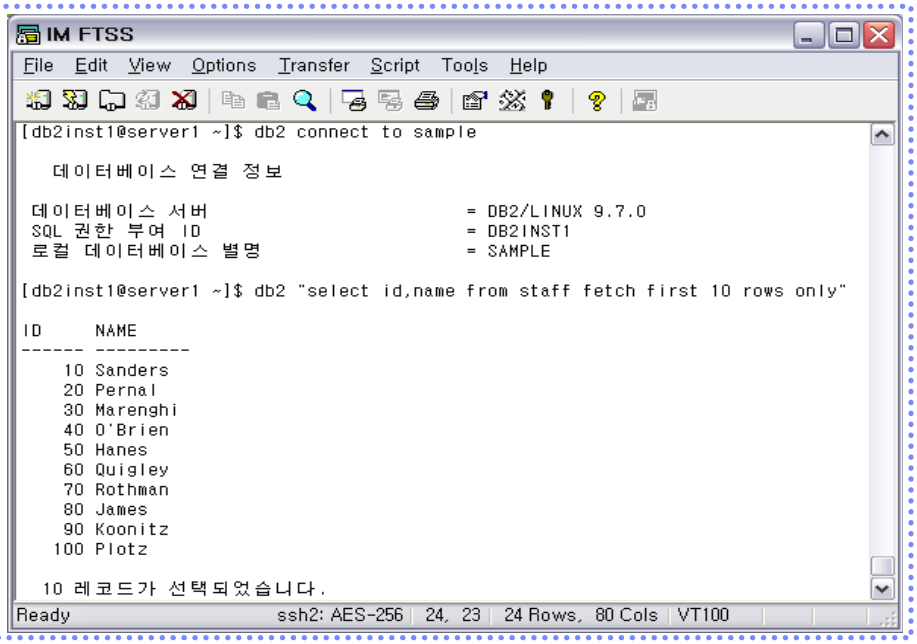


Figure 0506A • • • CONNECT 명령어

Tip

terminate 명령어를 사용하면 현재의 데이터베이스 접속을 해제하고, CLP사용자의 DB2 세션을 완전히 종료합니다.

2 데이터베이스에 대한 접속을 해제하려면, connect reset 문을 이용합니다.

```
$ db2 connect reset
```

3 원격 데이터베이스에 접속하려면, 반드시 사용자명과 암호를 지정해야 합니다.

```
$ db2 connect to <데이터베이스명> user <사용자명> using <암호>
$ db2 “select * from syscat.tables”
$ db2 connect reset
```

Tip

DFT 옵션을 사용하는 경우에는 DB2DBDFT 레지스터리 변수를 설정합니다.

4 DB2DBDFT 라는 레지스터리 변수를 이용하면, connect 문을 실행하지 않고 SQL문을 요청했을 때, 자동으로 데이터베이스에 접속됩니다.

```
$ db2set DB2DBDFT=<데이터베이스명>
$ db2 terminate
$ db2 “select * from syscat.tables”
```

Tip

DB2DBDFT 레지스터리 변수는 현재의 DB2 세션에는 적용되지 않으므로 terminate 가 필요합니다.

TOPIC 05 07 데이터베이스 접속된 응용프로그램 목록

Point

데이터베이스에 접속을 요청하면 중계자 프로세스인 db2agent 가 생성되어 활동에 필요한 개별 메모리를 할당받고, 접속된 응용프로그램은 고유한 핸들 번호로 엔진에 의해 관리됩니다. LIST APPLICATIONS 명령어로 확인합니다.

Tip

db2 명령어를 실행하면 DB2 세션이 시작되어 db2bp 프로세스가 생성되고, connect 명령을 실행하면 db2agent 프로세스가 생성됩니다.

Tip

Show detail로 내용에서 코디네이터 pid 항목은 데이터베이스에 접속을 요청한 응용프로그램의 pid가 아니라, db2agent 프로세스의 pid 입니다.

Tip

db2agent 프로세스를 위한 개별 메모리는 초기에는 최소량으로 할당되고, SQL문이 처리되는 동안 필요한 메모리가 추가적으로 할당됩니다.

1 <DB2 사용자>로 로그인하여 데이터베이스에 접속합니다.

```
$ login <DB2 사용자명>
$ db2 connect to <데이터베이스명>
$ db2 "select * from syscat.tables"
```

2 접속된 응용프로그램의 정보를 확인합니다. show detail 옵션이 제공됩니다.

```
$ db2 list applications
$ db2 list applications show detail
```

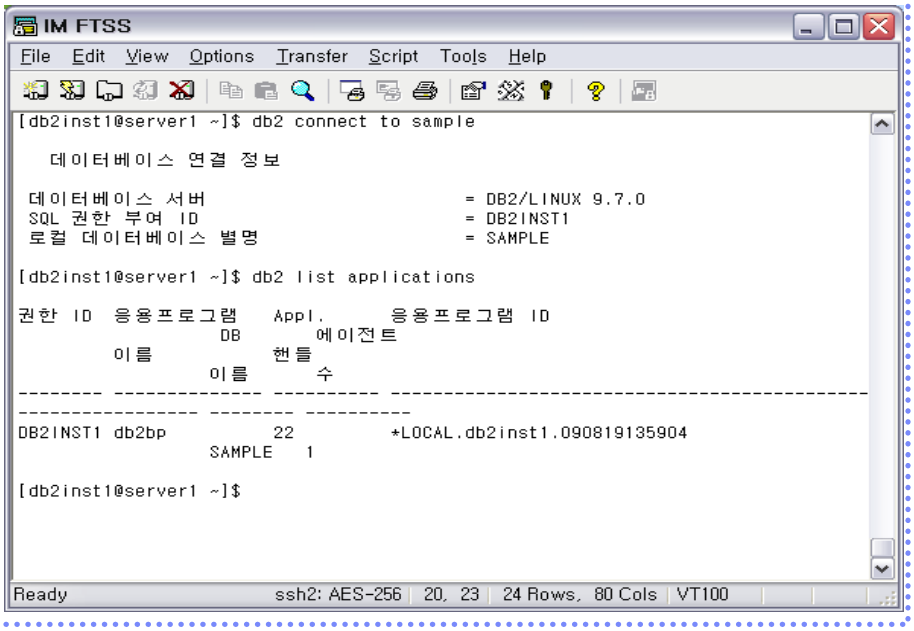


Figure 0507A • • • LIST APPLICATIONS 명령어

3 응용프로그램이 데이터베이스에 접속을 요청하면 db2agent 가 생성됩니다. db2agent 프로세스는 개별 메모리를 할당받습니다.

```
$ ps –ef | grep <인스턴스명> | grep db2agent
$ db2mtrk -p
에이전트 5284006용 메모리
  other    apph    appctlh
  16.0K    128.0K  16.0K
```

4 데이터베이스에 대한 접속을 해제하면, db2agent는 제거 되거나 idle 상태로 바뀌게 됩니다.

```
$ db2 connect reset
$ db2pd -d sample -edu | grep -i db2agent
```

TOPIC 05 08 응용프로그램 강제 종료

Point

데이터베이스에 접속했던 응용프로그램은 CONNECT RESET 명령어로 접속을 종료해야 합니다. FORCE APPLICATION 명령어는 응용프로그램의 핸들 번호를 이용하여 응용프로그램을 강제 종료합니다. 강제로 종료된 응용프로그램은 롤백됩니다.

Tip

force application 명령어는 비동기 모드로 처리되므로, 응용프로그램이 즉시 종료되지 않을 수 있습니다.

Tip

강제로 종료된 응용프로그램은 마지막으로 실행 중이던 트랜잭션의 롤백을 완료해야 하므로 종료되는데 상당한 시간이 걸릴 수도 있습니다.

Tip

SQL문이 실행 중인 경우에는, force application 명령을 실행하고, list application 명령어로 결과를 확인하도록 합니다. 긴 트랜잭션이 있었다면, Rollback으로 인해 정지하는데 시간이 소요될 수 있습니다.

1 <인스턴스 사용자>로 로그인하여 데이터베이스에 접속합니다.

```
$ login <DB2 사용자명>
$ db2 connect to <데이터베이스명>
$ db2 “select * from syscat.tables”
```

2 접속된 응용프로그램의 이름과 그 핸들 번호를 확인합니다. 핸들 번호는 응용프로그램에게 할당되는 OS의 프로세스 id가 아니라, DB2 엔진이 부여한 고유 번호입니다.

```
$ db2 list applications
```

3 force applications 명령어에서 핸들 번호를 이용하여 특정한 응용프로그램을 강제로 종료할 수 있습니다. 여러 개의 응용프로그램을 강제 종료하려면 ,(컴마) 를 이용하여 핸들 번호를 나열합니다.

```
$ db2 “force application (<핸들 번호>)”
$ db2 “force application (<핸들 번호 1>, <핸들 번호 2>)”
$ db2 list applications
```

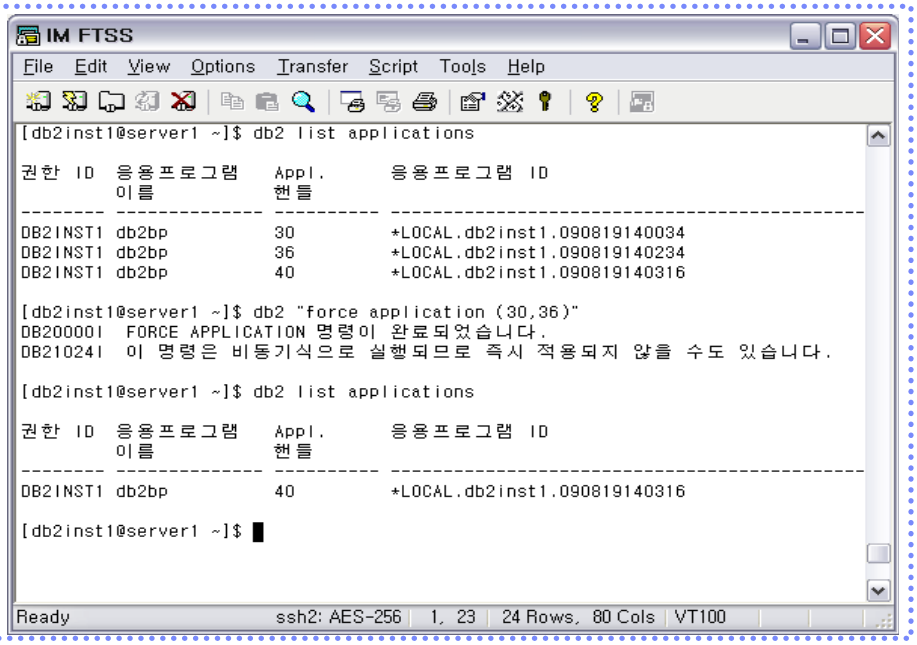


Figure 0508A • • • FORCE APPLICATION 명령어

4 접속된 모든 응용프로그램을 강제로 종료하려면 all 옵션을 사용합니다.

```
$ db2 force applications all
$ db2 list applications
```

TOPIC 05 09 원격 노드 등록

Point

원격 서버의 인스턴스에 존재하는 데이터베이스를 액세스하려면 <원격 서버의 IP 주소>와 <원격 서버 인스턴스의 TCP/IP 서비스 포트 번호>를 이용하여 원격 노드를 등록해야 합니다. CATALOG TCPIP NODE 명령어를 이용합니다.

Tip

Windows인 경우에는 C:\WINDOWS\system32\drivers\etc 디렉토리의 hosts 파일과 services 파일을 이용합니다.

1 서버에서 인스턴스 사용자로 로그인하여 <IP 주소>를 확인합니다. 이 값이 <원격 서버의 IP 주소> 입니다.

```
$ login <서버의 인스턴스 사용자명>
$ cat /etc/hosts
```

2 서버의 인스턴스 구성 변수인 SVCENAME 의 값이 서비스명이면/etc/services 파일을 확인합니다. 이 값이 < 원격 인스턴스의 TCP/IP 서비스 포트 번호>입니다.

```
$ db2 get dbm cfg | grep SVCENAME
$ cat /etc/services
```

3 클라이언트에서 인스턴스 사용자로 로그인하여 catalog tcpip node 명령어로 원격 서버의 인스턴스를 등록합니다. <노드명>은 임의로 정할 수 있으며, 서버에서 확인한 <원격 서버의 IP 주소>와 <원격 인스턴스의 TCP/IP 서비스 포트 번호>를 이용합니다.

Tip

원격 서버를 나타내는 <원격서버의 IP 주소> 대신에 클라이언트의 /etc/hosts 파일을 이용한 <호스트명>을 사용할 수 있습니다.

```
$ login <클라이언트의 인스턴스 사용자명>
$  db2 catalog tcpip node <노드명> remote <원격 서버의 IP 주소>  server <원격 인스턴스의 TCP/IP 서비스 포트 번호>
$ db2 list node directory
```

Tip

원격 인스턴스의 <원격 인스턴스의 TCP/IP 서비스 포트 번호> 대신에 클라이언트의 /etc/services 파일을 이용한 <서비스명>을 사용할 수 있습니다.

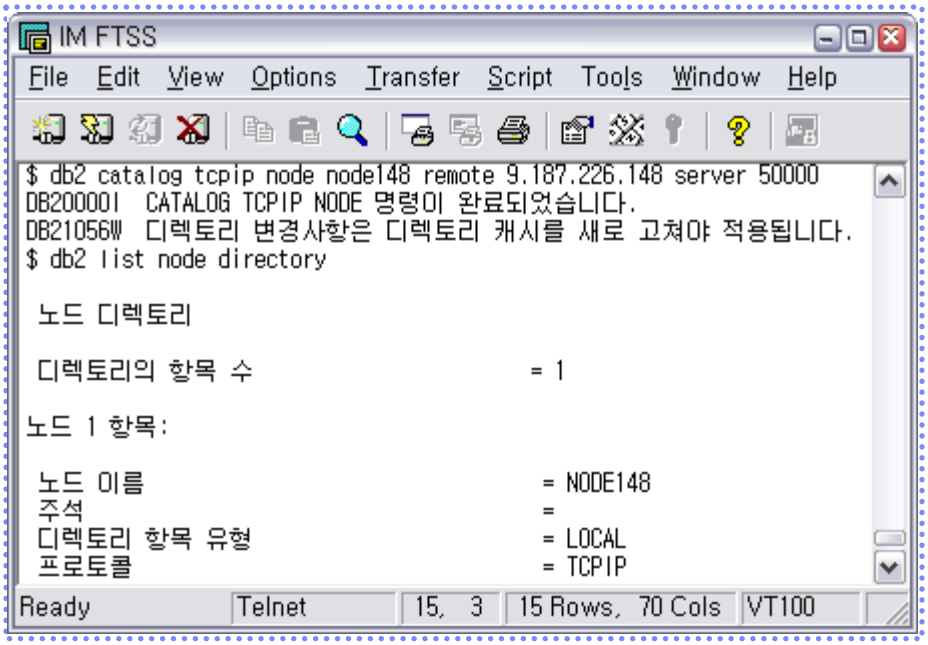


Figure 0509A ••• CATALOG TCPIP NODE 명령어

4 등록된 원격 노드는 uncatalog node 명령어로 제거합니다.

```
$ db2 uncatalog node <노드명>
```

TOPIC 05-10 지역 노드 등록

Point

동일한 서버에 존재하는 다른 인스턴스를 지역 노드라고 합니다. CATALOG LOCAL NODE 명령어를 이용하여 지역 노드를 등록하면, 등록된 지역 노드에 존재하는 데이터베이스를 액세스할 수 있습니다.

Tip

인스턴스명만 지정하면, TCP/IP 통신에 필요한 IP주소와 서비스 포트 번호를 내부적으로 인식합니다.

1 인스턴스 사용자로 로그인합니다.

```
$ login <인스턴스 사용자명>
```

2 catalog local node 명령어로 다른 인스턴스를 등록합니다. <노드명>은 임의로 정할 수 있습니다. 인스턴스명을 이용합니다.

```
$ db2 catalog local node <노드명> instance <다른 인스턴스명>
```

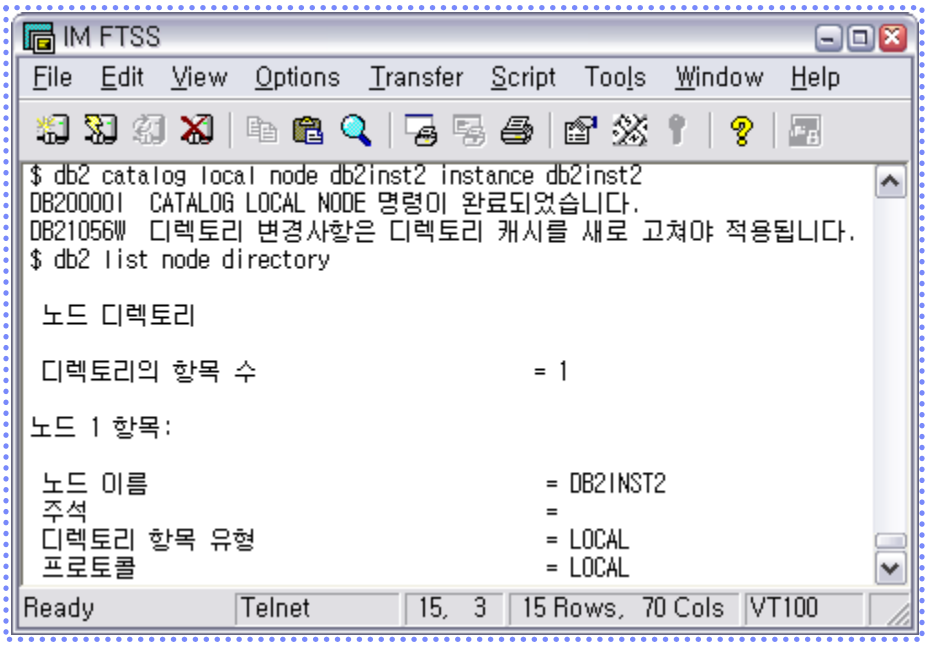


Figure 0510A CATALOG LOCAL NODE 명령어

3 catalog tcpip node 명령어로 원격 서버의 인스턴스와 동일한 방법으로 등록할 수도 있습니다. <노드명>은 임의로 정할 수 있으며, <현재 서버의 IP 주소>와 <지역 인스턴스의 TCP/IP 서비스 포트 번호>를 이용합니다.

```
$ login <인스턴스 사용자명>
$ db2 catalog tcpip node <노드명> remote <현재 서버의 IP 주소> server <지역 인스턴스의 TCP/IP 서비스 포트 번호>
```

4 list node directory 명령어로 등록한 지역 노드의 목록을 확인합니다.

```
$ db2 list node directory
```

5 등록한 지역 노드는 uncatalog node 명령어로 제거합니다.

```
$ db2 uncatalog node <노드명>
```

TOPIC 05-11 원격 데이터베이스 등록

Point

원격 노드 또는 지역 노드에 존재하는 원격 데이터베이스는 CATALOG DB 명령어를 이용하여 원하는 데이터베이스 별명으로 등록하여 액세스합니다.

1 서버에서 인스턴스 사용자로 로그인하여 <원격 데이터베이스의 별명>을 확인합니다.

```
$ login <서버의 인스턴스 사용자명>
$ db2 list db directory
```

2 클라이언트에서 인스턴스 사용자로 로그인하여 list node 명령어로 원격 데이터베이스가 존재하는 <원격 노드명>을 확인합니다.

```
$ login <클라이언트의 인스턴스 사용자명>
$ db2 list node directory
```

Tip

AS 옵션을 지정하지 않으면 원격 데이터베이스의 별명과 동일한 별명으로 등록됩니다.

3 catalog db 명령어를 이용하여 원격 데이터베이스를 등록합니다. <등록할 데이터베이스 별명> 은 고유한 데이터베이스 별명으로 임의로 정합니다.

```
$ db2 catalog <원격 데이터베이스의 별명> as <등록할 데이터베이스 별
    명> at node <원격 노드명>
```

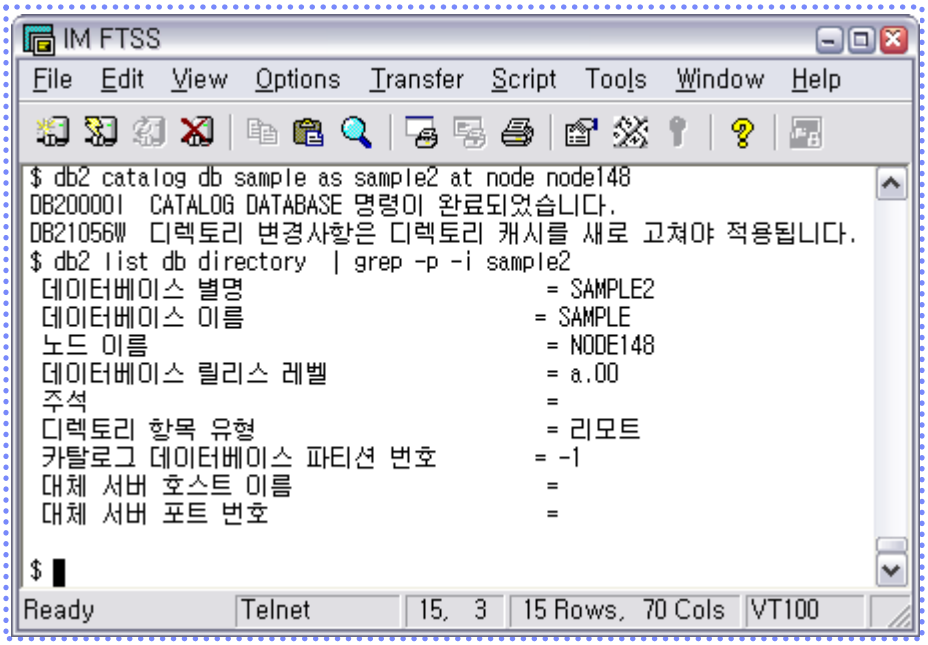


Figure 0511A ··· CATALOG DB 명령어

Tip

원격 데이터베이스에 접속할 때는 반드시 서버의 OS 계정의 <사용자명>과 <암호>를 지정해야 합니다.

4 connect 문을 이용하여 원격 데이터베이스에 접속합니다.

```
$ db2 connect to <등록 데이터베이스 별명>  user <원격 사용자명> using <원격
    사용자의 암호명>
```

Tip

db2ubind.lst 파일의 바인드 작업은 클라이언트 플랫폼 유형별로 한 번만 실행하면 됩니다.

5 원격 데이터베이스에 대해 DB2 유틸리티를 실행하려면 db2ubind.lst 파일을 바인드합니다.

```
$ cd <인스턴스 사용자의 홈디렉토리>/sqllib/bnd
$ db2  bind @db2ubind.lst blocking all grant public
```

TOPIC 05-12 시스템 데이터베이스 목록 확인

Point

현재 인스턴스에 등록된 모든 데이터베이스 목록을 시스템 데이터베이스 목록이라고 합니다. 현재 인스턴스에서 생성된 지역 데이터베이스와 등록된 원격 데이터베이스의 목록을 모두 포함합니다. LIST DB DIRECTORY 명령어를 이용합니다.

Tip

원격 데이터베이스는 '리모트' 유형으로 분류되며, catalog node 명령어로 등록된 원격 노드과 catalog db 명령어로 등록한 데이터베이스 별명이 표시됩니다.

Tip

지역 데이터베이스는 '간접' 유형으로 분류되며, create db 명령어에서 지정된 데이터베이스 생성 디렉토리 명이 표시됩니다.

Tip

create db 명령어로 데이터베이스를 생성하면 데이터베이스명과 동일한 데이터베이스 별명이 자동으로 등록됩니다.

1 <DB2 사용자>로 로그인합니다.

```
$ login <DB2 사용자명>
```

2 list db directory 명령어로 시스템 데이터베이스의 목록을 확인합니다.

```
$ db2 list db directory
```

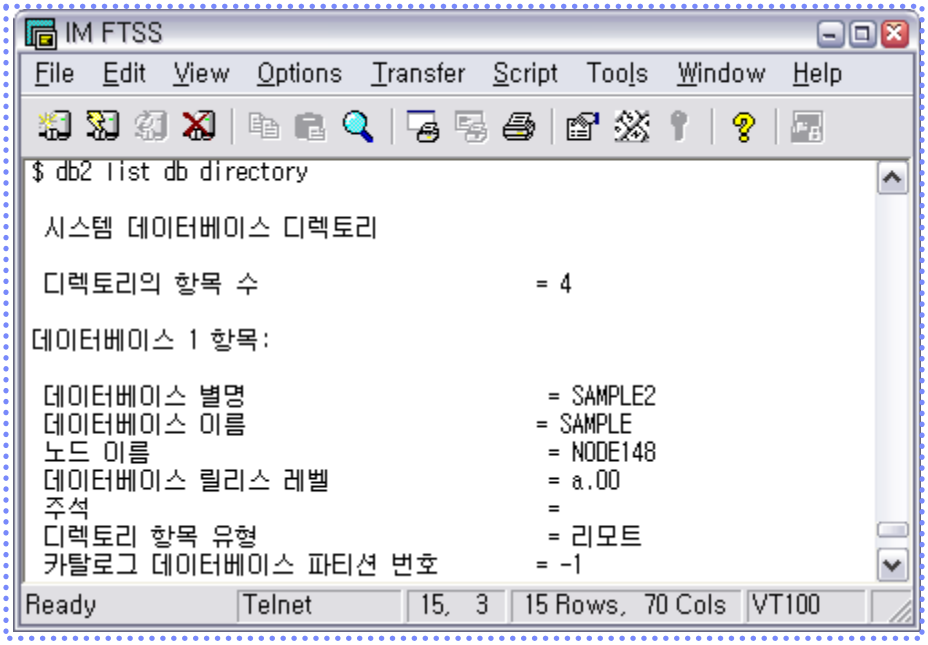


Figure 0512A • • • LIST DB DIRECTORY 명령어로 원격 데이터베이스 정보 확인

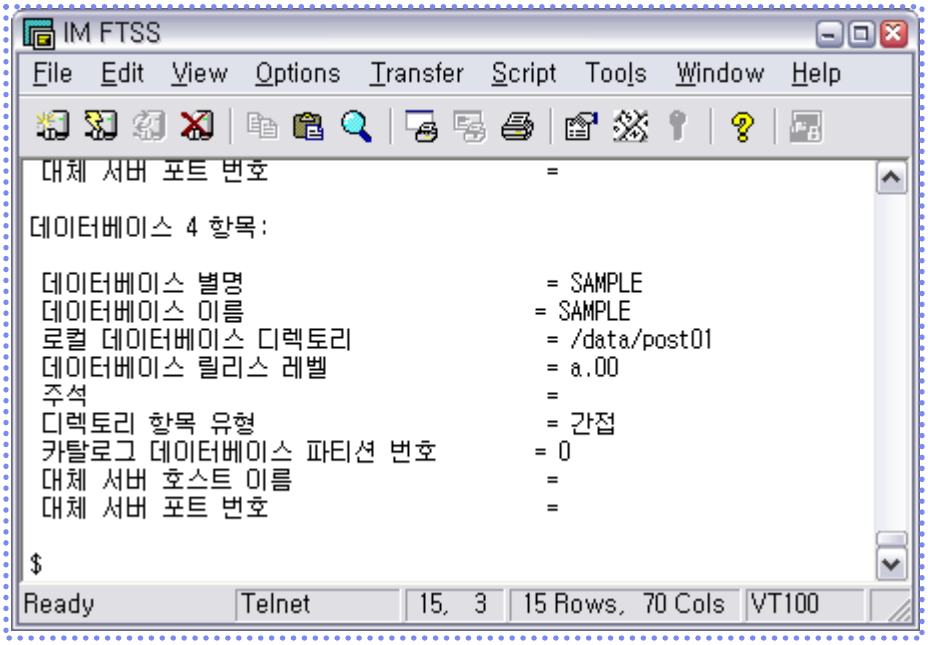


Figure 0512B • • • LIST DB DIRECTORY 명령어로 지역 데이터베이스 정보 확인

TOPIC 05-13 지역 데이터베이스 목록 확인

Point

현재 인스턴스에 실제로 존재하는 데이터베이스 중에서 특정한 디렉토리 또는 드라이브에 실제로 생성되어 있는 데이터베이스의 목록을 지역 데이터베이스 목록이라고 하며, LIST DB DIRECTORY 명령어의 ON 옵션으로 확인합니다.

Tip

'데이터베이스 디렉토리' 항목에 표시된 값은 현재 서버에서 데이터베이스와 관련된 실제 파일들이 생성된 서브 디렉토리명입니다.

1. <DB2 사용자>로 로그인합니다.

```
$ login <DB2 사용자명>
```

2. list db directory 명령어의 ON 옵션을 이용하여 특정한 <디렉토리명>에 실제로 생성되어 있는 <지역 데이터베이스>의 목록을 확인합니다.

```
$ db2 list db directory ON <디렉토리명>
```

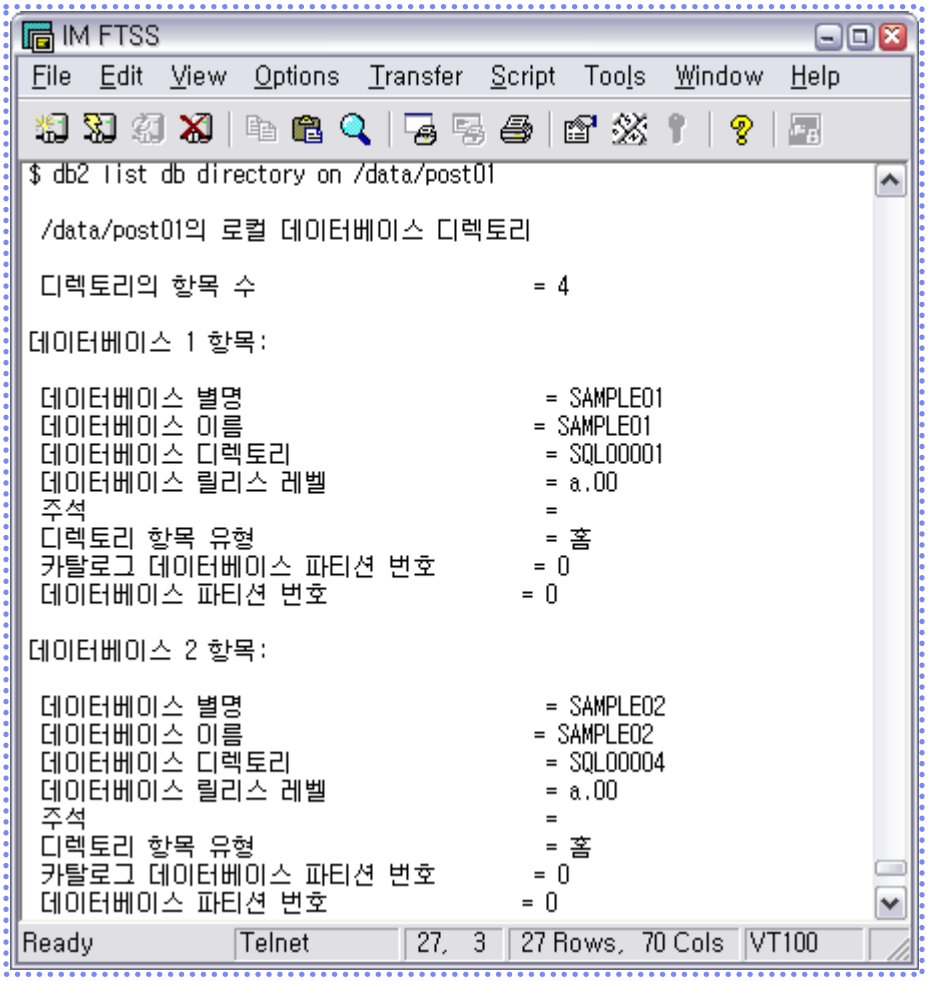


Figure 0513A ••• LIST DB DIRECTORY 명령어의 ON 옵션

3. Windows 에서는 특정한 <디렉토리명> 대신에 <드라이브명>을 이용합니다.

```
시작 → 실행 → db2cmd
C:\> db2 list db directory ON <드라이브명>
```

UNIT 06

버퍼풀

한 데이터베이스에는 한 개 이상의 버퍼 풀을 생성하여 운영할 수 있습니다. 버퍼 풀은 사용자의 액세스 요청을 처리하기 위해 데이터베이스의 데이터 페이지를 읽고 수정하기 위한 메모리 영역입니다. 기본 제공되는 버퍼 풀은 IBMDEFAULTBP이며, CREATE BUFFERPOOL 문으로 생성합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 버퍼풀 개요
- 버퍼풀 생성
- 버퍼풀 변경
- 버퍼풀 제거
- 블록 기반의 I/O

TOPIC 06 01 버퍼풀 개요

Point

한 데이터베이스에 한 개 이상의 버퍼 풀을 생성하여 운영할 수 있습니다. 버퍼 풀은 사용자의 액세스 요청을 처리하기 위해 데이터베이스 페이지를 읽고 수정하기 위한 메모리 영역입니다. 기본 버퍼 풀은 IBMDEFAULTBP입니다.

Tip

SYSADM 또는 SYSCTRL 권한의 사용자가 관리합니다.

Tip

버퍼 풀에 대한 사용 권한 또는 특권은 별도로 존재하지 않습니다. 테이블 공간 생성 시 지정하면 자동으로 사용됩니다.

1 <인스턴스 사용자>로 로그인합니다.

```
$ login <인스턴스 사용자명>
```

2 버퍼 풀의 정보는 SYSCAT.BUFFERPOOLS 뷰를 이용해서 확인합니다.

```
$ db2 connect to <데이터베이스명>
$ db2 "select * from syscat.bufferpools"
```

3 데이터베이스에 저장된 모든 데이터는 액세스 요청 시에 버퍼 풀을 통하여 검색되므로 데이터베이스에는 반드시 한 개 이상의 버퍼풀이 존재해야 합니다. 기본적으로 생성되는 버퍼 풀의 이름은 IBMDEFAULTBP 로써 1000개의 페이지가 할당됩니다.

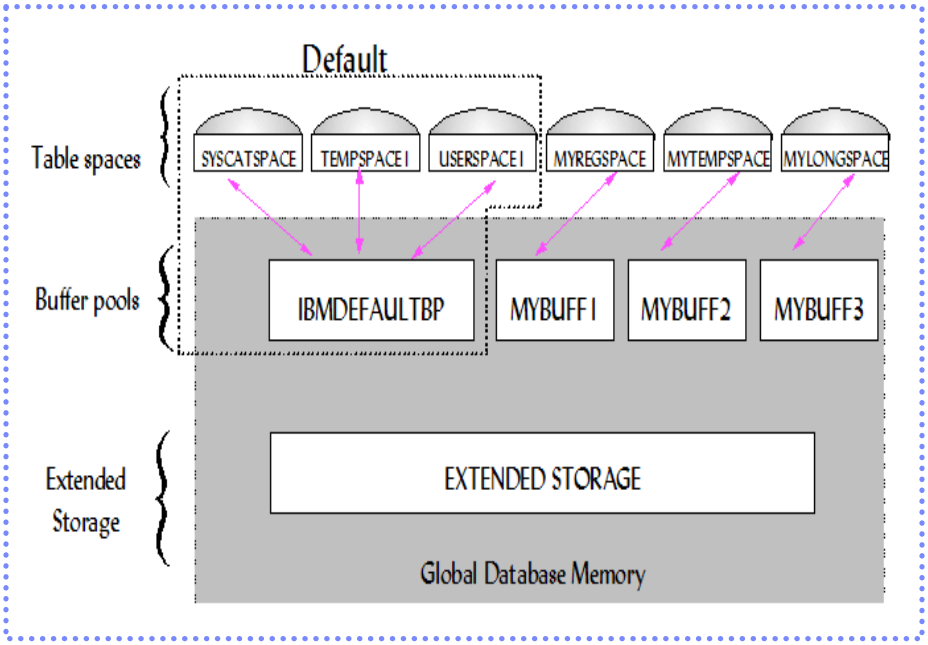


Figure 0601A ··· IBMDEFAULTBP와 사용자 정의 버퍼풀

4 데이터가 저장된 물리적인 저장 영역인 테이블스페이스 컨테이너의 페이지 크기와 대응되는 버퍼 풀의 페이지 크기는 동일해야 합니다. 여러 개의 테이블스페이스는 한 개의 버퍼 풀을 공유하여 사용할 수 있습니다.

5 DPF 환경에서는 데이터베이스 파티션 별로 버퍼 풀을 생성할 수 있습니다. 옵션을 지정하지 않으면, 모든 데이터베이스 파티션에 동일한 크기의 버퍼풀이 생성됩니다.

6 버퍼 풀의 최대 NPAGES는 64비트 플랫폼에서 2,147,483,647고, 32비트 플랫폼에서는 1,048,576 입니다.

TOPIC 06 02 버퍼풀 생성

Point

CREATE BUFFERPOOL 문을 이용하여 버퍼 풀을 생성합니다. 생성한 버퍼 풀은 즉시 할당되며, 가용 메모리가 없는 경우에는 데이터베이스가 재 활성화되었을 때 할당됩니다. 버퍼 풀을 생성할 때 페이지의 크기와 버퍼의 크기가 결정됩니다.

Tip

DB2 9.1부터 자체 성능 조정 메모리 사용 가능으로 dbm cfg의 self_tuning_mem -> on으로 변경 하고 create, alter bufferpool 시 버퍼 풀 크기를 AUTOMATIC으로 지정할 수 있습니다.

Tip

버퍼풀 명의 최대 길이는 18자입니다.

Tip

데이터베이스가 활성화된 상태에서 버퍼 풀을 생성할 때 충분한 크기의 메모리가 없으면, SQLSTATE 01657 이 반환되고, 생성된 버퍼 풀은 데이터베이스가 재 활성화될 때 할당됩니다.

Tip

데이터베이스 활성화 시에 정의된 버퍼 풀을 모두 할당할 가용 공간이 없으면 IBMDEFAULTBP 만 할당됩니다.

1. <인스턴스 사용자>로 로그인하여 데이터베이스에 접속합니다.

```
$ login <인스턴스 사용자명>
$ db2 connect to <데이터베이스명>
```

2. create bufferpool 명령어를 이용하여 버퍼 풀을 생성합니다. <페이지의 크기>는 생성시에 결정되어 변경될 수 없습니다. <페이지의 개수>는 향후에 변경할 수 있습니다. <페이지의 크기>는 별도로 지정하지 않으면 4K 가 됩니다.

```
$ db2 "create bufferpool <버퍼풀명> size <페이지의 개수>"
```

3. PAGESIZE 옵션을 이용하여 페이지의 크기를 지정합니다. 지원되는 페이지의 크기는 4K, 8K, 16K, 32K 입니다. 1024 바이트를 의미하는 K는 공백 문자 없이 붙여서 표현합니다.

```
$ db2 "create bufferpool <버퍼풀명> size <페이지의 개수> pagesize <페이지의 크기> "
```

4. 가용 공간이 있으면 생성한 버퍼 풀은 즉시 할당됩니다. 데이터베이스의 재 활성화 시에 할당하려면 DEFERRED 옵션을 사용합니다.

```
$ db2 "create bufferpool <버퍼풀명> DEFERRED size <페이지의 개수>"
```

5. 생성된 버퍼 풀에 대한 정보는 SYSCAT.BUFFERPOOLS 에서 확인합니다.

```
$ db2 "select * from syscat.bufferpools"
```

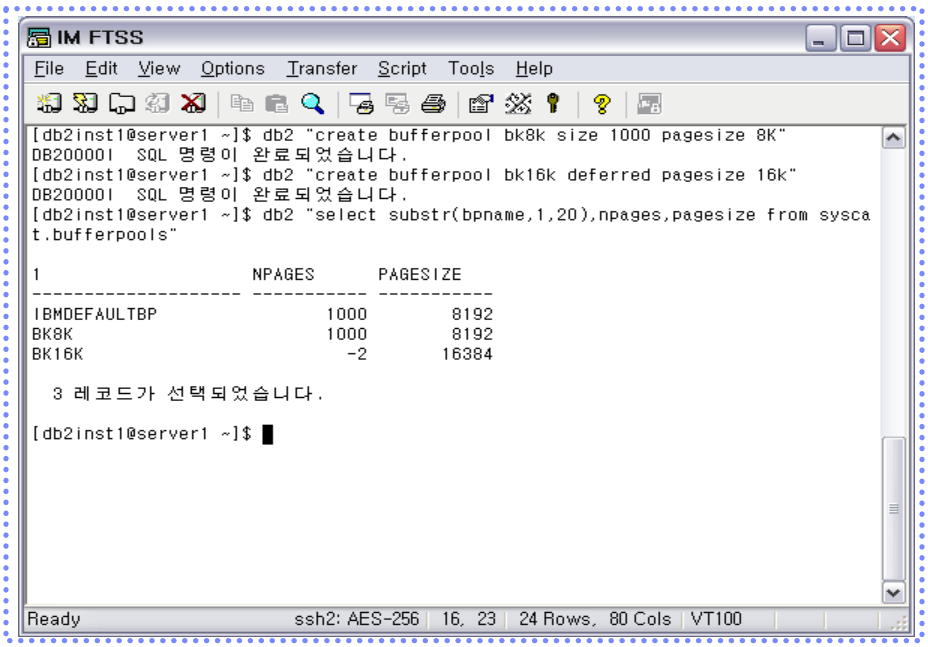


Figure 0602A • • • CREATE BUFFERPOOL 문

TOPIC 06 03 버퍼풀 변경

Point

ALTER BUFFERPOOL 문을 이용하여 버퍼풀의 특성을 변경합니다. 페이지의 크기는 변경할 수 없으며, 버퍼풀의 크기와 블록 기반 I/O 에 사용되는 버퍼 풀의 블록화 영역은 수정할 수 있습니다.

1 <인스턴스 사용자>로 로그인합니다.

```
$ login <인스턴스 사용자명>
```

2 데이터베이스에 대한 접속이 필요합니다.

```
$ db2 connect to <데이터베이스명>
```

Tip

db2pd 명령어를 이용하여 할당된 버퍼풀의 크기를 확인할 수 있습니다.

3 alter bufferppol 명령어의 SIZE 옵션을 이용하여 버퍼풀의 크기를 증가시키거나 감소시킵니다. 변경한 크기는 즉시 반영됩니다.

```
$ db2mtrk -d
$ db2 "select BPNAME, PAGESIZE, NPAGES  from syscat.bufferpools"
$ db2 "alter bufferpool <버퍼풀명> SIZE <페이지 개수>"
$ db2mtrk -d
```

Tip

버퍼풀의 블록 기반 I/O에 대한 변경 정보는 SYSCAT.BUFFERPOOLS 에는 즉시 반영됩니다.

4 alter bufferppol 명령어의 NUMBLOCKPAGES와 BLOCKSIZE 옵션을 이용하여 블록 기반의 I/O 영역의 크기를 변경합니다. 변경한 크기를 반영하려면 데이터베이스의 재활성화가 필요합니다.

```
$ db2 "alter bufferpool <버퍼풀명> NUMBLOCKPAGES <블록의 개수>"
$ db2 "alter bufferpool <버퍼풀명> BLOCKSIZE <블록의 크기>"
$ db2 deactivate db <데이터베이스명>
$ db2 activate db <데이터베이스명>
```

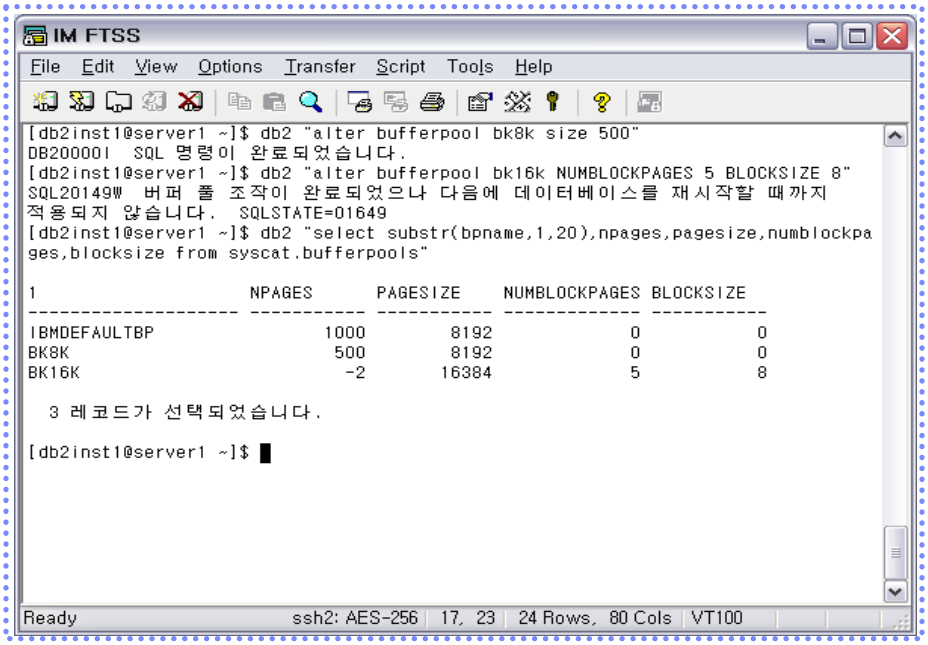


Figure 0603A ··· ALTER BUFFERPOOL 문

TOPIC 06 04 버퍼풀 제거

Point

DROP BUFFERPOOL 문을 이용하여 버퍼풀을 제거합니다. 버퍼풀을 사용하고 있는 테이블스페이스가 있으면 제거할 수 없습니다. 제거된 버퍼풀에 할당했던 메모리는 다른 용도로 사용될 수 있도록 즉시 해제됩니다.

1 <인스턴스 사용자>로 로그인하여 데이터베이스의 버퍼풀에 할당된 메모리 크기를 확인합니다.

```
$ login <인스턴스 사용자명>
$ db2mtrk -d
```

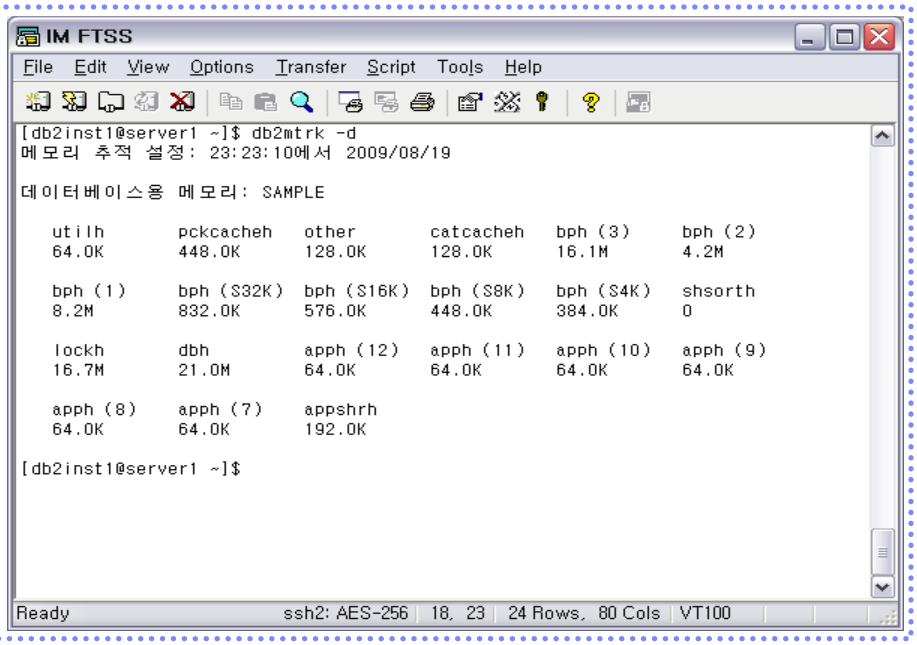


Figure 0604A • • • db2mtrk 명령어로 버퍼풀의 메모리 크기 확인

2 drop bufferppol 명령어로 버퍼풀을 제거하고, 메모리가 해제된 것을 확인합니다.

```
$ db2 "drop bufferpool <버퍼풀명>"
```

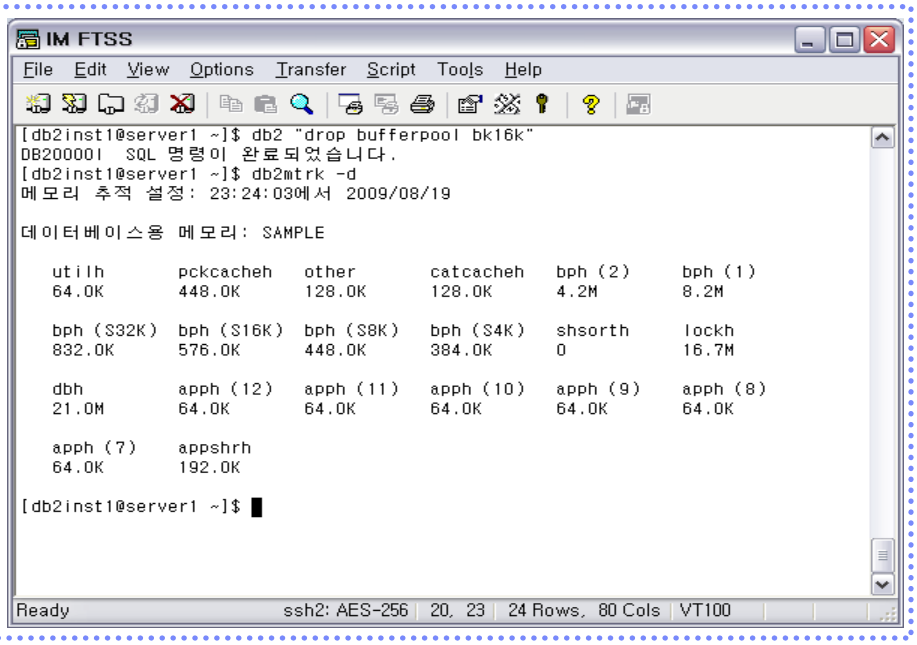


Figure 0604B • • • DROP BUFFERPOOL 문의 실행 후 메모리 해제 확인

Tip

IBMDEFAULTBP 버퍼풀은 삭제할 수 없습니다.

Tip

버퍼풀을 사용하고 있는 테이블스페이스가 있으면 SQLSTATE 42893 오류가 반환되고, 버퍼풀은 제거할 수 없습니다.

TOPIC 0605 블록 기반의 I/O

Point

디스크의 데이터는 기본적으로 한 페이지 단위로 버퍼풀로 전송됩니다. 몇 개의 페이지를 묶어서 블록으로 정의하고, 버퍼풀의 일부를 블록 단위의 I/O 영역으로 지정합니다. CREATE BUFFERPOOL 문의 BLOCKSIZE와 NUMBLOCKS 옵션을 이용합니다.

Tip

블록의 개수가 버퍼풀의 98%를 초과하면 SQLSTATE 54052가 반환됩니다. 값을 0으로 지정하면 블록 기반의 I/O를 사용할 수 없게 됩니다.

Tip

한 블록의 크기가 2 - 256 사이의 값이 아니면 SQLSTATE 54053 이 반환됩니다. 기본값은 32 페이지입니다.

1. <인스턴스 사용자>로 로그인합니다.

```
$ login <인스턴스 사용자명>
```

2. 데이터베이스에 대한 접속이 필요합니다.

```
$ db2 connect to <데이터베이스명>
```

3. create bufferpool 문의 BLOCKSIZE 와 NUMBLOCKS 옵션을 이용하면 blocked I/O 영역을 지정할 수 있습니다. <한 블록의 크기>는 페이지 단위로 지정합니다.

```
$ vi  <임의의 파일명>
create bufferpool <버퍼풀명>
size <페이지의 개수>
NUMBLOCKPAGES <블록의 개수>
BLOCKSIZE <블록의 크기>
PAGESIZE <페이지 크기>;
$ db2 –svtf <임의의 파일명>
```

4. SYSCAT.BUFFERPOOLS 에서 BLOCKSIZE, NUMBLOCKPAGES 컬럼의 값으로 블록 기반의 I/O 영역의 크기를 확인합니다.

```
$ db2 "select substr(BPNAME,1,18) BPNAME, BLOCKSIZE,
    NUMBLOCKPAGES from syscat.bufferpools"
```

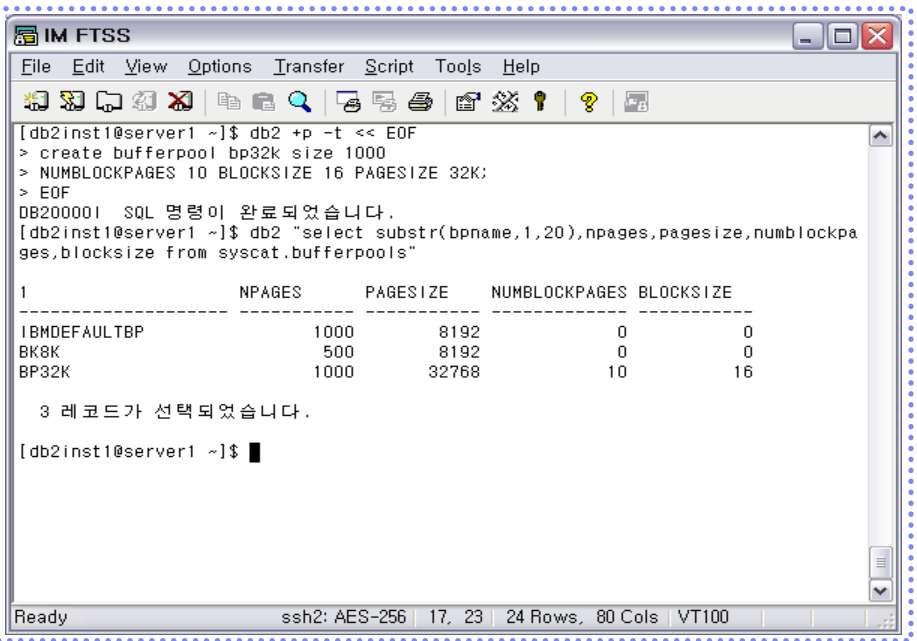


Figure 0605A ··· 블록 기반 I/O 영역의 크기 설정

UNIT 07

테이블스페이스

테이블스페이스는 한 개 이상의 컨테이너로 이루어진 논리적인 오브젝트입니다. 데이터를 테이블에 저장하는 SQL문을 실행하면, 실제적인 데이터는 연관된 테이블스페이스의 물리적인 저장 공간인 컨테이너에 라운드 로빈 방식으로 균등하게 저장됩니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 테이블스페이스
- 기본 테이블스페이스
- SMS 테이블스페이스
- DMS 테이블스페이스
- LARGE 테이블스페이스
- SYSTEM TEMPORARY 테이블스페이스
- USER TEMPORARY 테이블스페이스
- 디렉토리 컨테이너
- 파일 컨테이너
- 디바이스 컨테이너
- 페이지 (PAGE)
- 익스텐트 (EXTENT)
- 페이지 클리너 (I/O Cleaners)
- I/O 프리페치 (I/O Prefetch)
- 테이블스페이스 생성
- 테이블스페이스 특성 변경
- 테이블스페이스 컨테이너 변경
- High Water Mark 조정
- 테이블스페이스 제거
- SYSCAT.TABLESPACES 뷰

TOPIC 0701 테이블스페이스

Point

데이터베이스의 데이터는 물리적인 저장 공간인 컨테이너에 저장됩니다. 테이블스페이스는 한 개 이상의 컨테이너 묶음에 대한 논리적인 이름입니다. CREATE TABLESPACE문, ALTER TABLESPACE문, DROP TABLESAPCE 문으로 관리합니다.

Tip
테이블스페이스에 대한 정보는 SYSCAT.TABLESPACES 에 저장됩니다.

Tip
한 데이터베이스에 한 개 이상의 테이블스페이스를 생성하여 사용합니다.

Tip
한 테이블스페이스는 여러 개의 컨테이너로 구성됩니다. 한 컨테이너는 한 개의 테이블스페이스에만 속합니다.

Tip
한 테이블의 데이터는 기본적으로 한 개의 테이블스페이스에 저장됩니다.

Tip
지원되는 페이지 크기는 4K, 8K, 16K, 32K 입니다. 기본값은 4K 페이지입니다.

Tip
DB2 9.1부터 테이블스페이스 생성 시 Large 유형이 기본입니다.

Tip
LARGE 테이블스페이스는 반드시 DMS 유형이어야 합니다.

1 테이블스페이스는 한 개 이상의 컨테이너 묶음에 대한 논리적인 이름입니다. 컨테이너는 디렉토리, 파일, 논리적 파티션 중의 한 가지입니다.

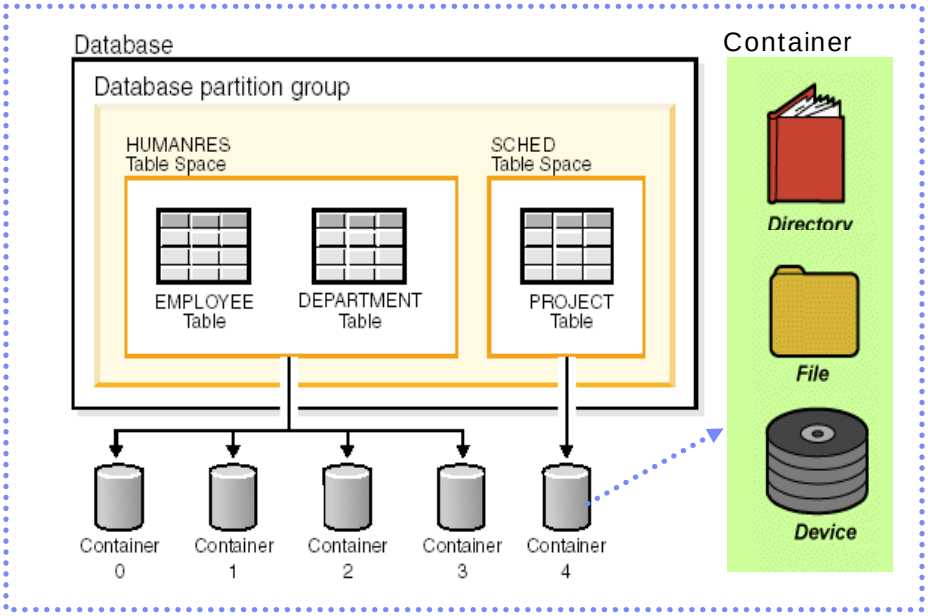


Figure 0701A ··· 테이블스페이스와 다양한 컨테이너

2 데이터베이스를 생성하면 3개의 기본 테이블스페이스가 자동으로 생성됩니다.

테이블스페이스	설명
SYSCATSPACE	시스템 카탈로그 테이블을 저장합니다.
TEMPSPACE1	SQL문을 처리하기 위한 임시 데이터를 저장합니다.
USERSPACE1	사용자 테이블을 저장합니다.

3 테이블스페이스는 관리 주체에 따라 2가지의 방식으로 구분됩니다.

방식	설명
SMS	System Managed Space의 약자입니다. 필요 시 OS가 컨테이너로 지정된 디렉토리에 파일을 할당하고 I/O를 관리합니다.
DMS	Database Managed Space 의 약자입니다. 미리 정의된 컨테이너를 이용하여 DB 엔진이 직접 I/O를 관리합니다.

4 테이블스페이스는 용도에 의해 4가지의 유형으로 구분됩니다.

유형	설명
REGULAR	모든 영구 데이터를 저장. SMS(기본), DMS (지원)
LARGE	모든 영구 데이터를 저장. DMS(기본), SMS(지원안함)
SYSTEM TEMPORARY	엔진이 SQL문을 실행하기 위해 임시로 사용하는 영역
USER TEMPORARY	세션의 전역 임시 테이블의 데이터를 임시로 저장하는 영역

TOPIC 07-02 기본 테이블스페이스

Point

데이터베이스를 생성하면 기본적으로 SYSCATSPACE, TEMPSPACE1, USERSPACE1 이라는 세 개의 테이블스페이스가 SMS 유형으로 생성됩니다. CREATE DB 문에서 사용자가 유형과 컨테이너 정보를 지정할 수도 있습니다.

Tip

SYSCATSPACE 에는 사용자의 테이블을 저장할 수 없으며, 사용자가 제거할 수 없습니다.

Tip

DB2 9.7에서 list table space [show detail]가 Deprecate 되어, 향후 릴리스에서 지원되지 않을 예정입니다. 9.7부터는 MON_GET_TABLESPACE, MON_GET_CONTAINER 의 사용을 권장합니다.

Tip

기본 생성된 USERSPACE1, TEMPSPACE1의 기능에 대응하는 테이블스페이스를 사용자가 생성한 후에는 제거할 수도 있습니다.

1 <인스턴스 사용자>로 로그인합니다.

```
$ login <인스턴스 사용자명>
```

2 create database 문에서 특별한 옵션을 지정하지 않으면, 3개의 기본 테이블스페이스는 SMS 유형으로 기본 디렉토리에 생성됩니다. SYSCATSPACE 는 시스템 카탈로그 테이블을 저장하며, TEMPSPACE1은 SQL문의 처리시에 정렬 또는 조인 작업을 위해 엔진이 사용하는 임시 공간입니다. USERSPACE1은 사용자의 테이블이 저장될 수 있는 공간입니다.

```
$ db2 create db <데이터베이스명>
$ db2 connect to <데이터베이스명>
$ db2 list tablespaces
또는 db2 “select * from table(mon_get_tablespace(‘’,-2)) as t”
$ db2 list tablespace containers for 0
또는 db2 “select * from table(mon_get_container(‘’,-2)) as t”
```

3 create database 문에서 3개의 기본 테이블스페이스의 유형 및 컨테이너에 대한 정보를 사용자가 지정할 수 있습니다.

```
$  vi <임의의 파일명>
create db <데이터베이스명>
CATALOG TABLESPACE <테이블스페이스 옵션>
USER TABLESPACE <테이블스페이스 옵션>
TEMP TABLESPACE <테이블스페이스 옵션>;
$ db2 –svtf <임의의 파일명>
```

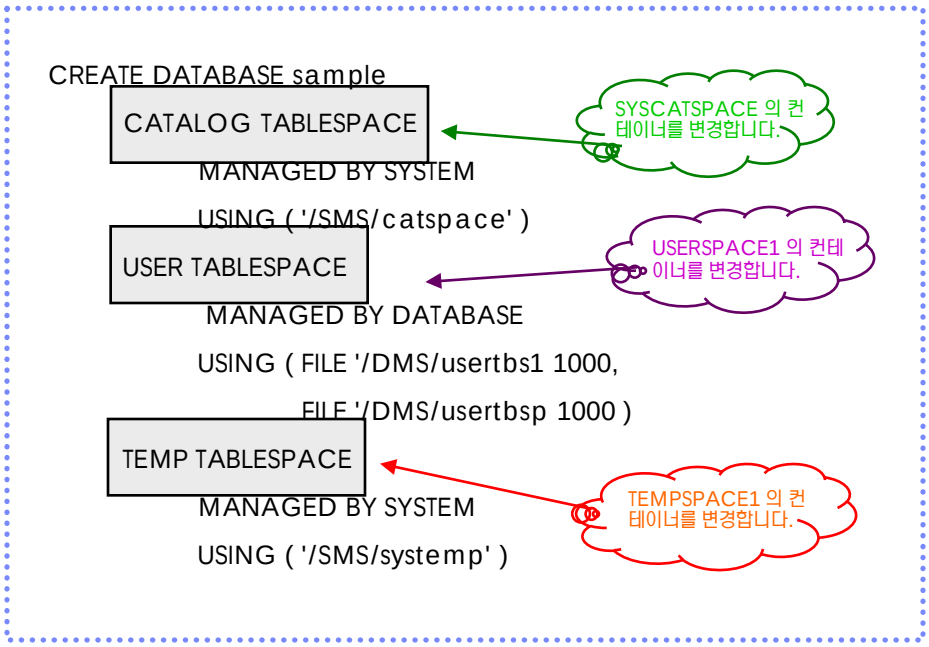


Figure 0702A ••• 기본 테이블스페이스 변경

TOPIC 07-03 SMS 테이블스페이스

Point

사용자는 한 개 이상의 디렉토리를 컨테이너로 지정하면 OS가 지정된 디렉토리에 파일을 생성하여 필요 시에 자동으로 공간을 할당하거나 축소하면서 관리하는 방식입니다. 시스템 임시 테이블스페이스 또는 사용자 임시 테이블스페이스에 주로 사용됩니다.

Tip

SMS 방식은 컨테이너의 크기를 지정하지 않습니다. 지정한 디렉토리가 속한 파일 시스템의 가용 공간이 없을 때까지 증가할 수 있습니다.

Tip

컨테이너로 사용되는 디렉토리에 생성되는 파일의 이름은 DB2 엔진에 의해 결정되며, OS에 의해 필요 시에 결정됩니다.

1. <인스턴스 사용자>로 로그인합니다.

```
$ login <인스턴스 사용자명>
```

2. 테이블스페이스를 생성하려면 데이터베이스에 대한 접속이 필요합니다.

```
$  db2 connect to <데이터베이스명>
```

3. create tablespace 문에서 MANAGED BY SYSTEM 옵션을 이용하여 SMS 방식의 테이블스페이스를 생성합니다. USING 옵션으로 한 개 이상의 디렉토리명을 나열합니다.

```
$ db2 “create tablespace <테이블스페이스명> managed by system
    using (‘<디렉토리명>’)”
$ db2 “create tablespace <테이블스페이스명> managed by system
    using (‘<디렉토리명 1>’, ‘<디렉토리명 2>’)”
```

4. ls 명령어를 이용하여 컨테이너로 사용되는 디렉토리를 확인할 수 있습니다. 컨테이너로 사용되는 디렉토리 또는 OS가 관리하는 파일이 손상되면 데이터베이스가 손상됩니다.

```
$ ls –lia <디렉토리명>
```

5. 시스템 임시 테이블스페이스와 사용자 임시 테이블스페이스에 저장되는 데이터는 한시적이므로 가능한 SMS 방식으로 생성하도록 합니다.

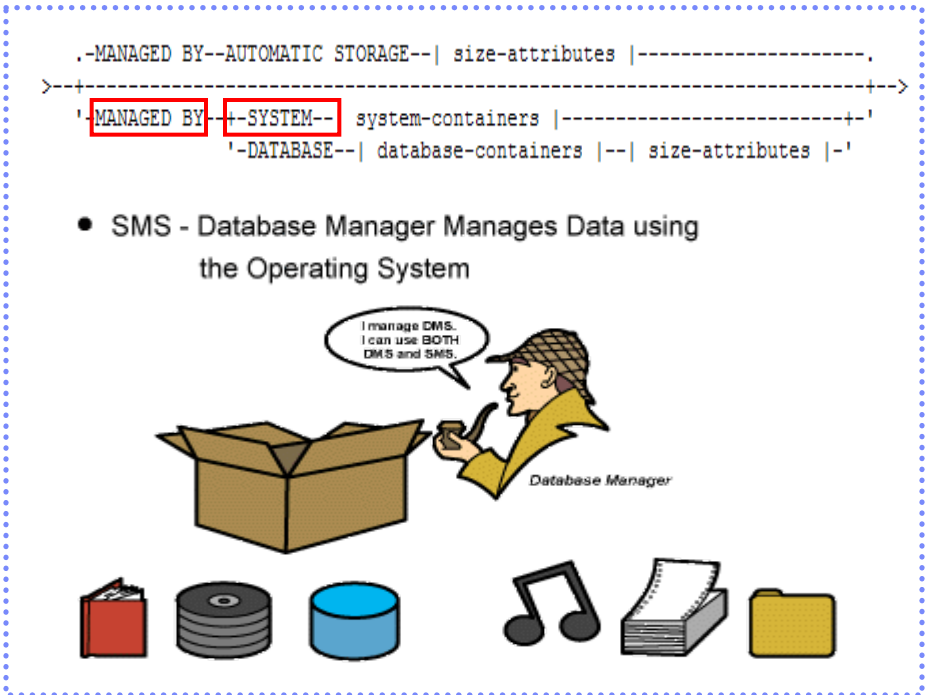


Figure 0703A SMS 방식 테이블스페이스

TOPIC 0704 DMS 테이블스페이스

Point

미리 정의한 한 개 이상의 컨테이너에 데이터베이스 관리 시스템이 직접 I/O 를 관리하는 방식입니다. 테이블스페이스 생성시에 파일 또는 디바이스 유형의 컨테이너의 이름과 크기가 결정되며, 향후에 컨테이너에 대한 변경 작업이 가능합니다.

Tip

CIO 또는 DIO를 지원하는 OS환경에서 NO FILE SYSTEM CACHING 기능을 사용하면, 파일 컨테이너를 사용하는 DMS 방식의 테이블스페이스도 디바이스 컨테이너와 동일한 성능을 냅니다.

Tip

DB2 9.5 이후 직접입출력을 지원하는 NO FILE SYSTEM CACHING 기능이 기본 설정 입니다.

Tip

DB2 9.7부터 기본 파일 시스템이 GPFS인 경우에도 NO FILE SYSTEM CACHING이 기본 설정 입니다.

Tip

DMS 유형의 테이블스페이스는 생성 후에도 컨테이너 구성을 변경할 수 있습니다.

1. <인스턴스 사용자>로 로그인합니다.

```
$ login <인스턴스 사용자명>
```

2. 테이블스페이스를 생성하려면 데이터베이스에 대한 접속이 필요합니다.

```
$  db2 connect to <데이터베이스명>
```

3. create tablespace 문에서 MANAGED BY DATABASE 옵션을 이용하여 DMS 방식의 테이블스페이스를 생성합니다.

```
$ db2 "create tablespace <테이블스페이스명> managed by
       database using (file '<파일명>'  <크기>) "
$ db2 "create tablespace <테이블스페이스명> managed by
       database using (device '<디바이스파일명>'  <크기>) "
```

4. ls 명령어로 컨테이너로 사용되는 파일과 논리적 볼륨에 대응되는 디바이스 파일을 확인할 수 있습니다. 파일 또는 논리적 볼륨이 손상되면 데이터베이스가 손상됩니다.

```
$ ls –lia <파일명>
$ ls –lia <디바이스파일명>
$ lslv –l <논리적파티션명>
```

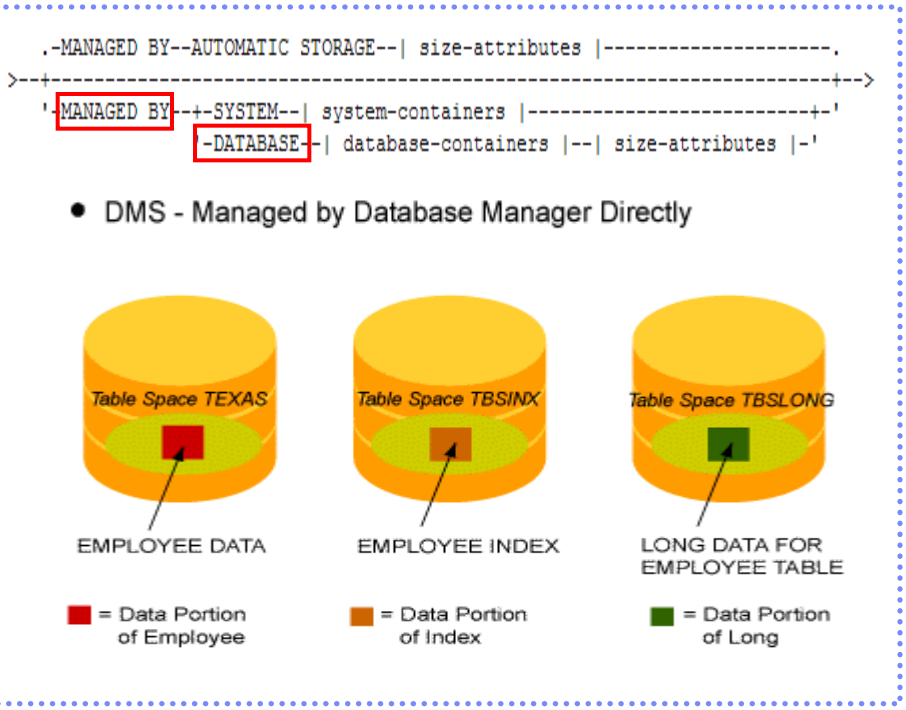


Figure 0704A... DMS 방식 테이블스페이스

TOPIC 07 05 LARGE 테이블스페이스

Point

한 테이블의 데이터를 테이블, 인덱스, LONG 데이터로 분리해서 저장할 때 LONG 데이터는 DMS 유형의 LARGE 테이블스페이스에 저장합니다. 기본적으로는 LARGE 테이블스페이스에 테이블, 인덱스 데이터와 함께 저장됩니다.

Tip

DB2 9.1 이후 Regular 기본값이 사라지고 유형을 지정하지 않을경우 managed by 절에서 테이블스페이스 유형을 지정합니다. Large 옵션이 기본값 입니다.

Tip

DMS 방식의 LARGE 유형으로 2개의 테이블스페이스를 생성하면, 한 테이블의 테이블 데이터와 인덱스 데이터를 따로 저장할 수 있습니다.

Tip

테이블을 생성하는 create table 문에서 IN <테이블스페이스명> INDEX IN <테이블스페이스명> 옵션으로 사용됩니다.

Tip

테이블을 생성하는 create table 문에서 LONG IN <LARGE 테이블스페이스명> 옵션으로 사용됩니다.

1 <인스턴스 사용자>로 로그인하여 데이터베이스에 접속합니다.

```
$ login <인스턴스 사용자명>
$ db2 connect to <데이터베이스명>
```

2 create tablespace 문에서 DMS 방식의 LARGE 유형으로 테이블스페이스를 생성합니다.

```
$ db2 "create tablespace <테이블스페이스명> managed by database
    using (file '<파일명>' <크기>)"
$ db2 "create tablespace <테이블스페이스명> managed by database
    using (device '<디바이스파일명>' <크기>) "
```

3 SYSCAT.TABLESPACES의 DATATYPE 컬럼의 값은 'L' 로 표시됩니다.

```
$ db2 "select substr(tbspace,1,20) tbspace, datatype, tbspaceid,
    tbspacetype from syscat.tablespaces"
```

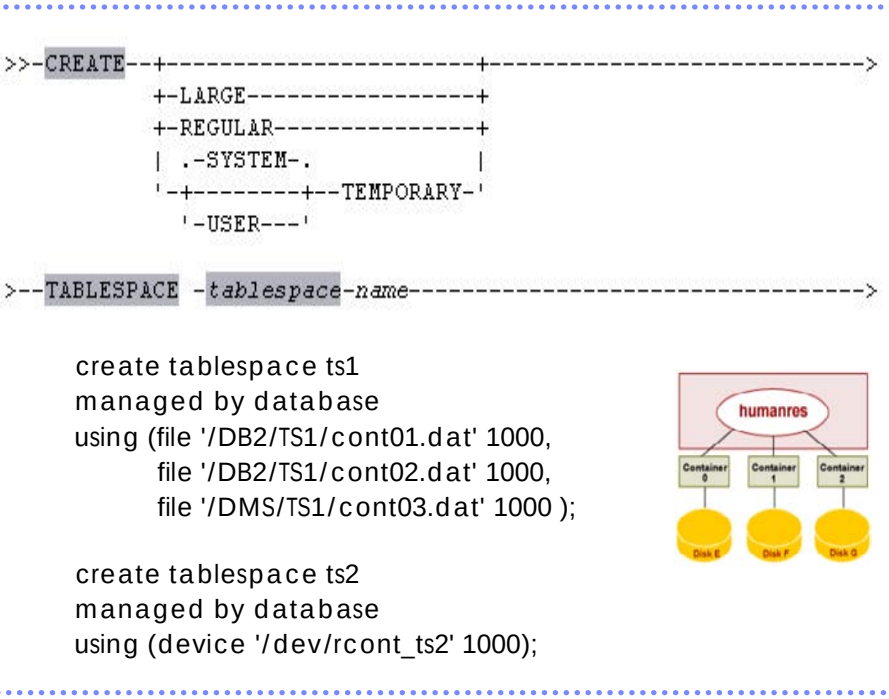


Figure 0705A LARGE 유형의 테이블스페이스

TOPIC 07-06 SYSTEM TEMPORARY 테이블스페이스

Point

요청된 SQL문을 처리하기 위해 DB2 엔진이 임시로 생성하는 파일 또는 테이블을 저장하는 테이블스페이스를 시스템 임시 테이블스페이스라고 합니다. 기본적으로 TEMPSPACE1 이 생성되며, 추가적으로 생성할 수 있습니다.

Tip

4K 이외의 페이지 크기를 가진 테이블스페이스가 존재하면 해당 페이지와 동일한 크기를 가진 시스템 임시 테이블스페이스가 필요합니다.

1. <인스턴스 사용자>로 로그인하여 데이터베이스에 접속합니다.

```
$ login <인스턴스 사용자명>
$ db2 connect to <데이터베이스명>
```

2. create tablespace 문에서 SYSTEM TEMPORARY 옵션을 이용하여 SMS 방식의 SYSTEM TEMPORARY 유형으로 테이블스페이스를 생성합니다.

```
$ db2 "create system temporary tablespace <테이블스페이스명>
    managed by system using ('<디렉토리명>')"
```

Tip

DMS 방식도 지원되지만, SMS 방식을 사용하는 것이 일반적입니다. 그러나 Large 파일이 자주 생성된다면, DMS를 고려하십시오.

3. create tablespace 문에서 SYSTEM TEMPORARY 옵션을 이용하여 DMS 방식의 SYSTEM TEMPORARY 유형으로 테이블스페이스를 생성합니다.

```
$ db2 "create system temporary tablespace <테이블스페이스명>
    managed by database using (file '<파일명>' <크기>)"
$ db2 "create system temporary tablespace <테이블스페이스명>
    managed by database using (device '<디바이스파일명>' <크기>)"
```

4. SYSCAT.TABLESPACES의 DATATYPE 컬럼의 값은 'T' 로 표시됩니다.

```
$ db2 "select substr(tbspace,1,20) tbspace, datatype, tbspaceid,
    tbspacetype from syscat.tablespaces"
```

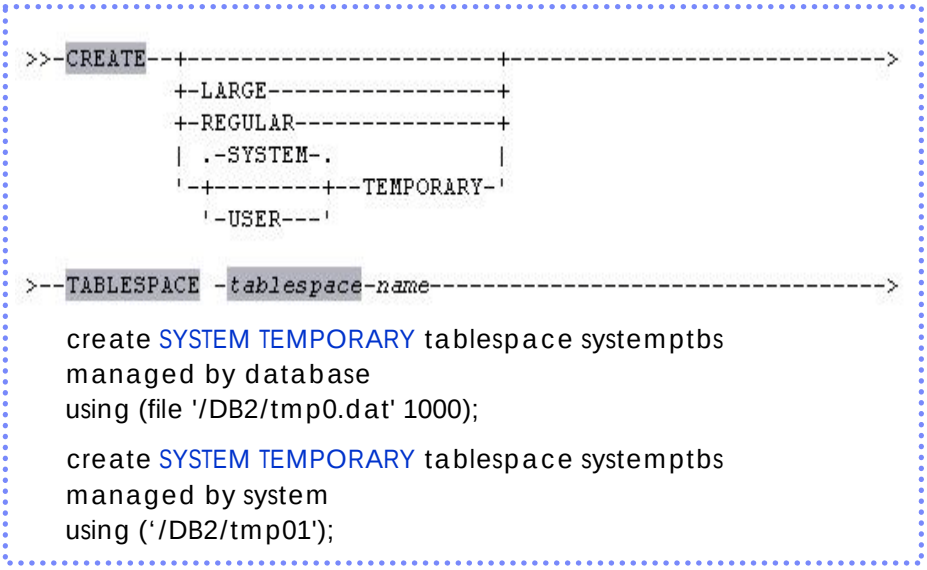


Figure 0706A ··· 시스템 임시 테이블스페이스 생성

TOPIC 0707 USER TEMPORARY 테이블스페이스

Point

세션별로 임시 테이블을 정의하여 일반 테이블과 동일하게 데이터를 처리할 수 있습니다. 임시 테이블은 세션 별로 독립적이며, 세션이 실행되는 동안에만 사용자 임시 테이블스페이스에 저장됩니다. 기본적으로 제공되는 사용자 임시 테이블스페이스는 없습니다.

Tip

임시 테이블을 사용하려면 반드시 사용자 임시 테이블스페이스를 미리 생성해야 합니다.

1 <인스턴스 사용자>로 로그인하여 데이터베이스에 접속합니다.

```
$ login <인스턴스 사용자명>
$ db2 connect to <데이터베이스명>
```

2 create tablespace 문에서 USER TEMPORARY 옵션을 이용하여 SMS 방식의 USER TEMPORARY 유형으로 테이블스페이스를 생성합니다.

```
$ db2 “create user temporary tablespace <테이블스페이스명> managed
    by system using (‘<디렉토리명>’)”
```

Tip

DMS 방식도 지원되지만, SMS 방식을 사용하는 것이 일반적입니다. 그러나 Large 파일이 자주 생성된다면, DMS를 고려하십시오.

3 create tablespace 문에서 USER TEMPORARY 옵션을 이용하여 DMS 방식의 USER TEMPORARY 유형으로 테이블스페이스를 생성합니다.

```
$ db2 “create user temporary tablespace <테이블스페이스명> managed
    by database using (file ‘<파일명>’ <크기>)”
$ db2 “create user temporary tablespace <테이블스페이스명> managed
    by database using (device ‘<디바이스파일명>' <크기>)“
```

4 SYSCAT.TABLESPACES의 DATATYPE 컬럼의 값은 ‘U’ 로 표시됩니다.

```
$ db2 "select substr(tbspace,1,20) tbspace, datatype, tbspaceid,
    tbspacetype from syscat.tablespaces"
```

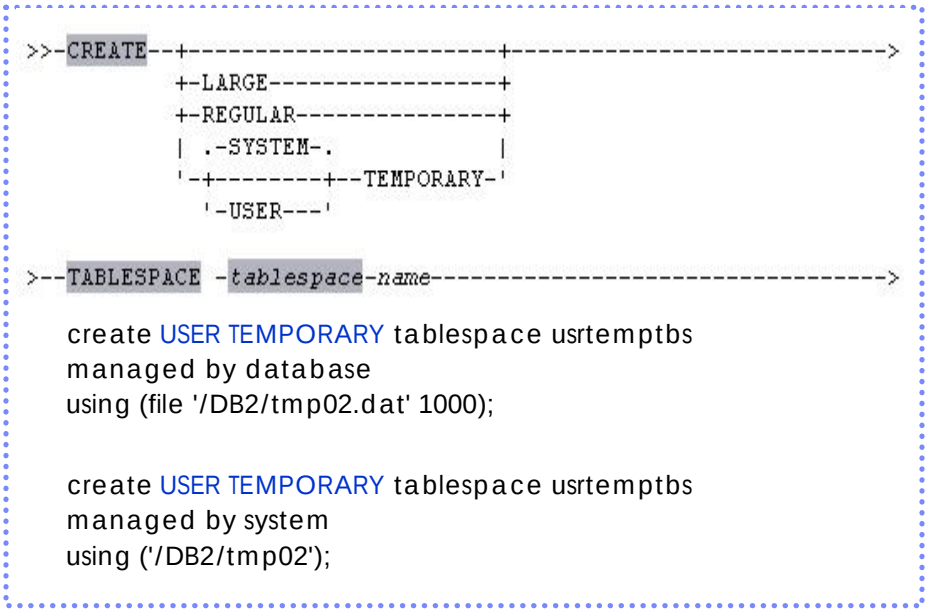


Figure 0707A ••• 사용자 임시 테이블스페이스 생성

TOPIC 07 08 디렉토리 컨테이너

Point

SMS 방식의 테이블스페이스는 디렉토리 유형의 컨테이너만 지원합니다. CREATE TABLESPACE 문의 USING 옵션에서 컨테이너로 사용할 디렉토리명을 지정합니다. 컨테이너로 지정된 디렉토리는 자동으로 생성됩니다.

Tip

지정한 디렉토리를 생성할 수 없으면, SQL0970N 오류가 반환됩니다.

Tip

컨테이너로 지정한 디렉토리에 파일이 존재하는 경우에는 SQL0298N 이 반환됩니다. 파일시스템 생성 시 기본 생성되는 (lost+found) 도 삭제해야 합니다.

Tip

DB2 9.7에서 list tablespaces[show detail] 이 Deprecate되었습니다.

Tip

디렉토리 유형의 컨테이너는 OS에 의해 필요할 때 공간이 할당되므로, <사용 가능한 페이지 수>는 대부분 <전체 페이지 수>와 일치합니다.

1. <인스턴스 사용자>로 로그인하여 데이터베이스에 접속합니다.

```
$ login <인스턴스 사용자명>
$ db2 connect to <데이터베이스명>
```

2. create tablespace 문의 USING 옵션에서 디렉토리명을 지정합니다. <디렉토리명>은 <절대 경로명> 또는 <상대 경로명>으로 지정할 수 있습니다.

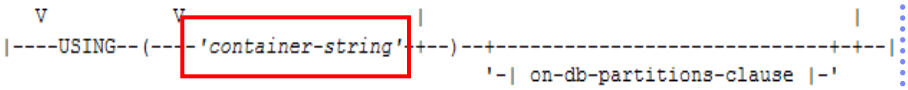
```
$ db2 "create tablespace <테이블스페이스명> managed by system
    using ('<디렉토리명>')"
$ db2 "create tablespace <테이블스페이스명> managed by system
    using ('<디렉토리명 1>', '<디렉토리명 2>')"
```

3. 컨테이너로 지정된 디렉토리는 자동으로 생성됩니다.

```
$ ls –lia <디렉토리명>
```

4. 테이블스페이스와 컨테이너의 연관 정보는 데이터베이스 내부의 제어 파일에 저장됩니다. list tablespaces 명령어를 이용하여 <테이블스페이스 ID>를 확인하고, list tablespace containers 명령어로 <컨테이너 유형> 항목이 '경로' 인 것을 확인합니다.

```
$ db2 list tablespaces show detail
또는 db2 "select * from table(mon_get_tablespace('',-2)) as t"
$ db2 list tablespace containers for <테이블스페이스 ID> show detail
또는 db2 "select * from table(mon_get_container('',-2)) as t"
```



UNIX :

```
create system temporary tablespace temptbs
managed by database
using (file '/DB2/tmp01/cont1' 1000,
        file '/DB2/tmp02'/cont2' 1000);
```

Intel :

```
create system temporary tablespace temptbs
managed by database
using (file 'C:\DB2\tmp01\cont1' 1000,
        file 'C:\DB2\tmp02\cont2' 1000);
```

Figure 0708A... 디렉토리 컨테이너를 가지는 SMS 테이블스페이스

TOPIC 07-09 파일 컨테이너

Point

DMS 방식의 테이블스페이스는 파일 유형의 컨테이너를 지원합니다. CREATE TABLESPACE 문의 USING 옵션에서 컨테이너의 유형을 file로 지정하고, 컨테이너로 사용될 파일명과 크기를 지정합니다. 컨테이너로 지정된 파일은 자동으로 생성됩니다.

Tip
컨테이너로 사용되는 파일들의 크기는 동일한 것이 좋습니다.

Tip
지정한 파일을 생성할 수 없으면, SQL0970N 오류가 반환됩니다.

Tip
컨테이너의 크기는 적어도 <2 * 익스텐트의 크기> 페이지 이상이어야 합니다. 파일의 크기가 유효하지 않으면 SQL1422N 가 반환됩니다.

Tip
<전체 페이지 수> 이상의 공간은 사용될 수 없으므로, <사용 가능한 페이지 수>가 0 이 되면, SQL0289N이 반환됩니다.

Tip
DB2 9.7에서 list tablespaces[show detail] 이 Deprecate되었습니다.

1 <인스턴스 사용자>로 로그인하여 데이터베이스에 접속합니다.

```
$ login <인스턴스 사용자명>
$ db2 connect to <데이터베이스명>
```

2 create tablespace 문의 USING 옵션에서 컨테이너의 유형을 file 이라고 지정하고, 컨테이너로 사용될 파일명과 크기를 지정합니다. <파일명>은 <절대 경로명> 또는 <상대 경로명>을 이용하여 지정할 수 있습니다. <크기>는 테이블스페이스의 페이지 단위로 지정합니다. K, M, G 단위를 이용한 지정도 가능합니다.

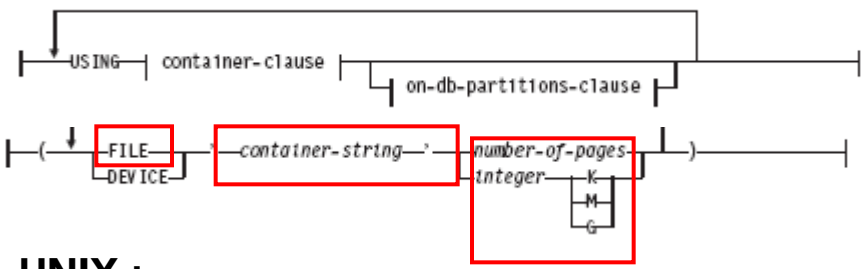
```
$ db2 "create tablespace <테이블스페이스명> managed by database
    using (file '<파일명1>' <크기>, file '<파일명2>' <크기>)"
```

3 컨테이너로 지정된 파일이 자동으로 생성됩니다.

```
$ ls –lia <파일명>
```

4 테이블스페이스와 컨테이너의 연관 정보는 데이터베이스 내부의 제어 파일에 저장됩니다. list tablespaces 명령어를 이용하여 <테이블스페이스 ID>를 확인하고, list tablespace containers 명령어로 <컨테이너 유형> 항목이 '파일' 인 것을 확인합니다.

```
$ db2 list tablespaces show detail
또는 db2 "select * from table(mon_get_tablespace('',-2)) as t"
$ db2 list tablespace containers for <테이블스페이스 ID> show detail
또는 db2 "select * from table(mon_get_container('',-2)) as t"
```



UNIX :

```
create tablespace indextbs managed by database
using (file '/DB2/INDEX/cont02.dat' 1024);
```

Intel :

```
create tablespace indextbs managed by database
using (file 'C:\DB2\INDEX\cont02.dat' 1024);
```

Figure 0709A ••• 파일 컨테이너를 가지는 DMS 테이블스페이스

TOPIC 07-10 디바이스 컨테이너

Point

DMS 방식의 테이블스페이스는 디바이스 유형의 컨테이너를 지원합니다. CREATE TABLESPACE 문의 USING 옵션에서 컨테이너의 유형을 device로 지정하고, 미리 생성된 디바이스파일명과 크기를 지정합니다.

Tip

논리적 파일명이 X 인 경우에 디바이스파일명은 /dev/rX 입니다.

Tip

지정한 디바이스파일의 소유자가 유효하지 않으면, SQL0970N 오류가 반환됩니다.

Tip

지정한 디바이스파일이 존재하지 않으면 SQL0298N 오류가 반환됩니다.

Tip

파일의 크기는 논리적 볼륨의 크기 이하입니다. 파일의 크기가 유효하지 않으면 SQL1422N 가 반환됩니다.

Tip

<전체 페이지 수> 이상의 공간은 사용될 수 없으므로, <사용 가능한 페이지 수>가 0 이 되면, SQL0289N이 반환됩니다.

Tip

DB2 9.7에서 list tablespaces[show detail] 이 Deprecate되었습니다.

1 root 사용자가 컨테이너로 사용될 논리적 볼륨을 정의합니다. 논리적 볼륨에 대한 디바이스파일의 소유자를 인스턴스 사용자로 변경합니다.

```
$ login root
$ mklv -y'<논리적파티션명>'  <볼륨그룹명> <논리파티션개수>
$ chown <인스턴스의 사용자명>.<인스턴스의 그룹명> <디바이스파일명>
$ ls –lia <디바이스파일명>
```

2 <인스턴스 사용자>로 로그인하고, 데이터베이스에 접속합니다.

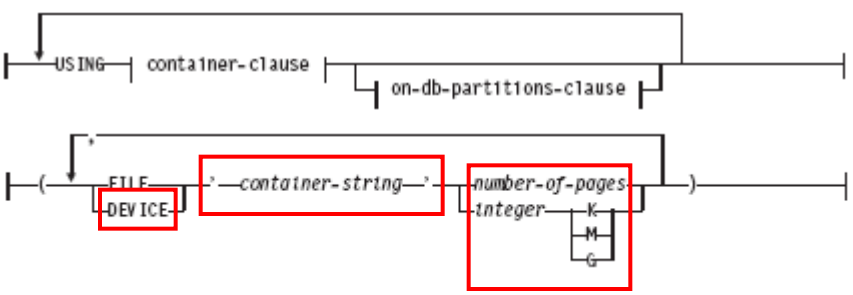
```
$ login <DB2 사용자>
$ db2 connect to <데이터베이스명>
```

3 create tablespace 문의 USING 옵션에서 컨테이너의 유형을 device 라고 지정하고, 컨테이너로 사용될 디바이스파일명과 크기를 지정합니다. <크기>는 테이블스페이스의 페이지 단위로 지정합니다. K, M, G 단위를 이용한 지정도 가능합니다.

```
$ db2 “create tablespace <테이블스페이스명> managed by database
    using (device ‘<디바이스파일명>’ <크기>)"
```

4 테이블스페이스와 컨테이너의 연관 정보는 데이터베이스 내부의 제어 파일에 저장됩니다. list tablespaces 명령어를 이용하여 <테이블스페이스 ID>를 확인하고, list tablespace containers 명령어로 <컨테이너 유형> 항목이 ‘디스크’ 인 것을 확인합니다.

```
$ db2 list tablespaces show detail
또는 db2 “select * from table(mon_get_tablespace(‘’,-2)) as t”
$ db2 list tablespace containers for <테이블스페이스 ID> show detail
또는 db2 “select * from table(mon_get_container(‘’,-2)) as t”
```



UNIX : create tablespace kestbs managed by database using (device '/dev/rcont03' 1024);

NT : create tablespace kestbs managed by database using (device '\\.\PhysicalDrive3' 1024);

Figure 0710A 디바이스를 컨테이너로 가진 DMS 테이블스페이스

TOPIC 07-11 페이지

Point

컨테이너에서 데이터의 입출력이 발생되는 기본 단위는 페이지이며, 지원되는 페이지의 크기는 4K, 8K, 16K, 32K 바이트입니다. 페이지의 크기는 CREATE TABLESPACE 문의 PAGESIZE 옵션으로 지정되며 향후 변경될 수 없습니다. .

Tip

테이블스페이스의 페이지 크기와 동일한 페이지 크기를 가지는 버퍼풀을 사용해야 합니다.

Tip

테이블스페이스가 생성되면 페이지의 크기는 변경할 수 없습니다.

Tip

테이블의 레코드 길이가 테이블 공간의 페이지 크기를 초과시에는 생성할 수 없습니다.

1. <인스턴스 사용자>로 로그인하여 데이터베이스에 접속합니다.

```
$ login <인스턴스 사용자명>
$ db2 connect to <데이터베이스명>
```

2. create tablespace 문의 PAGESIZE 옵션으로 페이지의 크기를 지정합니다. <크기>는 바이트 또는 K 단위를 이용하여 지정합니다.

```
$ db2 "create tablespace <테이블스페이스명> PAGESIZE <크기>
    managed by system using ('<디렉토리명>')"
```

3. db2look 유틸리티를 이용하여 테이블스페이스에 대한 DDL을 확인하면, PAGESIZE 옵션에서 페이지 크기를 바이트 단위로 확인할 수 있습니다.

```
$ db2look –d <데이터베이스명> -l | grep –p –i <테이블스페이스명>
```

4. 시스템 카탈로그를 조회하여 PAGESIZE 항목을 확인합니다.

```
$ db2 "select substr(tbspace,1,20) tbspace, tbspaceid, pagesize
    from syscat.tablespaces"
```

5. list tablespaces 명령어로 테이블스페이스의 정보를 확인합니다. <페이지 크기> 항목에서 페이지의 크기를 바이트 단위로 확인할 수 있습니다.

```
$ db2 list tablespaces show detail
또는 db2 "select * from table(mon_get_tablespace('',-2)) as t"
$ db2 list tablespace containers for <테이블스페이스 ID> show detail
또는 db2 "select * from table(mon_get_container('',-2)) as t"
```

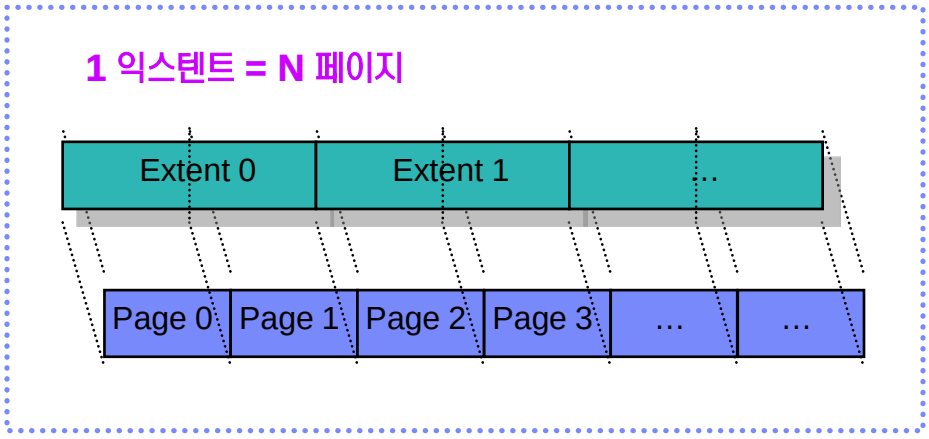


Figure 0711A ··· 페이지와 익스텐트

6. http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.dbobj.doc/doc/c0052318.html 에서 페이지의 크기 별 제약사항을 확인합니다.

TOPIC 07-12 익스텐트

Point

컨테이너에 데이터를 저장할 때 할당되는 기본 단위를 익스텐트라고 합니다. 한 개의 테이블스페이스는 여러 개의 컨테이너로 구성되므로, 데이터의 고른 분배를 위해 각 컨테이너에 익스텐트 크기만큼씩 라운드 로빈 방식으로 저장합니다.

Tip

한 익스텐트는 2개 이상의 페이지로 구성됩니다.

Tip

익스텐트의 크기를 명시적으로 지정하지 않으면, DFT_EXTENT_SZ 구성 변수에 정의된 값이 사용됩니다.

Tip

테이블스페이스가 생성되면 익스텐트 크기는 변경할 수 없습니다.

Tip

디스크의 RAID stripe set 의 크기와 일치시키는 것이 성능에 유리합니다.

1 <인스턴스 사용자>로 로그인하여 데이터베이스에 접속합니다.

```
$ login <인스턴스 사용자명>
$ db2 connect to <데이터베이스명>
```

2 create tablespace 문의 EXTENTSIZE 옵션으로 페이지의 크기를 지정합니다. <크기>는 테이블스페이스의 페이지 단위로 지정합니다. K, M, G 단위를 이용한 지정도 가능합니다.

```
$ db2 "create tablespace <테이블스페이스명> managed by system
    using ('<디렉토리명>') EXTENTSIZE <크기> "
```

3 db2look 유틸리티를 이용하여 테이블스페이스에 대한 DDL을 확인하면, EXTENTSIZE 옵션에서 익스텐트의 크기를 페이지 단위로 확인할 수 있습니다.

```
$ db2look –d <데이터베이스명> -l | grep –p –i <테이블스페이스명>
```

4 시스템 카탈로그를 조회하여 EXTENTSIZE 항목을 확인합니다.

```
$ db2 "select substr(tbspace,1,20) tbspace, tbspaceid, pagesize,
    extentsize from syscat.tablespaces"
```

5 list tablespaces 명령어로 테이블스페이스의 정보를 확인합니다. <Extent 크기> 항목에서 익스텐트의 크기를 페이지 단위로 확인할 수 있습니다.

```
$ db2 list tablespace containers for <테이블스페이스 ID> show detail
또는 db2 "select * from table(mon_get_container('',-2)) as t"
```

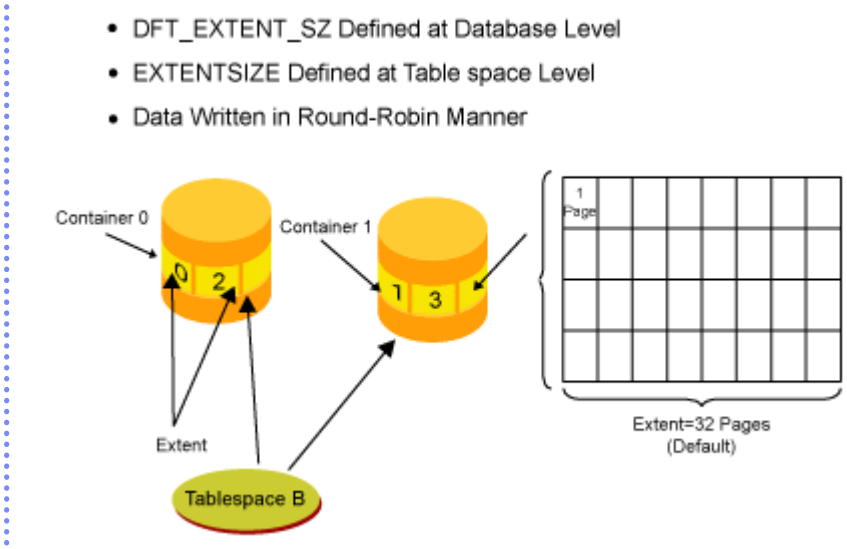


Figure 0712A • • • 익스텐트와 라운드 로빈 방식

TOPIC 07-13 페이지 클리너

Point

액세스 요청 시에 버퍼풀에 가용 공간이 없으면, 변경된 데이터를 컨테이너로 반영하여 가용 공간을 확보하므로 성능이 저하됩니다. 페이지 클리너 프로세스는 비동기 방식으로 버퍼풀의 변경된 페이지를 디스크로 미리 반영하여 버퍼풀에 가용 공간을 확보합니다.

Tip

'I/O 클리너' 라고도 합니다.

Tip

Automatic으로 지정하면 DB2가 자동으로 페이지 클리너 수를 정합니다.

Tip

NUM_IOCLEANERS 의 값은 데이터베이스가 사용하는 물리적인 디바이스의 개수 이하의 값으로 설정합니다.

Tip

SOFTMAX 구성 변수는 Crash 복구에 필요한 로그 파일의 개수에 영향을 주며, 기본값은 로그 파일 한 개의 크기인 100%로 설정됩니다.

Tip

CHNGPGS_THRESH 구성 변수는 버퍼풀이 허용하는 변경 페이지(dirty page)의 최대 비율입니다. 기본값은 60% 입니다.

Tip

DB2_USE_ALTERNATE_PAGE_CLEANING 레지스트리 변수 사용시 CHNGPGS_THRESH값은 무시되고 데이터베이스 관리 프로그램이 버퍼 풀에서 유지되는 더티 페이지 수를 자동으로 판별합니다 .

1 <인스턴스 사용자>로 로그인하여 데이터베이스 구성 변수인 NUM_IOCLEANERS를 이용하여 페이지 클리너용 프로세스의 개수를 숫자로 지정합니다.

```
$ login <인스턴스 사용자명>
$ db2 update db cfg for <데이터베이스> using NUM_IOCLEANERS<값>
```

2 데이터베이스를 재 활성화합니다.

```
$ db2 deactivate db <데이터베이스명>
$ db2 activate db <데이터베이스명>
```

3 db2pd 명령어를 이용하여 페이지 클리너를 담당하는 프로세스인 db2pclnr 를 확인합니다.

```
$ db2pd -d sample -edu | grep -i db2pclnr
```

4 페이지 클리너용 프로세스는 데이터베이스 구성 변수인 CHNGPGS_THRESH 와 SOFTMAX 값에 의해 비동기적으로 활동 시점이 결정됩니다.

```
$ db2 get db cfg for <데이터베이스명> | grep CHNGPGS_THRESH
$ db2 get db cfg for <데이터베이스명> | grep SOFTMAX
```

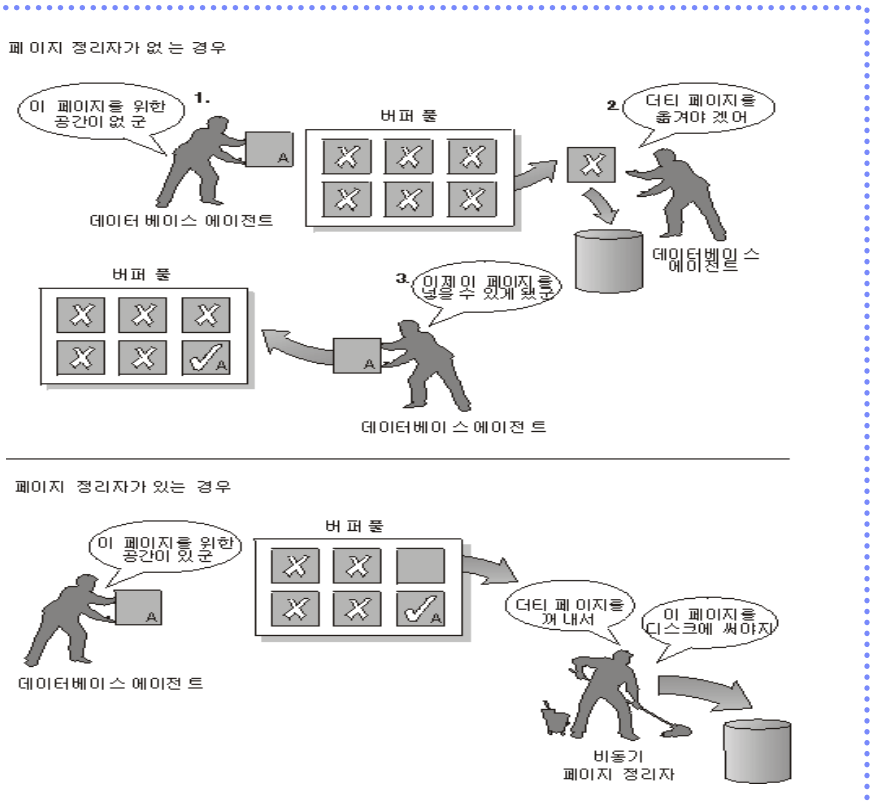


Figure 0713A 페이지 클리너

TOPIC 07-14 페이지 프리페처

Point

페이지 프리페치 프로세스는 비동기 방식으로 컨테이너에서 여러 페이지씩 미리 버퍼풀로 로드합니다. CREATE TABLESPACE 문의 PREFETCHSZ 옵션에서 익스텐트 크기의 배수로 설정하며, ALTER TABLESPACE 문으로 변경할 수 있습니다.

Tip

'I/O 프리페처' 라고도 합니다.

Tip

Automatic으로 지정하면 DB2가 자동으로 페이지 프리페처 수를 정합니다.

Tip

SEQDETECT 데이터베이스 구성 변수는 기본적으로 YES로 설정되어 있습니다. NO로 설정하면 프리페치를 실행하지 않습니다.

1 <인스턴스 사용자>로 로그인하여 데이터베이스 구성 변수인 NUM_IOSERVERS 를 이용하여 페이지 프리페처용 프로세스의 개수를 숫자로 지정하고, 데이터베이스를 재 활성화합니다

```
$ login <인스턴스 사용자명>
$ db2 update db cfg for <데이터베이스> using NUM_IOSERVERS <값>
$ db2 deactivate db <데이터베이스명>
$ db2 activate db <데이터베이스명>
```

2 ps 명령어를 이용하여 I/O 프리페치를 담당하는 프로세스인 db2pfchr를 확인합니다.

```
$ db2pd -d sample -edu | grep -i db2pfchr
```

3 테이블스페이스의 프리페치의 크기는 create tablespace문 또는 alter tablespace 문의 PREFETCHSZ 옵션으로 설정합니다. 익스텐트 크기의 배수로 설정하는 것이 좋습니다.

```
$ db2 "create tablespace <테이블스페이스명> managed by system
    using ('<디렉토리명>') PREFETCHSZ <크기> "
$ db2 alter tablespace <테이블스페이스명> PREFETCHSZ <크기>
```

4 프리페치의 유형은 다음과 같이 구분됩니다.

유형	이용시기
Index prefetch	인덱스 스캔, RUNSTATS, REORG 실행
Sequential prefetch	테이블 스캔
List prefetch	클러스터링되지 않은 테이블 스캔

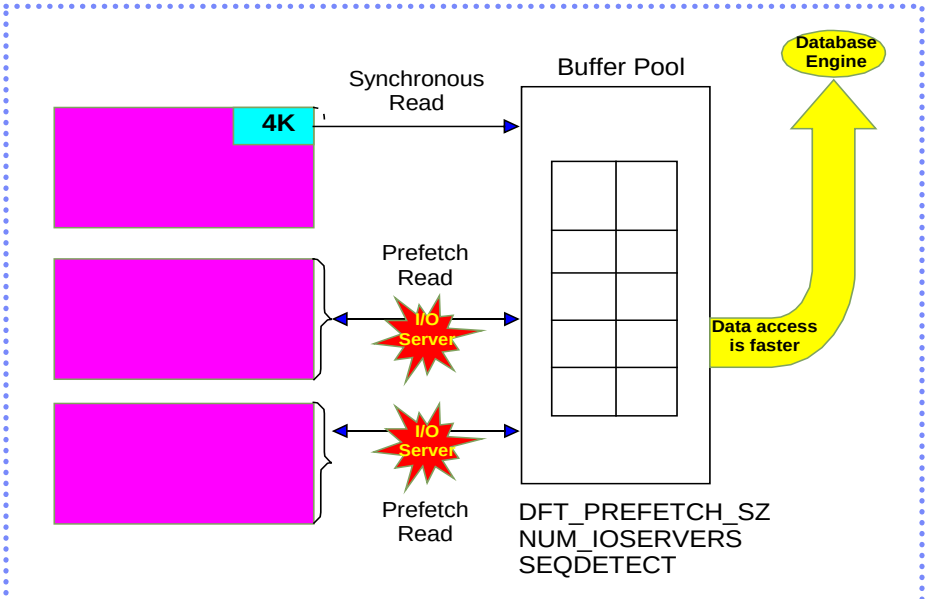


Figure 0714A ••• 페이지 프리페처

TOPIC 07-15 테이블스페이스 생성

Point

CREATE TABLESPACE 문을 이용하여 테이블스페이스를 생성합니다. 테이블스페이스의 관리 방식, 용도, 컨테이너 옵션, 페이지 크기, 익스텐트 크기, 프리페치 크기, 버퍼풀명 등을 지정하는 옵션이 있습니다.

1 create tablespace 문의 형식은 다음과 같습니다.

```
>>-CREATE--+----------------------+--------------------------->
           +-LARGE----------------+
           +-REGULAR--------------+
           | .-SYSTEM-.           |
           '-+--------+--TEMPORARY-'
             '-USER---'

>--TABLESPACE -tablespace-name-------------------------------->
>--+--------------------------+--------------------------------->
   '-PAGESIZE--integer--+---+-'
                        '-K-'
   .-MANAGED BY--AUTOMATIC STORAGE--| size-attributes |---------------------.
>--+-------------------------------------------------------------------------+-->
   '-MANAGED BY--+-SYSTEM--| system-containers |--------------------------+-'
                 '-DATABASE--| database-containers |--| size-attributes |-'

>--+-----------------------------------+-------------------------->
   '-EXTENTSIZE--+-number-of-pages-+-'
                 '-integer--+-K-+--'
                            '-M-'

>--+-----------------------------------+-------------------------->
   '-PREFETCHSIZE--+-AUTOMATIC-------+-'
                   +-number-of-pages-+
                   '-integer--+-K-+--'
                              +-M-+
                              '-G-'
```

Figure 0715A ··· create tablespace 문

2 create tablespace 문에서 사용되는 주요 옵션은 다음과 같습니다.

옵션	설명
<유형>	테이블스페이스에 저장될 데이터의 유형을 지정합니다. REGULAR, LARGE, SYSTEM TEMPORARY, USER TEMPORARY 중에 한 가지를 선택합니다. 생략하면 LARGE가 사용됩니다.
<테이블스페이스명>	18자 이내의 임의의 이름으로 생성합니다. 한 데이터베이스 내에서는 고유해야 합니다.
PAGESIZE <크기>	한 페이지의 크기를 바이트 단위로 지정합니다. 4K, 8K, 16K, 32K 중에 선택할 수 있습니다. 컨테이너의 기본 I/O 단위로 사용됩니다.
MANAGED BY <관리방식>	테이블스페이스의 컨테이너에 I/O를 관리하는 방식을 지정합니다. SMS 또는 DMS 방식 중에 한가지를 지정합니다.
USING (<컨테이너 옵션>)	관리 방식에 따라 디렉토리, 파일, 논리적 파티션 유형 중에 한 가지를 지정합니다. 컨테이너가 여러 개인 경우에는 ,(쉼표) 를 이용하여 반복적으로 지정합니다. 파일과 논리적 파티션 유형인 경우에는 PAGESIZE 옵션으로 지정한 페이지 단위로 크기를 지정합니다.
EXTENTSIZE <크기>	컨테이너에 데이터를 저장할 때 할당되는 기본 단위입니다. PAGESIZE 옵션으로 지정한 페이지 단위로 지정합니다.
PREFETCHSIZE <크기>	프리페치의 크기를 PAGESIZE에서 지정한 페이지 단위로 지정합니다. EXTENTSIZE 옵션에서 지정한 값의 배수로 설정하도록 합니다. 기본값으로 데이터베이스 구성 변수인 DFT_PREFETCH_SZ가 사용됩니다.
BUFFERPOOL <버퍼풀명>	사용할 버퍼풀명을 지정합니다. PAGESIZE 옵션에서 지정한 크기와 동일한 페이지 크기를 가진 버퍼풀을 지정할 수 있습니다. 다른 테이블스페이스가 사용하는 버퍼풀도 지정할 수 있습니다.

Tip

PAGESIZE의 기본 단위는 바이트입니다. K 단위로 표현할 수 있습니다.

Tip

EXTENTSIZE, PREFETCHSZ 의 기본 단위는 페이지입니다. K, M 단위로 표현할 수 있습니다.

Tip

액세스 요청이 빈번하고 데이터가 많은 테이블스페이스는 전용 버퍼풀을 지정하는 것이 좋습니다.

TOPIC 07-16 테이블스페이스 특성 변경

Point

ALTER TABLESPACE 문을 이용하여 기존의 모든 테이블스페이스의 프리페치 크기와 버퍼풀 명을 변경할 수 있습니다. 페이지 크기와 익스텐트 크기는 변경할 수 없습니다.

1 alter tablespace 문의 형식은 다음과 같습니다.

```
>>-ALTER TABLESPACE--tablespace-name--------------------------->
  +-+-EXTEND-+---+-| database-container-clause |-+--+-------------------------+
  | '-RESIZE-'   '-| all-containers-clause |-----'   '-| on-db-partitions-clause |-'
  +-REBALANCE-----------------------------------------------------------------
  +-PREFETCHSIZE--+-AUTOMATIC-------+-------------------------------------------
  |               +-number-of-pages-+
  |               '-integer--+-K-+--'
  |                          +-M-+
  |                          '-G-'
  +-BUFFERPOOL -bufferpool-name-----------------------------------------------
  +-OVERHEAD -number-of-milliseconds------------------------------------------
  +-TRANSFERRATE -number-of-milliseconds---------------------------------------
  +-+-FILE SYSTEM CACHING----+-------------------------------------------------
  | '-NO FILE SYSTEM CACHING-'
```

Figure 0716A • • • 테이블스페이스의 특성을 변경하는 alter tablespace 문

2 alter tablespace 문에서 사용되는 주요 옵션은 다음과 같습니다.

옵 션	이 용 시 기
PREFETCHSIZE	프리페치의 크기를 변경합니다.
BUFFERPOOL	사용할 버퍼풀 명을 변경합니다.
NO FILE SYSTEM CACHING	이중 버퍼링을 하지 않도록 변경합니다.

3 alter tablespace 문에서 PREFETCHSIZE 옵션으로 테이블스페이스의 프리페치 크기를 변경합니다.

```
$ db2 "alter tablespace <테이블스페이스명> PREFETCHSIZE <크기>"
```

4 alter tablespace 문에서 PREFETCHSIZE 옵션으로 AUTOMATIC을 지정하면 테이블스페이스의 컨테이너 수가 변경될 때마다 DB2 엔진이 프리페치 크기를 자동으로 갱신합니다. 갱신된 프리페치 크기는 데이터베이스가 재활성화될 때 적용됩니다. PREFETCHSIZE 절에 숫자 값을 지정하면 자동 갱신 모드는 해제됩니다.

```
$ db2 "alter tablespace <테이블스페이스명> PREFETCHSIZE
  AUTOMATIC"
```

5 alter tablespace 문에서 BUFFERPOOL 옵션으로 테이블스페이스가 사용하는 버퍼풀명을 변경합니다.

```
$ db2 "alter tablespace <테이블스페이스명> BUFFERPOOL <버퍼풀명>"
```

TOPIC 07-17 테이블스페이스 컨테이너 변경

Point

ALTER TABLESPACE 문에서 ADD, DROP, EXTEND, REDUCE, RESIZE 등의 옵션으로 DMS 테이블스페이스의 컨테이너에 대한 변경 작업이 가능합니다.

Tip

SMS 유형의 테이블스페이스는 컨테이너를 변경할 수 없습니다.

Tip

컨테이너를 DROP 할 때는 다른 컨테이너에 기존 데이터를 수용할 충분한 공간이 있어야 합니다.

Tip

개별적인 컨테이너를 명시하거나, ALL 키워드를 이용하여 모든 컨테이너를 지정합니다.

1 alter tablespace 문을 이용하여 DMS 유형의 테이블스페이스의 특성을 변경합니다.

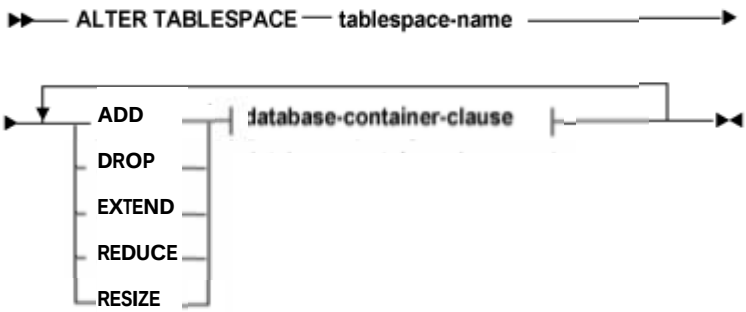


Figure 0717A ••• 컨테이너의 특성을 변경하는 alter tablespace 문

옵션	설명
ADD	새로운 컨테이너를 추가합니다.
DROP	기존 컨테이너를 제거합니다.
EXTEND	기존 컨테이너의 크기를 증가시킵니다.
REDUCE	기존 컨테이너의 크기를 감소시킵니다.
RESIZE	기존 컨테이너의 크기를 변경합니다.
AUTORESIZE	테이블 스페이스가 가득 차면 자동 증가합니다.

2 alter tablespace 문의 ADD, DROP, EXTEND, REDUCE, RESIZE 옵션으로 DMS 테이블스페이스의 컨테이너를 변경합니다.

```
$ db2 “alter tablespace <테이블스페이스명> ADD <컨테이너 정의>"
$ db2 “alter tablespace <테이블스페이스명> EXTEND <컨테이너 정의>"
$ db2 “alter tablespace <테이블스페이스명> REDUCE <컨테이너 정의>"
$ db2 “alter tablespace <테이블스페이스명> RESIZE <컨테이너 정의>"
$ db2 “alter tablespace <테이블스페이스명> DROP <컨테이너 정의>“
$ db2 “alter tablespace <테이블스페이스명> AUTORESIZE YES”
```

```
create tablespace ts1 managed by database using (file
    '/DB2/ts1/cont1.dat' 2000)
alter tablespace ts1 add (file '/DB2/ts1/cont2.dat' 5000)
alter tablespace ts1 extend (file '/DB2/ts1/cont1.dat' 2000)
alter tablespace ts1 reduce (file '/DB2/ts1/cont2.dat' 4000)
alter tablespace ts1 resize (all 6000)
alter tablespace ts1 drop (file ‘/DB2/ts1/cont1.dat')
alter tablespace ts1 autoresize yes
```

Figure 0717B ••• 컨테이너의 추가, 제거, 변경

TOPIC 07-18 High Water Mark 조정

Point

HWM(하이워터마크)로 인해 빈 공간이 많지만 테이블스페이스 사이즈를 줄일 수 없을 경우 하이워터 마크를 내리고 사이즈를 변경 할 수 있습니다. Automatic Storage와 파일로 구성되는 DMS 테이블 공간에 해당됩니다.

Tip

일반 DMS 는 lower high water mark 수행 후 reduce를 따로 진행합니다.

1. Automatic storage table space 인 경우 HWM 사이즈 및 tablespace 사이즈를 줄일 수 있습니다.

```
ALTER TABLESPACE <tsname> REDUCE --+-----------------------------+---
                                   '-- <size> --+-------------+--+--'
                                                | +----- K -----+ |
                                                | +----- M -----+ |
                                                | +----- G -----+ |
                                                | '-- PERCENT --' |
                                                '--- MAX ---------------------'
                                                '--- STOP --------------------'
```

2. DMS table space 인 경우 다음 명령어로 HWM을 줄일 수 있습니다.

```
ALTER TABLESPACE <tsname> LOWER HIGH WATER MARK --+----------+---
                                                 '-- STOP --'
```

```
db2 "select varchar(tbsp_name, 15) as tbsp_name,
    tbsp_free_pages,tbsp_total_pages, tbsp_page_top from table
    (mon_get_tablespace('TS01',-2)) as t"

TBSP_NAME       TBSP_FREE_PAGES     TBSP_TOTAL_PAGES    TBSP_PAGE_TOP
--------------- ------------------- ------------------- -------------------
TS01                      46080               90112               87904

  1 record(s) selected

db2 "alter tablespace ts01 reduce max"

DB20000I  The SQL command completed successfully.
```

Figure 0718A ••• 빈 공간과 하이워터 마크 확인 후 reduce 진행

3. 테이블 오브젝트 삭제 후 HWM 조정 도식을 설명합니다.

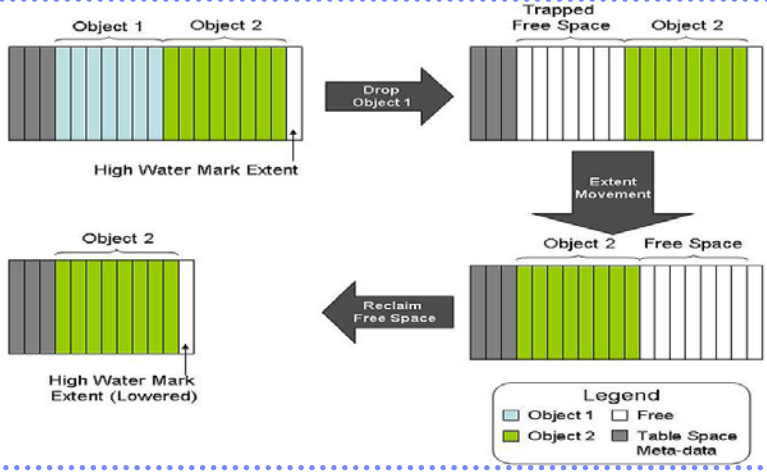


Figure 0718B ••• 하이 워터 마크 조정

TOPIC 07-19 테이블스페이스 제거

Point

DROP TABLESPACE 문을 이용하여 테이블스페이스를 제거합니다. 테이블스페이스를 위한 컨테이너도 모두 제거되며, 테이블스페이스에 저장된 테이블과 그 데이터도 모두 제거됩니다.

1 drop tablespace 문의 형식은 다음과 같습니다.

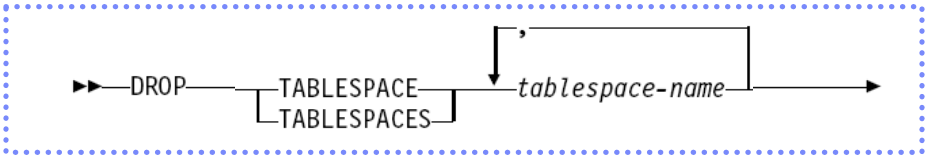


Figure 0719A drop tablespace 문

2 drop tablespace 문으로 특정한 테이블스페이스를 제거합니다.

```
$ db2 “drop tablespace <테이블스페이스명>”
```

3 한 테이블이 여러 테이블스페이스에 분리되어 정의된 경우에는 관련된 테이블스페이스를 한 개의 drop tablespace 문으로 함께 제거합니다. 일부 테이블스페이스만 제거하려고 하면 SQL0282N 오류가 반환됩니다.

```
$ db2look –d <데이터베이스명> -e  –t  <테이블명>
$ db2 “drop tablespace <테이블스페이스명 1>, <테이블스페이스명 2>”
```

4 SMS 테이블스페이스를 제거하면, 디렉토리에 생성된 데이터 파일은 제거됩니다. 디렉토리와 제어 파일은 제거되지 않으므로 rm 명령어로 제거합니다.

```
$ ls –lia <디렉토리명>
$ rm -Rf <디렉토리명>
```

5 DMS 테이블스페이스를 제거하면, 컨테이너로 사용된 파일은 제거됩니다.

```
$ ls -lia <파일명>
```

6 DMS 테이블스페이스를 제거하면 컨테이너로 사용된 논리적 파티션은 다른 테이블스페이스의 컨테이너로 사용될 수 있습니다. 사용자가 생성한 논리적 파티션은 자동으로 제거되지 않으므로 필요시에는 rmlv 명령어로 제거합니다.

```
$ ls -lia <디바이스파일명>
$ rmlv –f  '<논리적파티션명>'
```

7 테이블스페이스를 제거하면 컨테이너에 저장된 태그 정보도 자동으로 제거되므로, 해당 컨테이너는 다른 테이블스페이스에서 사용될 수 있습니다. 비정상적인 방법으로 테이블스페이스를 제거하면 엔진이 생성한 태그 정보가 남게 되므로 db2untag 명령어를 이용하여 강제로 태그 정보를 제거하여 다른 테이블스페이스가 사용할 수 있도록 합니다.

```
$ db2untag –f <파일명>
$ db2untag –f <디바이스파일명>
```

Tip

제거된 테이블스페이스의 ID는 다른 테이블스페이스를 생성하면 다시 사용됩니다.

Tip

데이터베이스를 제거하면 모든 테이블스페이스는 제거됩니다. 제거되지 않은 컨테이너는 db2untag 명령어로 태그 정보를 제거합니다.

TOPIC 07-20 SYSCAT.TABLESPACES 뷰

Point

CREATE TABLESPACE문, ALTER TABLESPACE문, DROP TABLESPACE문으로 관리하는 테이블스페이스에 대한 정보는 시스템 카탈로그 뷰인 SYSCAT.TABLESPACES 를 통해 확인할 수 있습니다.

1 describe table 명령어로 SYSCAT.TABLESPACES 뷰의 컬럼에 대한 정보를 확인합니다.

```
$ login <인스턴스 사용자명>
$ db2 conect to <데이터베이스명>
$ db2 describe table SYSCAT.TABLESPACES
```

2 SYSCAT.TABLESPACES 뷰의 주요 컬럼에 대한 설명은 다음과 같습니다.

컬 럼	설 명
TBSPACE	테이블스페이스의 이름으로 한 데이터베이스에서 고유합니다.
TBSPACEID	관리를 위해 자동으로 할당되는 테이블스페이스의 ID입니다.
TBSPACETYPE	테이블스페이스의 관리 방식을 표시합니다. 'S' 는 SMS, 'D' 는 DMS를 의미합니다.
DATATYPE	테이블스페이스에 저장된 데이터의 유형 또는 용도를 표시합니다. 'A' 는 REGULAR, 'L' 은 LARGE, 'T' 는 SYSTEM TEMPORARY, 'U' 는 USER TEMPORARY를 나타냅니다.
EXTENTSIZE	익스텐트의 크기를 페이지 단위로 표시합니다.
PREFETCHSIZE	프리페치의 크기를 페이지 단위로 표시합니다.
PAGESIZE	페이지의 크기를 바이트 단위로 표시합니다.
BUFFERPOOLID	사용하는 버퍼풀명을 표시합니다.

Tip

db2look 유틸리티를 이용하여 테이블스페이스에 대한 DDL을 추출할 수 있습니다.

3 select 문을 이용하여 특정한 테이블스페이스에 대한 정보를 확인합니다.

```
$ db2 "select * from syscat.tablespaces"
```

```
select TBSPACE, TBSPACEID, TBSPACETYPE,  DATATYPE,
       PAGESIZE, EXTENTSIZE, PREFETCHSIZE, BUFFERPOOLID

TBSPACE     ID     TYPE DATATYPE PAGE   EXTENT PREFETCHSIZE BUFFERPOOL
----------- ------ ---- -------- ------ ------ ------------ ----------
SYSCATSPACE      0 S    A           4096     32           64          1
TEMPSPACE1       1 S    T           4096     32           64          1
USERSPACE1       2 S    A           4096     32           64          1
TS5              3 S    A           4096     32           -1          1
TS6              4 S    A           4096     32           -1          1
TS1              5 D    A           4096     32           -1          1
TS2              6 S    A           4096     32           64          1

  7 레코드가 선택됨.
```

Figure 0720A syscat.tablespaces 뷰

UNIT 08

데이터베이스 오브젝트

한 데이터베이스에는 테이블, 뷰, 인덱스, 트리거, 시퀀스 등의 다양한 오브젝트들을 생성할 수 있습니다. 버퍼풀과 테이블스페이스를 제외한 여러 가지 데이터베이스 오브젝트에 대한 소개와 CREATE 문, ALTER 문, DROP 문을 이용한 생성, 변경, 제거 방법을 설명합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 데이터베이스 파티션 그룹
- 스키마
- 스키마 지정 방법
- 테이블
- CREATE TABLE 문
- ALTER TABLE 문
- 데이터 유형
- NULL 값과 DEFAULT 값
- 테이블스페이스 지정
- 고유키
- 기본키
- 외부키
- 참조 무결성
- 점검 제한 조건
- IDENTITY 컬럼
- NOT LOGGED INITIALLY 옵션
- 뷰
- CREATE VIEW 문
- MQT
- 인덱스
- CREATE INDEX문
- 시퀀스
- 트리거
- CREATE TRIGGER
- AFTER 트리거
- BEFORE 트리거
- INSTEAD OF 트리거
- 사용자 정의 유형
- 사용자 정의 함수
- CREATE FUNCTION 문
- SQL 사용자 정의 함수
- 저장 프로시저
- CREATE PROCEDURE 문
- SQL/PL 저장 프로시저
- PL/SQL 저장 프로시저

TOPIC 08 01 데이터베이스 파티션 그룹

Point

DPF를 이용하여 다중 데이터베이스 파티션을 구성하면 데이터베이스 파티션의 묶음인 데이터베이스 파티션 그룹을 정의할 수 있습니다. CREATE DATABASE PARTITION GROUP, DROP DATABASE PARTITION GROUP 문으로 관리합니다.

1 데이터베이스를 생성하면 3개의 데이터베이스 파티션 그룹이 기본적으로 생성됩니다.

파티션그룹	설명
IBMCATGROUP	카탈로그 테이블스페이스가 생성되는 파티션입니다.
IBMDEFAULTTEMPGROUP	시스템 임시 테이블스페이스가 생성되는 파티션입니다.
IBMDEFAULTGROUP	사용자 테이블스페이스가 생성되는 기본 파티션입니다.

2 create database partition group 문의 형식은 다음과 같습니다.

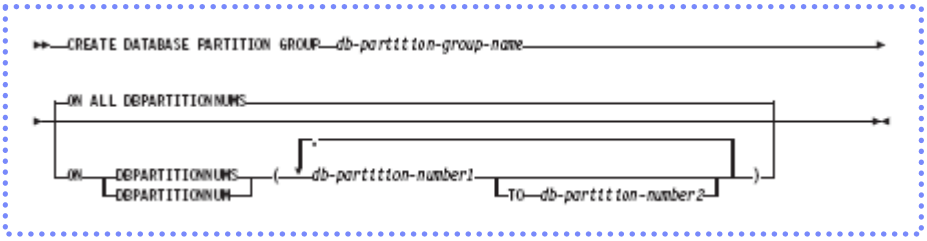


Figure 0801A CREATE DATABASE PARTITION GROUP 문

옵션	설명
<DB 파티션 그룹명>	임의의 고유한 이름으로 지정합니다.
ON ALL DBPARTITIONNUMS	모든 파티션에 생성되도록 합니다.
ON DBPARTITIONNUMS	한 개 이상의 지정한 파티션에 생성되도록 합니다.
<파티션 번호>	db2nodes.cfg 파일에 정의된 파티션 번호입니다.

Tip

SYSADM, SYSCTRL 권한이 필요합니다.

3 create database partition group 명령어에서 한 개 이상의 데이터베이스 파티션 번호를 이용하여 새로운 데이터베이스 파티션 그룹을 생성합니다.

```
$ db2 "create database partition group <DB 파티션 그룹명>  on
    dbpartitionnum (<파티션 번호 1>, <파티션 번호 2>)"
```

4 drop database partition group 명령어로 데이터베이스 파티션을 제거합니다.

```
$ db2 "drop database partition group <DB 파티션 그룹명>"
$ db2 "drop nodegroup <DB 파티션 그룹명>"
```

Tip

IBMDEFAULTTEMPGROUP 은 list nodegroups 명령어로 표시되지 않습니다.

5 list database partition groups 명령어를 이용하여 정의된 데이터베이스 파티션 그룹의 정보를 확인합니다. list nodegroups 명령어를 사용해도 됩니다.

```
$ db2 list database partition groups  show detail
```

6 SYSCAT.NODEGROUPS 뷰에서 연관된 정보를 확인합니다.

```
$ db2 list database partition groups   show detail
```

TOPIC 08 02 스키마

Point

데이터베이스 오브젝트의 이름은 <스키마명>.<오브젝트명> 과 같이 2-part 형식으로 구성됩니다. 스키마는 오브젝트의 이름을 수식하는 수식자 역할을 합니다. CREATE SCHEMA, DROP SCHEMA 문으로 관리합니다.

1 데이터베이스를 생성하면 다음과 같이 4가지의 스키마가 기본적으로 생성됩니다.

스키마	설명
SYSIBM	시스템 카탈로그 테이블의 스키마입니다.
SYSCAT	시스템 카탈로그 뷰의 스키마입니다.
SYSSTAT	통계 자료와 관련된 시스템 카탈로그 뷰의 스키마입니다.
SYSFUN	기본적으로 제공되는 사용자 정의 함수의 스키마입니다.

2 create schema 문의 형식은 다음과 같습니다.

옵션	설명
<스키마명>	임의의 고유한 이름으로 지정합니다.
AUTHORIZATION	스키마의 소유자를 지정합니다.

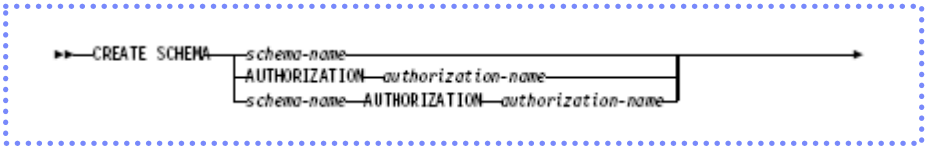


Figure 0802A CREATE SCHEMA 문

3 create schema 문을 이용하여 새로운 스키마를 생성합니다.

```
$ db2 "create schema <스키마명>"
```

4 create schema 문에서 AUTHORIZATION 옵션을 이용하여 스키마의 소유자를 지정할 수 있습니다.

```
$ db2 "create schema <스키마명> AUTHORIZATION <스키마의 소유자명>"
```

5 drop schema 문을 이용하여 기존의 스키마를 제거합니다. 반드시 RESTICT 옵션을 지정하도록 합니다.

```
$ db2 "drop schema <스키마명> restrict"
```

6 list tables 명령어에서 FOR SCHEMA 옵션을 이용하면 동일한 스키마를 가지는 테이블과 뷰의 목록을 확인할 수 있습니다.

```
$ db2 list tables for schema <스키마명>
```

7 생성된 스키마에 대한 정보는 SYSCAT.SCHEMA 뷰를 이용해서 확인합니다.

```
$ db2 "select * from syscat.schemata"
```

Tip
SYSADM, DBADM 권한이 필요합니다.

Tip
SYS로 시작되는 스키마명은 사용하지 않도록 합니다.

Tip
새로 생성된 스키마를 이용하려는 사용자는 스키마의 소유자로부터 CREATEIN, ALTERIN, DROPIN 등의 특권을 부여받아야 합니다.

Tip
제거하려는 스키마명을 가진 오브젝트가 존재하면, SQL0478N 오류 코드가 반환되고, 스키마는 제거되지 않습니다.

TOPIC 08 03 스키마 지정 방법

Point

데이블 등의 데이터베이스 오브젝트의 이름을 명시할 때는 <스키마명>.<오브젝트명> 형식의 2-part name 을 사용하는 것이 원칙입니다. <스키마명>을 명시적으로 지정하지 않으면 접속 사용자명이 스키마 명입니다.

1. SQL문에서 <스키마명> 없이 <테이블명>만 지정하면, <현재 세션의 로그온 사용자명>이 기본 <스키마명>으로 인식됩니다. <테이블명>은 <사용자명1>.<테이블명>으로 인식됩니다.

```
$ login <사용자명1>
$ db2 connect to <데이터베이스명>
$ db2 "select * from <테이블명>"
```

2. 데이터베이스에 접속하는 connect 문에서 USER 와 USING 옵션을 이용하면, <현 세션의 로그온 사용자명>에 관계 없이 <데이터베이스 접속 시에 사용된 사용자명>이 기본 <스키마명>으로 인식됩니다. <테이블명>은 <사용자명2>.<테이블명>으로 인식됩니다.

```
$ login <사용자명1>
$ db2 connect to <데이터베이스명> user <사용자명2> using <암호명2>
$ db2 "select * from <테이블명>"
```

Tip

set current schema 문은 데이터베이스에 접속한 상태에서실행할 수 있으며, 접속이 해제되면 <로그온 사용자명>으로 복원됩니다.

3. CURRENT SCHEMA 특수 레지스터리 변수는 스키마명을 명시적으로 지정하지 않는 경우에 기본 스키마로 적용될 값을 저장하고 있습니다. values 문으로 현재값을 확인할 수 있습니다. set current schema 문으로 CURRENT SCHEMA 특수 레지스터리 변수를 변경하면, <데이터베이스 접속시 사용된 사용자명> 보다 우선적으로 적용됩니다. <테이블명>은 <스키마명 1>.<테이블명>으로 인식됩니다.

```
$ login <사용자명1>
$ db2 connect to <데이터베이스명> user <사용자명2> using <암호명2>
$ db2 values(current schema)
$ db2 set current schema <스키마명1>
$ db2 values(current schema)
$ db2 "select * from <테이블명>"
```

4. 데이터베이스의 오브젝트를 지정할 때는 개별적인 SQL문에서 <스키마명>을 명시적으로 지정하는 것이 권장됩니다. <현재 세션의 로그온 사용자명>, <데이터베이스 접속시에 사용된 사용자명>, <CURRENT SCHEMA 특수 레지스터리 변수의 현재값> 보다 SQL문에서 명시적으로 지정한 <스키마명>이 가장 우선적으로 적용됩니다. <테이블명>은 <스키마명2>.<테이블명>로 인식됩니다.

```
$ login <사용자명1>
$ db2 connect to <데이터베이스명> user <사용자명2> using <암호명2>
$ db2 set current schema <스키마명1>
$ db2 "select * from <스키마명2>.<테이블명>"
```

TOPIC 0804 테이블

Point

사용자의 데이터는 테이블에 저장됩니다. 테이블의 데이터는 기본적으로 한 개의 테이블스페이스에 저장됩니다. CREATE TABLE, ALTER TABLE, DROP TABLE 문으로 관리합니다.

Tip

테이블스페이스의 페이지 크기 별 제약 사항은 정보 센터를 참조합니다.

1 한 테이블스페이스에는 한 개 이상의 테이블이 저장됩니다.

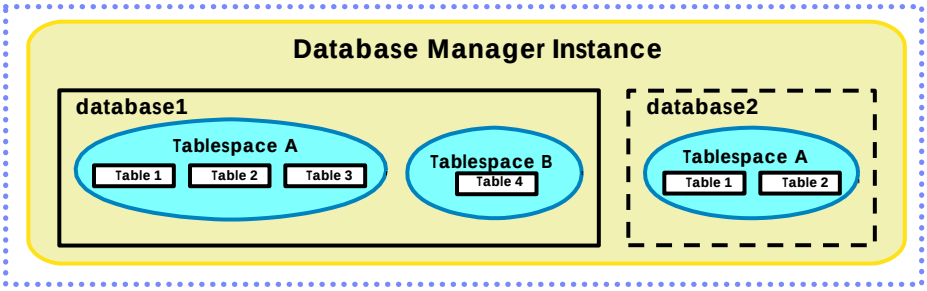


Figure 0804A ••• 테이블스페이스와 테이블

Tip

테이블의 한 행의 총 길이는 지정한 테이블 스페이스의 페이지 크기보다 작아야 합니다.

2 create table 문을 이용하여 테이블을 정의합니다. IMPLICIT_SCHEMA 특권이 있으면, 존재하지 않는 <스키마명>을 이용하여 테이블을 정의할 수 있습니다.

```
$ db2 "create table <스키마명>.<테이블명> (<컬럼정의>)"
```

3 alter table 문을 이용하여 컬럼 추가, 고유키 추가 및 제거, 기본키 추가 및 제거, 외부키 추가 및 제거, 점검 제한 조건 추가 및 제거 등의 변경 작업이 가능합니다.

```
$ db2 "alter table <스키마명>.<테이블명> ADD <제한조건>"
```

4 drop table문으로 테이블을 제거합니다.

```
$ db2 "drop table <스키마명>.<테이블명>"
```

5 list tables 명령어에서 테이블의 목록을 확인할 수 있습니다.

```
$ db2 list tables
$ db2 list tables for schema <스키마명>
$ db2 list tables for system
$ db2 list tables for all
```

6 테이블에 대한 정보는 SYSCAT.TABLES 뷰를 이용해서 확인합니다.

```
$ db2 "select * from syscat.tables"
```

7 describe table 문으로 테이블의 컬럼에 대한 정보를 확인합니다.

```
$ db2 "describe table <스키마명>.<테이블명>"
```

8 db2look 명령어로 테이블에 대한 DDL문을 추출할 수 있습니다.

```
$ db2look –d <DB명> –e –z <스키마명> –t <테이블명> -o <출력파일명>
```

TOPIC 08 05 CREATE TABLE 문

Point

테이블을 생성할 때 이용하는 SQL문입니다. 컬럼, 고유키, 기본키, 외부키, 점검 제한 조건 등을 정의하고, 데이터와 인덱스를 저장할 테이블스페이스를 지정합니다.

Tip

SYSADM, DBADM 권한 또는 데이터베이스에 대한 CREATAB 특권이 필요할 수 있습니다.

Tip

새로운 스키마명을 이용하여 테이블명을 지정하려면, 데이터베이스에 대한 IMPLICIT_SCHEMA 특권이 필요합니다.

Tip

기존의 스키마명을 이용하여 테이블명을 지정하려면, 스키마에 대한 CREATIN 특권이 필요합니다.

Tip

IN, INDEX IN, LONG IN 옵션을 사용하려면 테이블스페이스에 대한 USE 특권이 필요합니다.

Tip

외부키를 정의하려면, 부모 테이블에 대한 REFERENCES 특권이 필요합니다.

Tip

NOT LOGGED INITIALLY 특성은 ALTER TABLE 문으로 활성화할 때만 적용됩니다.

1 create table 문의 형식은 다음과 같습니다.

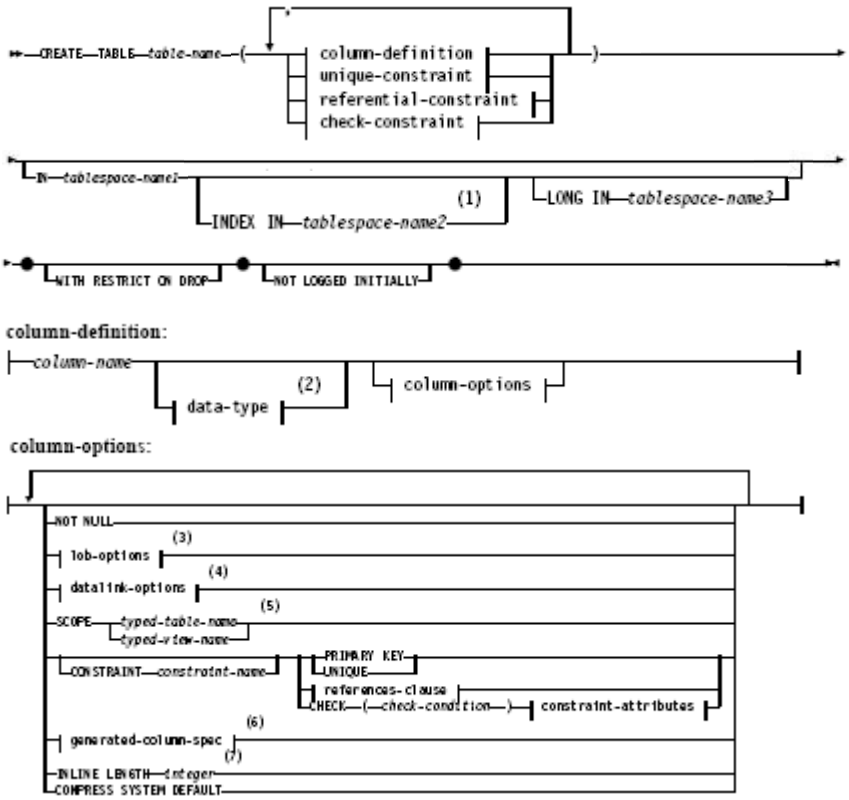


Figure 0804A ••• CREATE TABLE 문

2 옵션에 대한 설명은 다음과 같습니다.

옵션	설명
<테이블명>	임의의 고유한 이름으로 지정합니다. <스키마명>을 생략하면 CURRENT SCHEMA 특수 레지스터리 변수의 값이 기본 스키마명으로 사용됩니다. 기존의 스키마명을 사용하려면,
<컬럼명>	테이블 내에서 고유한 임의의 이름으로 지정합니다.
<데이터 유형>	컬럼의 데이터 유형을 지정합니다.
<NULL 허용 여부>	NULL값을 허용하지 않으려면 'NOT NULL' 로 지정합니다.
<기본값>	WITH DEFAULT 옵션으로 기본값을 지정합니다.
IN <TS명>	테이블의 데이터가 저장될 테이블스페이스명을 지정합니다.
INDEX IN <TS명>	인덱스 데이터가 저장될 테이블스페이스명을 지정합니다.
LONG IN <TS명>	LONG 데이터가 저장될 테이블스페이스명을 지정합니다.
NOT LOGGED INITIALLY	트랜잭션에서 해당 테이블에 대한 변경 사항을 로그에 기록하지 않게 합니다.

TOPIC 08 06 ALTER TABLE 문

Point

테이블의 특성을 변경할 때 사용하는 SQL문입니다. 컬럼 추가, 고유키 추가 및 제거, 기본키 추가 및 제거, 외부키 추가 및 제거, 점검 제한 조건 추가 및 제거 등의 변경 작업이 가능합니다.

Tip

SYSADM, DBADM 권한 또는 테이블에 대한 CONTROL, ALTER 특권이 필요할 수 있습니다.

Tip

기존의 스키마명을 이용하여 테이블을 변경하려면, 스키마에 대한 ALTERIN 특권이 필요합니다.

Tip

외부키를 정의하려면, 부모 테이블에 대한 REFERENCES 특권이 필요합니다.

1 alter table 문의 형식은 다음과 같습니다.

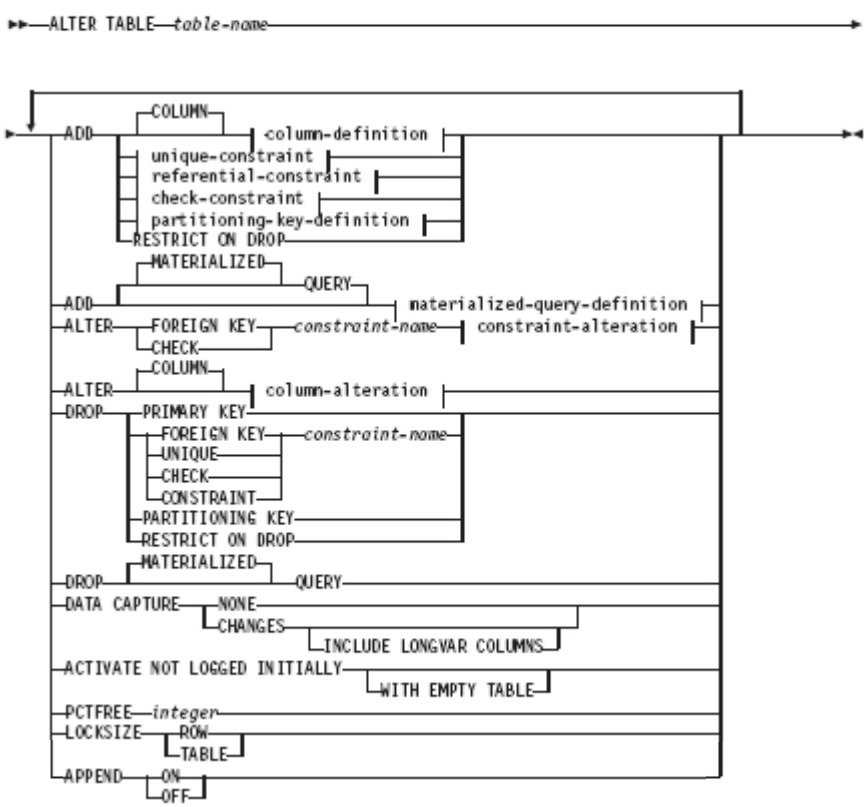


Figure 0805A ··· ALTER TABLE 문

2 옵션에 대한 설명은 다음과 같습니다.

옵 션	설 명
<테이블명>	변경할 테이블명을 지정합니다.
ADD <컬럼정의>	컬럼을 추가합니다.
ADD <고유키제한조건>	고유키를 추가합니다.
ADD <기본키제한조건>	기본키를 추가합니다.
ADD <외부키제한조건>	외부키를 추가합니다.
ADD <점검제한조건>	점검 제한 조건을 추가합니다.
ALTER <컬럼명>	VARCHAR 유형에서 컬럼의 길이를 증가시킬 수 있습니다.
DROP <제한조건명>	지정된 <제한조건>을 제거합니다.
ACTIVATE NOT LOGGED INITIALLY	NOT LOGGED 모드로 전환합니다. 트랜잭션이 종료될 때까지는 해당 테이블에 대한 변경 작업이 로깅되지 않습니다.

TOPIC 0807 데이터 유형

Point

기본적으로 지원되는 컬럼의 데이터 유형은 다음과 같습니다. CREATE DISTINCT TYPE 문으로 사용자가 새로운 데이터 유형을 추가로 생성하여 사용할 수도 있습니다.

1 기본적으로 지원되는 데이터의 유형은 다음과 같습니다.

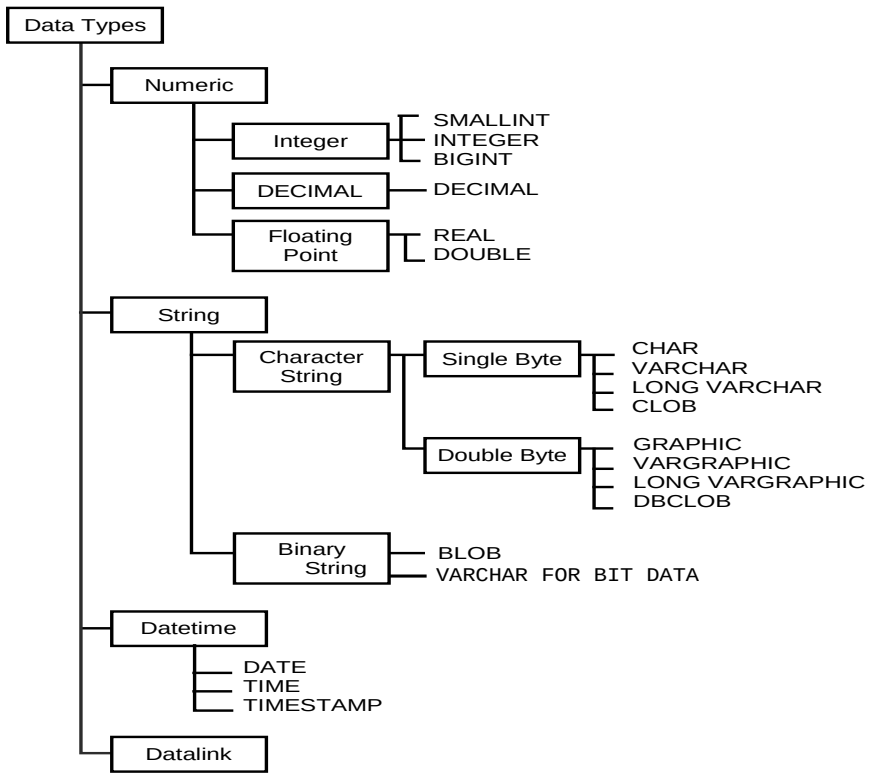


Figure 0807A ··· 데이터 유형

2 대표적인 데이터 유형에 대한 설명은 다음과 같습니다.

구 분	유 형	저장 BYTE 수	최 대 범 위
숫 자	SMALLINT	2	-32,768 ~ +-32,767
	INT	4	-2,147,483,648 ~ +2,147,483,647
	BIGINT	8	-9,223,372,036,854,775,808 ~ +9,223,372,036,854,775,807
	DEC(p,s)	(p+s)/2+1	31 자리
	DOUBLE	8	-1.79769E+308 ~ +1.79769E+308
문 자	CHAR(n)	n	254 바이트
	VARCHAR(n)	n + 4	32672 바이트 (32K 페이지인 경우)
날 짜	DATE	10	0001-01-01 ~ 9999-12-31
	TIME	8	00:00:00 ~ 24:00:00
	TIMESTAMP	26	0001-01-01-00.00.00.000000 ~ 9999-12-31-24.00.00.000000

Tip

VARCHAR의 최대 길이는 페이지 크기에 따라 다릅니다.

Tip

LOB 유형의 최대 길이는 2G입니다 .

TOPIC 0808 NULL 값과 DEFAULT 값

Point

컬럼에 NULL 값과 DEFAULT 값을 허용하게 할 수 있습니다. DEFUALT 속성을 가지지 않는 컬럼은 반드시 명시적으로 값을 지정해야 합니다. CREATE TABLE 문에서 NOT NULL 옵션과 WITH DEFULAT 옵션을 이용합니다.

Tip

NULL 값은 0 또는 공백 (blank) 또는 empty string 이 아닙니다. empty string은 길이가 0인 값을 의미하며, 공백 문자와는 다릅니다.

1 NULL 값은 알려지지 않은 값을 의미합니다. 테이블을 정의할 때, 컬럼에 NULL 값을 허용하지 않으려면 CREATE TABLE 문에서 NOT NULL 옵션을 이용합니다.

```
$ db2 "create table <테이블명> ( <컬럼명>  <데이터유형명> NOT NULL, ....)"
```

2 CREATE TABLE 문에서 WITH DEFAULT 옵션만 지정하면 시스템 기본값이 제공됩니다.

```
$ db2 "create table <테이블명> ( <컬럼명>  <데이터유형명> WITH DEFAULT)"
```

유 형	기본값	설 명
숫자	0	0
문자	EMPTY STRING	길이가 0 인 문자
DATE	CURRENT DATE	현재 시스템 날짜
TIME	CURRENT TIME	현재 시스템 시간
TIMESTAMP	CURRENT TIMESTAMP	현재 시스템 시간 소인

Tip

날짜 유형의 표현식은 데이터베이스의 코드 페이지에 따라 달라집니다. 코드 페이지가 970(ko_KR)인 경우에 DATE 유형은 'yyyy-mm-dd', TIME 유형은 'hh:mm:ss' 가 되고, TIMESTAMP 유형은 'yyyy-mm-dd-hh.mm.ss.uuuuuu' 으로 표현됩니다.

3 CREATE TABLE 문에서 WITH DEFAULT <기본값> 옵션을 지정하면 사용자가 지정한 값이 기본값으로 사용됩니다.

```
$ db2 "create table <테이블명> ( <컬럼명>  <데이터유형명> WITH DEFAULT
    <기본값>)"
```

유 형	기 본 값	예
숫자	WITH DEFAULT <숫자>	WITH DEFAULT 10
문자	WITH DEFAULT '<문자열>'	WITH DEFAULT 'xx'
DATE	WITH DEFAULT '<날짜>'	WITH DEFAULT '2006-04-17'
TIME	WITH DEFAULT '<시간>'	WITH DEFAULT '14:12:30'
TIMESTAMP	WITH DEFAULT '<시간소인>'	WITH DEFAULT '2006-04-17-14.12.30.694001'

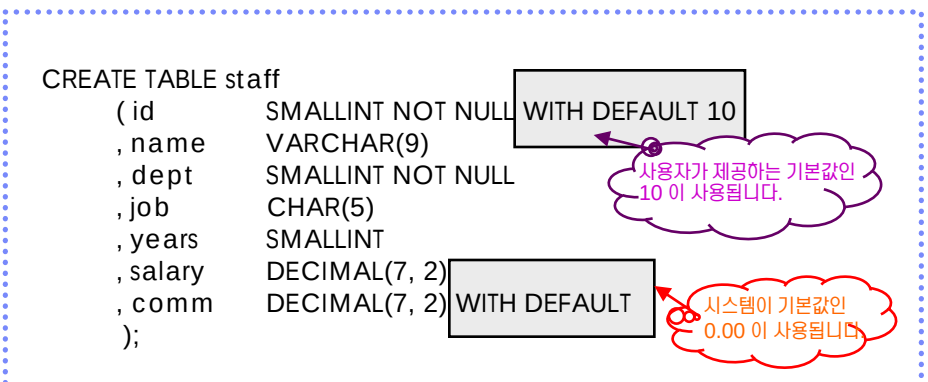


Figure 0808A • • • 기본값 지정

TOPIC 08 09 테이블스페이스 지정

Point

CREATE TABLE 문에서 IN 키워드를 이용하여 테이블스페이스를 지정할 수 있습니다. INDEX IN, LONG IN 키워드로 테이블, 인덱스, LONG 데이터를 개별적인 DMS 테이블스페이스에 저장할 수 있습니다. 지정한 테이블스페이스는 변경될 수 없습니다.

Tip

해당 테이블의 행의 총 길이를 수용할 수 있는 적합한 기본 테이블스페이스가 없다면, SQL0286N 오류가 반환됩니다.

Tip

특정한 테이블 스페이스를 지정하려면, 해당 테이블 스페이스에 대한 USE 특권이 있어야 합니다.

Tip

사용자가 정의한 테이블스페이스 중에서 해당 테이블의 행의 총 길이를 수용할 수 있는 페이지 크기를 가진 첫 번째 REGULAR 유형의 테이블스페이스가 기본 사용자 테이블스페이스로 사용됩니다.

Tip

사용자가 정의한 테이블스페이스가 없다면, USERSPACE1이 기본 사용자 테이블스페이스입니다.

Tip

한 테이블이 여러 테이블스페이스에 저장되었다면, 테이블스페이스는 함께 drop 되어야 합니다.

1. CREATE TABLE 문에서 IN 옵션을 지정하지 않으면, 테이블은 기본 사용자 테이블스페이스에 저장됩니다.

```
$ db2 "create table <스키마명>.<테이블명> (<컬럼 정의>) "
```

2. CREATE TABLE 문에서 IN 옵션으로 테이블이 저장될 테이블스페이스를 지정합니다. 테이블의 모든 데이터와 인덱스 데이터는 동일한 테이블스페이스에 저장됩니다.

```
$ db2 "create table <스키마명>.<테이블명> (<컬럼 정의>) IN <테이블스페이스명>"
```

3. CREATE TABLE 문에서 INDEX IN 키워드를 이용하여 인덱스를 위한 데이터를 별도의 테이블스페이스에 저장할 수 있습니다. IN 옵션과 INDEX IN 옵션에서 지정한 테이블스페이스는 DMS 방식의 REGULAR 유형이어야 합니다. INDEX IN 옵션만 지정할 수는 없습니다.

```
$ db2 "create table <스키마명>.<테이블명> (<컬럼 정의>) IN <테이블스페이스명> INDEX IN <테이블스페이스명> "
```

4. CREATE TABLE 문에서 LONG IN 키워드를 이용하여 LONG 데이터를 별도의 테이블스페이스에 저장할 수 있습니다. IN 옵션에서 지정한 테이블스페이스는 DMS 방식의 REGULAR 유형이고, LONG IN 옵션에서 지정한 테이블스페이스는 DMS 방식의 LARGE 유형이어야 합니다. LONG IN 옵션만 지정할 수는 없습니다.

```
$ db2 "create table <스키마명>.<테이블명> (<컬럼 정의>) IN <테이블스페이스명> LONG IN <테이블스페이스명> "
```

5. CREATE TABLE 문에서 IN, INDEX IN, LONG 옵션을 모두 사용하여 테이블 데이터, 인덱스 데이터, LONG 데이터를 별도의 DMS 테이블스페이스를 저장할 수 있습니다.

```
$ db2 "create table <스키마명>.<테이블명> (<컬럼 정의>) IN <테이블스페이스명> INDEX IN <테이블스페이스명> LONG IN <테이블스페이스명> "
```

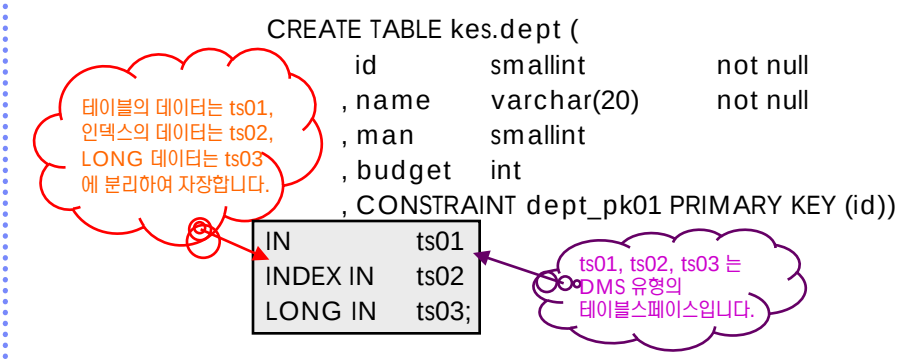


Figure 0809A ••• 테이블스페이스 지정

TOPIC 08-10 고유키

Point

고유 키는 한 개 이상의 컬럼들로 구성되어 테이블의 각 행을 고유하게 구별하는 값입니다. 한 테이블에 한 개 이상의 고유키를 지정할 수 있습니다. 고유 키를 정의하면, 해당 컬럼들로 구성된 고유 인덱스가 자동으로 생성됩니다.

Tip

고유키에 대응하는 인덱스가 이미 존재하면, SQL0000W 라는 경고 메시지가 반환되지만, 무시해도 됩니다.

Tip

<제한조건명>은 데이터베이스 내에서 고유해야 합니다.

Tip

<제한조건명>을 명시하지 않으면, 엔진이 생성한 이름으로 관리하게 됩니다. ALTER 또는 DROP문으로 관리하려면, 사용자가 제한조건명을 명시하는 것이 편리합니다.

1 고유키 제한 조건을 정의하는 구문은 다음과 같습니다. 고유키를 구성하는 각 컬럼은 NOT NULL 속성을 지정해야 합니다.

Figure 0810A ••• 고유키 제한 조건절

2 create table 문에서 CONSTRAINT ~ UNIQUE 라는 옵션으로 지정합니다. <제한조건명>과 동일한 이름을 가진 고유 인덱스가 자동으로 생성됩니다. 고유키는 한 테이블에 여러 개 정의할 수 있습니다. CONSTRAINT 옵션을 지정하지 않으면, 제한조건명과인덱스의 이름은 'SQLyymmddhhmmssxxx' 형식으로 엔진이 부여합니다.

```
$ db2 "create  table <스키마명>.<테이블명> (..., <고유키 제한 조건절>, ...)"
```

3 alter table 문을 이용하여 고유키를 추가할 수 있습니다.

```
$ db2 "alter  table <스키마명>.<테이블명> ADD <고유키 제한 조건절>"
```

4 alter table 문을 이용하여 고유키를 제거할 수 있습니다.

```
$ db2 "alter  table <스키마명>.<테이블명> DROP CONSTRAINT  <제한조건
    명>"
```

```
CREATE TABLE kes.empl (
        id          smallint        not null
      , name        varchar(30)     not null
      , sex         char(1)
      , mydept      smallint
      , salary      smallint
      , email       varchar(30)     not null
      , hiredate    date
      , CONSTRAINT empl_uk01 UNIQUE(email)
);

ALTER TABLE kes.empl DROP CONSTRAINT empl_uk01;

ALTER TABLE kes.empl ADD CONSTRAINT empl_uk01
                     UNIQUE (email) ;
```

email 컬럼은 NOT NULL로 지정합니다.

email 컬럼이 고유키가 됩니다.

제한조건명은 empl_uk01 입니다.

자동으로 생성되는 고유 인덱스의 이름도 empl_uk01 입니다.

Figure 0810B ••• 고유키 생성

TOPIC 08 11 기본키

Point

기본키는 고유 키와 동일한 특성을 갖지만, 한 테이블에 한 개만 지정할 수 있습니다. 기본키는 한 개 이상의 컬럼들로 구성될 수 있으며, 해당 컬럼들로 구성된 고유 인덱스가 자동으로 생성됩니다.

Tip

고유키와 기본키는 기능적으로 동일합니다. 단, IMPORT 명령어의 INSERT_UPDATE 옵션을 사용할 때에는 반드시 기본키가 필요합니다.

Tip

고유키에 대응하는 인덱스가 이미 존재하면, SQL0000W 라는 경고 메시지가 반환되지만, 무시해도 됩니다.

Tip

<제한조건명>은 데이터베이스 내에서 고유해야 합니다.

Tip

<제한조건명>을 명시하지 않으면, 엔진이 생성한 이름으로 관리하게 됩니다. ALTER 또는 DROP문으로 관리하려면, 사용자가 제한조건명을 명시하는 것이 편리합니다.

1 기본키 제한 조건을 정의하는 구문은 다음과 같습니다. 기본키를 구성하는 각 컬럼은 NOT NULL 속성을 지정해야 합니다.

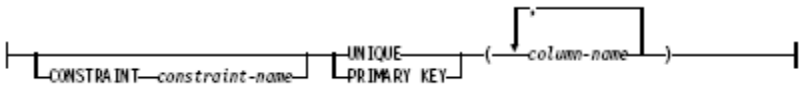


Figure 0811A ••• 기본키 제한 조건절

2 create table 문에서 CONSTRAINT ~ PRIMARY KEY 라는 옵션으로 지정합니다. <제한조건명>과 동일한 이름을 가진 고유 인덱스가 자동으로 생성됩니다. 기본키는 한 테이블에 한 개만 정의할 수 있습니다. CONSTRAINT 옵션을 지정하지 않으면, 제한조건명과 인덱스의 이름은 'SQLyymmddhhmmssxxx' 형식으로 엔진이 부여합니다.

```
$ db2 "create  table <스키마명>.<테이블명> (..., <기본키 제한 조건절>, ...)"
```

3 alter table 문을 이용하여 기본키를 추가할 수 있습니다.

```
$ db2 "alter  table <스키마명>.<테이블명> ADD <기본키 제한 조건절>"
```

4 alter table 문을 이용하여 기본키를 제거할 수 있습니다.

```
$ db2 "alter  table <스키마명>.<테이블명> DROP CONSTRAINT  <제한조건
    명>"
```

```
CREATE TABLE kes.empl (
        id          smallint        not null
      , name        varchar(30)     not null
      , sex         char(1)
      , mydept      smallint
      , salary      smallint
      , email       varchar(30)     not null
      , hiredate    date
      , CONSTRAINT empl_pk01 PRIMARY KEY (id)
);

ALTER TABLE kes.empl DROP CONSTRAINT empl_pk01 ;

ALTER TABLE kes.empl ADD CONSTRAINT empl_pk01
                     PRIMARY KEY(id) ;
```


Figure 0811B ••• 기본키 생성

TOPIC 08-12 외부키

Point

외부 키는 부모 테이블의 고유키 또는 기본키를 참조하는 키입니다. 참조하는 고유키 또는 기본키와 호환되는 컬럼들로 구성되어야 하며, 각 컬럼은 NULL 값을 허용합니다.

1 외부키 제한 조건을 정의하는 구문은 다음과 같습니다.

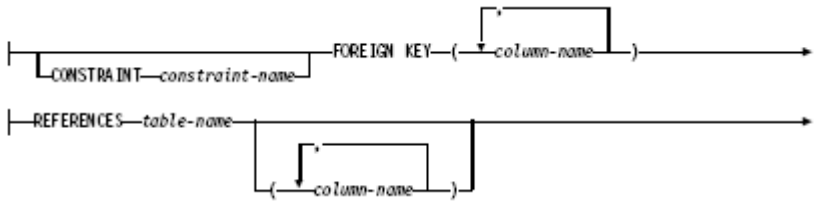


Figure 0812A ••• 외부키 제한 조건절

2 create table 문에서 CONSTRAINT ~ FOREIGN KEY 라는 옵션으로 지정합니다. 외부키는 한 테이블에 여러 개 정의할 수 있습니다. CONSTRAINT 옵션을 지정하지 않으면, 제한조건명은 'SQLyymmddhhmmssxxx' 형식으로 엔진이 부여합니다.

```
$ db2 "create  table <스키마명>.<테이블명> (..., <외부키 제한 조건절>, ...)"
```

Tip

<제한조건명>은 데이터베이스 내에서 고유해야 합니다.

3 alter table 문을 이용하여 외부키를 추가할 수 있습니다.

```
$ db2 "alter  table <스키마명>.<테이블명> ADD <외부키 제한 조건절>"
```

Tip

<제한조건명>을 명시하지 않으면, 엔진이 생성한 이름으로 관리하게 됩니다. ALTER 또는 DROP문으로 관리하려면, 사용자가 제한조건명을 명시하는 것이 편리합니다.

4 alter table 문을 이용하여 외부키를 제거할 수 있습니다.

```
$ db2 "alter  table <스키마명>.<테이블명> DROP CONSTRAINT  <제한조건명>"
```

Tip

외부키는 SQL문에서 부모 테이블의 기본키 또는 고유키와 JOIN 하게 되므로, 외부키에 대한 인덱스를 생성하는 것이 좋습니다.

```
CREATE TABLE kes.empl (
        id          smallint        not null
      , name        varchar(30)
      , sex         char(1)
      , mydept      smallint
      , salary      smallint
      , email       varchar(30)     not null
      , hiredate    date
      , CONSTRAINT empl_fk01 FOREIGN KEY(mydept)
        REFERENCES kes.dept
);

ALTER TABLE kes.empl DROP CONSTRAINT empl_fk01

ALTER TABLE kes.empl
        ADD CONSTRAINT empl_fk01 FOREIGN KEY(mydept)
        REFERENCES kes.dept ;
```

mydept 컬럼은 NULL 값을 허용할 수 있습니다.

mydept 컬럼이 외부키가 됩니다.

부모 테이블의 이름은 kes.dept 입니다.

제한조건명은 empl_fk01 입니다.

Figure 0812B ••• 외부키 생성

TOPIC 08-13 참조 무결성

Point

두 테이블이 고유키와 외부키로 연결되어 외부키를 가진 테이블에 데이터를 추가, 변경하는 경우에 데이터의 참조 무결성이 유지됩니다. 고유키를 가진 테이블에 데이터를 변경, 제거하는 경우에는 UPDATE 규칙과 DELETE 규칙이 적용됩니다.

Tip

부모 테이블에 INSERT문을 실행할 때, 점검 규칙은 필요하지 않습니다.

Tip

외부키는 자신의 테이블에 있는 고유키를 참조할 수도 있습니다.

Tip

동일한 부모 테이블의 고유키를 여러 자손 테이블의 외부키가 참조할 수 있습니다.

Tip

한 개의 DELETE 문 또는 UPDATE 문을 실행할 때, 두 개 이상의 RI가 존재하는 경우는 흔하지 않으므로, 보통은 NO ACTION과 RESTRICT 를 동일하게 생각하면 됩니다.

1 외부키 제한 조건에서 UPDATE 규칙과 DELETE 규칙을 정의하는 옵션은 다음과 같습니다.

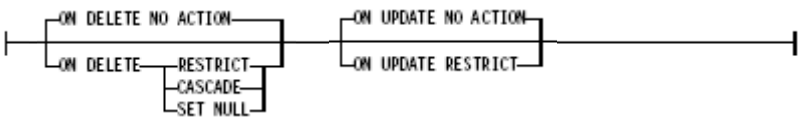


Figure 0813A • • • UPDATE 규칙과 DELETE 규칙 옵션

2 외부키를 가진 자손 테이블에 INSERT 문으로 데이터를 추가할 때, 제공된 외부키가 부모 테이블의 고유키에 존재하는 값인지 점검합니다. 존재하지 않는 값인 경우에는 SQL0530N 오류 코드가 반환되고, INSERT 문은 실패합니다. 자손 테이블에 외부키에 입력된 데이터는 부모 테이블의 고유키에 존재하는 값이므로 항상 참조가 가능합니다. 이러한 기능을 '참조 무결성 (RI, Refrential Integrity)' 라고 합니다.

3 고유키를 가진 부모 테이블에서 UPDATE 문을 실행할 때는 다음과 같이 2 가지의 UPDATE 규칙을 적용받게 할 수 있습니다. CREATE TABLE 문에서 외부키를 정의할 때 ON UPDATE 옵션을 이용하여 지정합니다.

UPDATE 규칙	설 명
NO ACTION	다른 RI 관계의 UPDATE 규칙을 먼저 적용하고, 자신의 UPDATE 규칙을 적용합니다. 변경되는 행의 고유키 값을 참조하고 있는 자손 테이블의 행이 존재하면, SQL0531N 오류 코드가 반환되고, UPDATE 문이 실패합니다. 기본값으로 사용됩니다.
RESTICT	다른 RI 관계의 UPDATE 규칙보다 자신의 UPDATE 규칙을 먼저 적용하는 점을 제외하고, NO ACTION과 동일합니다.

4 고유키를 가진 부모 테이블에서 DELETE 문을 실행할 때는 다음과 같이 4 가지의 DELETE 규칙을 적용받게 할 수 있습니다. CREATE TABLE 문에서 외부키를 정의할 때 ON DELETE 옵션을 이용하여 지정합니다.

DELETE 규칙	설 명
NO ACTION	다른 RI 관계의 DELETE 규칙을 먼저 적용하고, 자신의 DELETE 규칙을 적용합니다. 삭제되는 행의 고유키 값을 참조하고 있는 자손 테이블의 행이 존재하면, SQL0532N 오류 코드가 반환되고, DELETE 문이 실패합니다. 기본값으로 사용됩니다.
RESTICT	다른 RI 관계의 DELETE 규칙보다 자신의 DELETE 규칙을 먼저 적용하는 점을 제외하고, NO ACTION과 동일합니다.
CASCADE	삭제되는 행의 고유키 값을 참조하고 있는 자손 테이블의 행을 함께 삭제합니다.
SET NULL	삭제되는 행의 고유키 값을 참조하고 있는 자손 테이블의 행의 외부키의 값을 NULL 값으로 변경합니다. 자손 테이블의 외부키 컬럼이 NULL 을 허용하는 컬럼일 때 지정할 수 있습니다.

TOPIC 08-14 점검 제한 조건

Point

테이블의 컬럼에 입력되는 값을 제한하는 조건입니다. INSERT와 UPDATE 문을 실행하면 자동으로 점검이 실행되어 조건에 맞지 않는 값인 경우에는 오류를 반환합니다.

Tip

<제한조건명>은 데이터베이스 내에서 고유해야 합니다.

Tip

<제한조건명>을 명시하지 않으면, 엔진이 생성한 이름으로 관리하게 됩니다. ALTER 또는 DROP문으로 관리하려면, 사용자가 제한조건명을 명시하는 것이 편리합니다.

Tip

점검 제한 조건을 표현하는 방법은 SQL 문의 WHERE 조건절의 표현식과 동일합니다.

1 점검 제한 조건을 정의하는 구문은 다음과 같습니다.

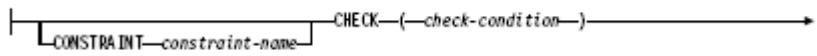


Figure 0814A ••• 점검 제한 조건절

2 create table 문에서 CONSTRAINT ~ CHECK 라는 옵션으로 지정합니다. 점검 제한 조건은 한 테이블에 여러 개 정의할 수 있습니다. CONSTRAINT 옵션을 지정하지 않으면, 제한조건명은 'SQLyymmddhhmmssxxx' 형식으로 엔진이 부여합니다.

```
$ db2 "create  table <스키마명>.<테이블명> (..., <점검 제한 조건절>, ...)"
```

3 alter table 문을 이용하여 점검 제한 조건을 추가할 수 있습니다.

```
$ db2 "alter  table <스키마명>.<테이블명> ADD <점검 제한 조건절>"
```

4 alter table 문을 이용하여 점검 제한 조건을 제거할 수 있습니다.

```
$ db2 "alter  table <스키마명>.<테이블명> DROP CONSTRAINT  <제한조건
    명>"
```

```
CREATE TABLE kes.empl (
        id          smallint          not null
      , name        varchar(30)       not null
      , sex         char(1)
      , mydept      smallint
      , salary      smallint
      , email       varchar(30)       not null
      , hiredate    date
      , CONSTRAINT empl_cc01 CHECK (sex in ('M','F'))
);

ALTER TABLE kes.empl DROP CONSTRAINT empl_cc01 ;

ALTER TABLE kes.empl ADD CONSTRAINT empl_cc01
                     CHECK (sex = 'M' or sex ='F') ;
```

sex 컬럼은 NULL값을 허용할 수 있습니다.

제한조건명은 empl_cc01 입니다.

sex 컬럼에는 'M' 또는 'F' 만 허용됩니다.

표현식은 SQL문의 WHERE 절과 동일합니다.

Figure 0814B ••• 외부키 생성

TOPIC 08-15 Identity 컬럼

Point

INSERT 문이 실행되면 자동으로 그 값이 증가되는 컬럼입니다. 컬럼의 데이터 유형은 숫자 유형으로 한 테이블에 한 개만 정의할 수 있습니다. CREATE TABLE 문에서 GENERATED 라는 옵션으로 지정할 수 있습니다.

1 create table 문에서 GENERATED 라는 옵션으로 지정합니다.

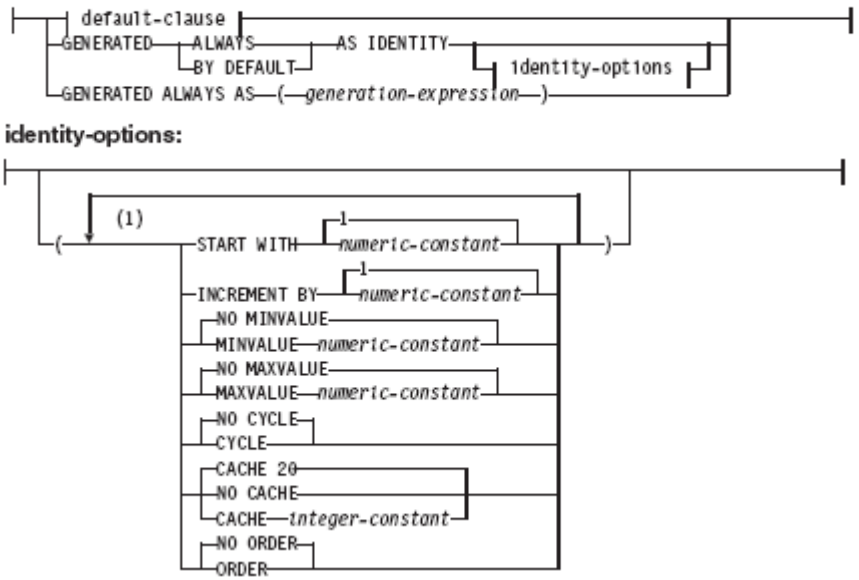


Figure 0815A • • • INDENTITY 컬럼 정의

2 주요한 옵션은 다음과 같습니다.

옵션	설명
GENERATED ALWAYS	엔진에 의해 자동으로 생성된 값만 허용됩니다. 사용자가 명시적으로 값을 지정하여 입력할 수 없습니다.
GENERATED BY DEFAULT	사용자가 값을 지정하지 않는 경우에만 엔진이 자동으로 값을 생성합니다. 값의 고유성을 보장할 수 없습니다.
START WITH	양수 또는 음수의 시작값을 지정합니다.
INCREMENT BY	양수 또는 음수의 증가값을 지정합니다.
MAXVALUE	양수 또는 음수의 최대값을 지정합니다.
CYCLE	최대값 또는 최소값에 도달하면 최소값 또는 최대값을 생성합니다.
CACHE	지정된 개수의 생성값을 미리 캐쉬에 보관하여 성능에 유리합니다.

Tip

데이터베이스가 비활성화되면, 사용되지 않고 캐쉬에 남아있던 자동 생성 값은 유실됩니다.

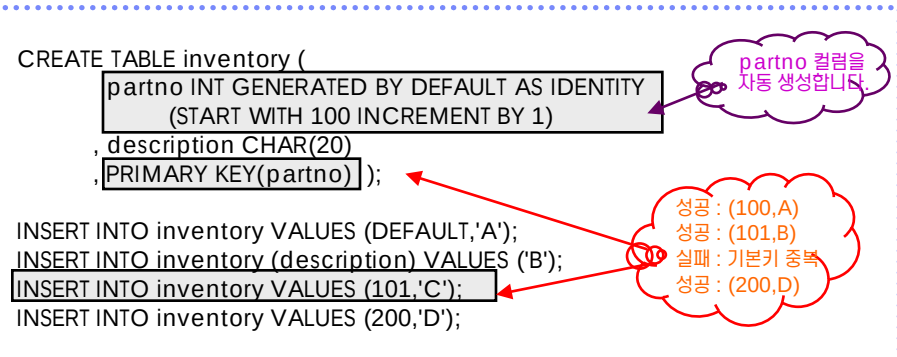


Figure 0815B • • • GENEARTED BY DEFAULT 로 생성된 IDENTITY 컬럼

TOPIC 08 16 NOT LOGGED INITIALLY 옵션

Point

필요시에 ALTER TABLE 문을 이용하여 NOT LOGGED 모드를 활성화시키면, 데이터베이스 로깅 없이 SQL문을 실행할 수 있으므로 대량의 데이터를 입력하는 경우에 유리합니다. UOW가 실패하면, 테이블은 재생성되어야 합니다.

Tip

NOT LOGGED 모드로 테이블에 입력된 데이터는 롤포워드 복구시에 rollforward db 명령어로 복구될 수 없습니다.

1 create table 문에서 NOT LOGGED INITIALLY 라는 옵션으로 지정해야 합니다.

```
$ db2 "create table <테이블명> (<컬럼 정의>) NOT LOOGED INITIALLY"
```

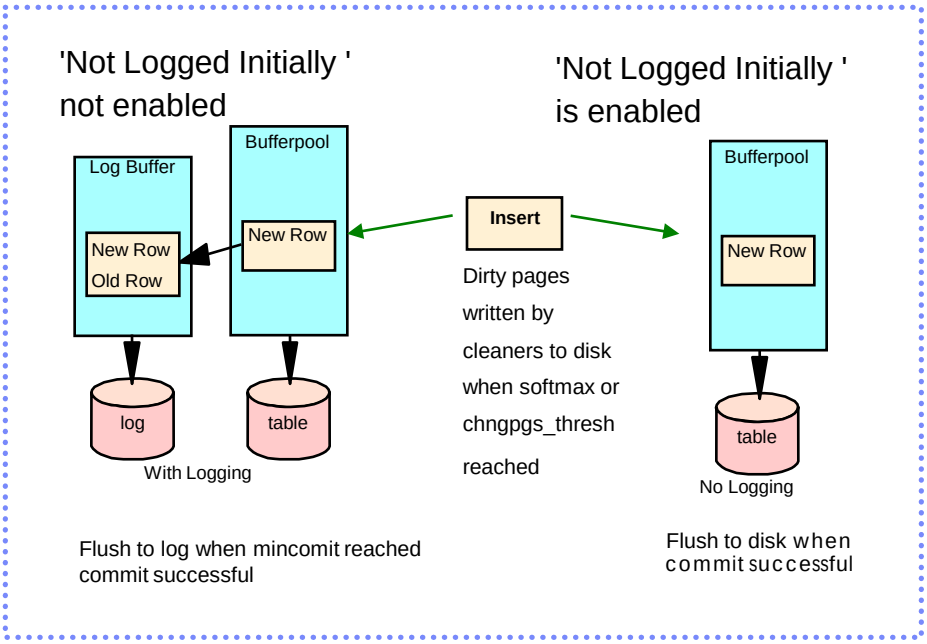


Figure 0816A • • • NOT LOGGED 모드와 LOGGED 모드

Tip

create table 문을 실행하고 commit 하면 'Not Logged Initially' 상태가 해제됩니다.

2 NOT LOGGED INITIALLY 옵션은 기본적으로 비활성화 상태이므로 테이블의 변경 내역은 데이터베이스 로그 파일에 기록됩니다. alter table 문으로 NOT LOGGED INITIALLY 옵션을 활성화시키면, COMMIT 또는 ROLLBACK 문이 실행될 때까지 해당 테이블에 대한 데이터베이스 로깅이 중지됩니다.

Tip

ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE 옵션을 이용하면 테이블의 기존 데이터를 로깅하지 않고 삭제합니다.

```
$ vi <입력파일명>
ALTER TABLE <테이블명> ACTIVATE NOT LOGGED INITIALLY;
INSERT INTO ... SELECT FROM ... ;
COMMIT;
$ db2 +c –svtf <입력파일명>
```

3 NOT LOGGED 옵션은 UOW가 성공적으로 종료되면 자동으로 비활성화되고, 테이블에 대한 데이터베이스 로깅은 다시 시작됩니다.

4 UOW가 실패하면 해당 테이블은 재생성해야 합니다.

```
$ db2 drop table <테이블명>
$ db2look –d <데이터베이스명> -z <스키마명> -t <테이블명> -e –o <출력파일명>
db2 –svtf <출력파일명>
```

TOPIC 0817 뷰

Point

사용자가 원본 테이블의 특정 행과 특정 컬럼들만 액세스할 수 있도록 하려면 뷰를 생성합니다. 한 개 이상의 원본 테이블의 데이터를 조인하여 뷰를 생성하는 것도 가능합니다. 기본 테이블에 대해 최소한 SELECT 특권이 있어야 합니다..

Tip

원본 테이블이 제거되면 뷰는 작동 불능 상태가 되어서 액세스를 할 수 없습니다. 원본 테이블이 다시 생성되어도 작동 불능 상태의 뷰는 액세스할 수 없으므로, 재생성이 필요합니다.

1. create view 문으로 생성하며, SELECT문으로 액세스가 허용되는 데이터를 제한합니다.

```
$ db2 "create view kes.t1_v1 as select * from kes.t1 where c2 > 200 "
```

2. WITH CHECK OPTION을 이용하면, 뷰의 정의에 맞지 않는 데이터를 추가, 삭제, 변경할 수 없습니다. INSERT 또는 UPDATE를 실행할 때 뷰의 정의에 맞는지를 확인합니다.

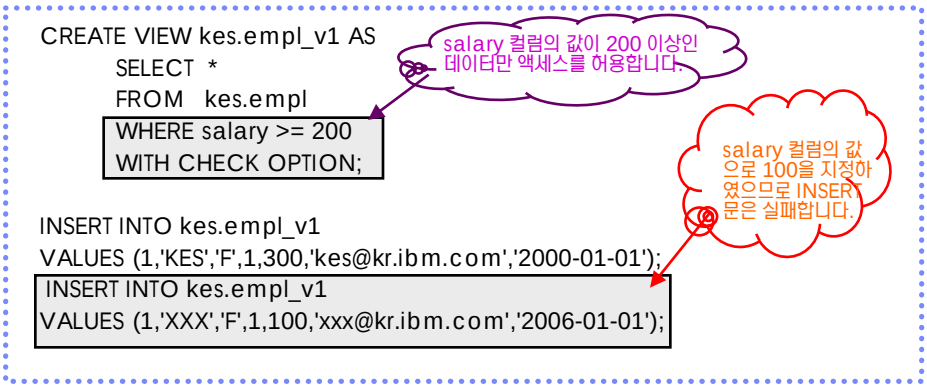


Figure 0817A ••• WITH CHECK OPTION 옵션이 있는 뷰

3. VIEW를 통한 액세스가 가능합니다.

```
$ db2 "insert into kes.t1_v1 values (1,300)"
$ db2 "select * from kes.t1_v1"
```

4. drop view 문을 이용하여 제거합니다.

```
$ db2 drop view kes.t1_v1
```

5. list tables 명령어에서 뷰의 목록을 확인할 수 있습니다.

```
$ db2 list tables
$ db2 list tables for schema <스키마명>
$ db2 list tables for system
$ db2 list tables for all
```

6. 뷰에 대한 정보는 SYSCAT.VIEWS, SYSCAT.VIEWDEP, SYSCAT.TABLES 뷰를 이용해서 확인합니다. SYSCAT.TABLES 뷰에서 TYPE 컬럼의 값이 'V' 입니다.

```
$ db2 "select * from syscat.tables"
```

7. db2look 명령어를 이용하여 뷰에 대한 DDL을 추출합니다.

```
$ db2look –d <DB명>  –e –z <스키마명> –v <뷰명> -o <출력파일명>
```

TOPIC 08-18 CREATE VIEW 문

Point

뷰를 생성하는 SQL문입니다. SELECT 문을 이용하여 뷰를 통하여 액세스할 수 있는 데이터의 범위를 결정하며, WITH CHECK OPTION 옵션으로 뷰의 정의에 맞지 않는 데이터에 대한 액세스를 허용하지 않게 합니다.

Tip

SYSADM, DBADM 권한 또는 테이블에 대한 SEELCT 특권이 필요할 수 있습니다.

Tip

새로운 스키마명을 이용하여 테이블명을 지정하려면, 데이터베이스에 대한 IMPLICIT_SCHEMA 특권이 필요합니다.

Tip

기존의 스키마명을 이용하여 테이블명을 지정하려면, 스키마에 대한 CREATIN 특권이 필요합니다.

1 create view 문의 형식은 다음과 같습니다.

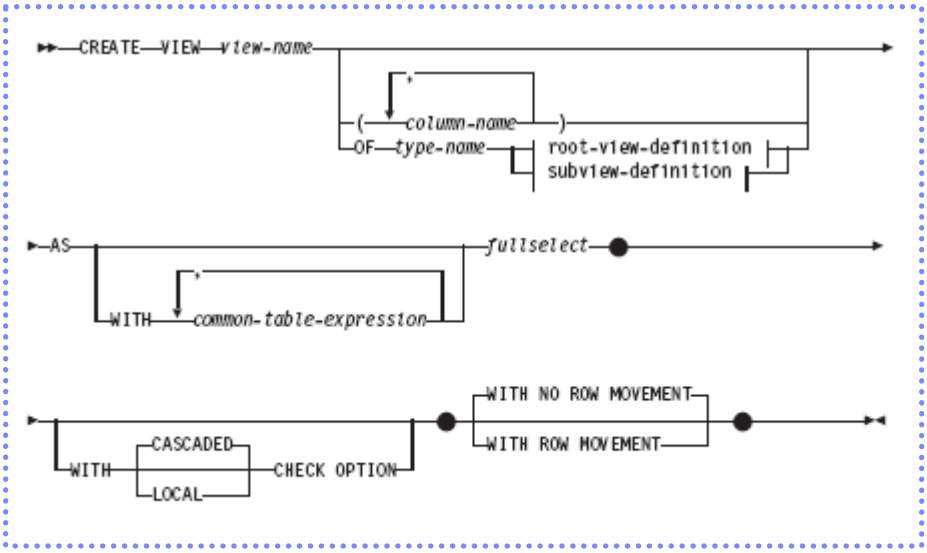


Figure 0818A CREATE VIEW 문

2 옵션에 대한 설명은 다음과 같습니다.

모 드	설 명
<뷰명>	임의의 고유한 이름으로 지정합니다.
<컬럼명>	지정하지 않으면 베이스 테이블의 컬럼명 또는 SELECT문의 결과 컬럼명이 사용됩니다.
WITH <공통 테이블 표현식>	SELECT문에서 사용될 공통 테이블을 정의합니다.
AS <select문>	뷰의 내용이 되는 SELECT문을 지정합니다.
WITH CASCADE CHECK OPTION	뷰의 정의에 맞지 않는 데이터를 처리하지 않습니다. 해당 뷰를 이용한 다른 뷰를 생성했를 때, 이 특성을 전달합니다.
WITH LOCAL CHECK OPTION	뷰의 정의에 맞지 않는 데이터를 처리하지 않습니다. 해당 뷰를 이용한 다른 뷰를 생성했를 때, 이 특성을 전달하지 않습니다.

TOPIC 08-19 MQT

Point

집계 함수를 이용한 summary table 을 미리 정의해 두면, 집계 함수를 이용한 복잡한 쿼리의 실행 시간을 단축시킬 수 있습니다. CREATE TABLE 문으로 생성하고, REFRESH TABLE 문으로 관리합니다.

Tip

SUMMARY 테이블이라고도 합니다.

Tip

immediate 옵션을 사용해도 최초에 한 번은 refresh table 명령어를 실행해야 합니다.

Tip

refresh deferred 옵션을 사용하면, refresh table 문을 실행하기 전까지는 완벽한 데이터 일관성을 보장하지 않습니다.

1 Meterialized Query Table의 약자입니다. create table 문에서 REFRESH 옵션으로 생성합니다.

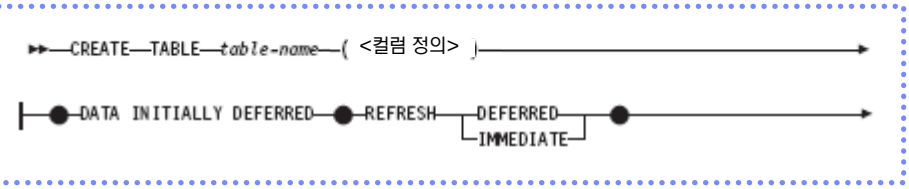


Figure 0819A CREATE TABLE 문의 MQT 생성 옵션

2 사용하기 전에 REFRESH TABLE 문을 이용하여 최초의 결과 집합을 생성해야 합니다.

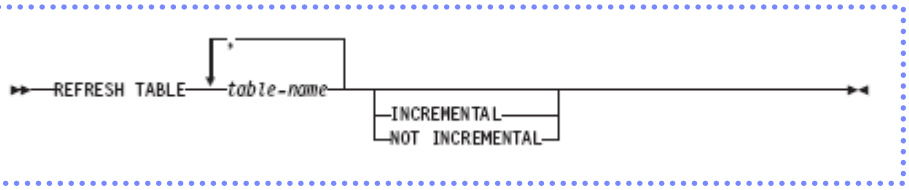


Figure 0819B REFRESH TABLE 문

3 SELECT문에서 직접 MQT를 이용할 수 있습니다.

```
$ db2 "select * from <MQT명>"
```

4 베이스 테이블을 대상으로 하는 SELECT문에서 명시한 조건문이 MQT의 정의 부분과 일치하면, MQT를 이용한 SELECT문으로 자동으로 변환됩니다.

```
CREATE TABLE staff_summary AS
  (SELECT   dept
           ,COUNT(*) AS count_rows
           ,SUM(id)  AS sum_id
   FROM     staff
   GROUP BY dept)
DATA INITIALLY DEFERRED REFRESH IMMEDIATE;

REFRESH TABLE staff_summary;

ORIGINAL QUERY
==============
SELECT   dept
        ,AVG(id)
FROM     staff
GROUP BY dept

OPTIMIZED QUERY
===============================
SELECT  Q1.dept AS "dept"
       ,Q1.sum_id / Q1.count_rows
FROM    staff_summary AS Q1
```

SELECT문이 자동으로 MQT를 사용할 수 있도록 변환됩니다.

AVG는 SUM/COUNT 로 변형이 가능합니다.

Figure 0819C MQT를 이용하도록 변환되는 쿼리

5 drop table 문을 이용하여 제거합니다.

```
$ db2 drop table <MQT명>
```

TOPIC 08 20 인덱스

Point

효율적인 데이터 액세스를 위해서 한 테이블에 한 개 이상의 인덱스를 생성할 수 있습니다. CREATE INDEX 문과 DROP INDEX 문으로 관리합니다.

Tip

고유 인덱스는 NULL 값을 허용하며, NULL 값을 가진 행은 한 개만 허용됩니다.

Tip

테이블에 기본키 또는 고유키를 정의하면 해당 컬럼에 대한 자동으로 UNIQUE 인덱스가 생성됩니다.

Tip

CLUSTER 옵션을 가진 인덱스는 한 테이블에 한 개만 가능하므로, 가장 중요한 인덱스를 CLUSTER 인덱스로 정의합니다.

Tip

INCLUDE 옵션은 UNIQUE 인덱스에서만 사용 가능합니다.

1. create index 문으로 컬럼명과 컬럼별 정렬 순서를 지정합니다. 기본적으로 인덱스는 중복된 값을 허용하므로 중복된 행을 허용하지 않는 인덱스를 생성하려면 UNIQUE 옵션을 이용합니다.

```
$ db2 "create index <스키마명>.<인덱스명> on <테이블명> (<컬럼명>)"
$ db2 "create UNIQUE index <인덱스명> on <테이블명> (<컬럼명>)"
```

2. CLUSTER 옵션을 이용하면, 해당 인덱스의 정렬 순서를 기준으로 테이블의 데이터가 물리적으로 배치되므로 효율적인 액세스가 가능합니다.

```
$ db2 "create unique index <인덱스명> on <테이블명>(<컬럼명>) CLUSTER"
```

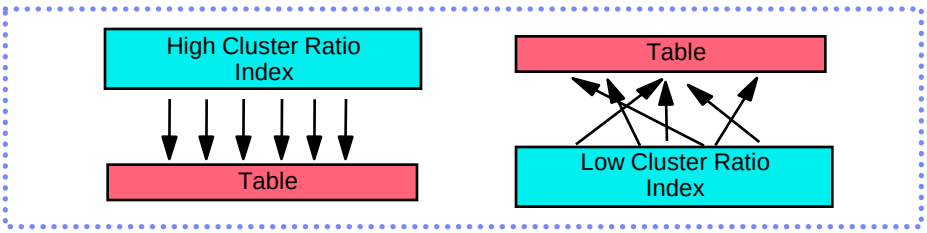


Figure 0820A 인덱스의 클러스터 비율

3. INCLUDE 옵션으로 추가된 컬럼들은 인덱스의 데이터 페이지에 RID 와 함께 저장되어, 인덱스 전용 액세스를 가능하게 합니다. 반드시 UNIQUE 옵션을 함께 지정해야 합니다.

```
$ db2 "create UNIQUE  index <인덱스명> on <테이블명> (<컬럼명 1>)
INCLUDE (<컬럼명 2>, <컬럼명 3>, ...)"
```

4. ALLOW REVERSE SCANS 옵션으로 생성된 인덱스는 양방향 액세스를 허용합니다.

```
$ db2 "create unique index <인덱스명> on <테이블명> (<컬럼명>) cluster
    ALLOW REVERS SCANS"
```

5. drop index 문으로 제거하며, 테이블이 제거되면 자동으로 제거됩니다.

```
$ db2 "drop index <스키마명>.<인덱스명>"
```

6. 인덱스에 대한 정보는 SYSCAT.INDEXES 뷰 또는 describe indexes 명령어를 이용하여 확인합니다.

```
$ db2 "select * from syscat.indexes"
$ db2 describe indexes for table <스키마명>.<테이블명> show detail
```

7. db2look 명령어를 이용하여 테이블과 함께 인덱스에 대한 DDL을 추출합니다.

```
$ db2look –d <DB명> –e –z <스키마명> –v <테이블명> -o <출력파일명>
```

TOPIC 08 21 CREATE INDEX 문

Point

테이블에 인덱스를 생성하는 SQL문입니다. UNIQUE, CLUSTER, INCLUDE, ALLOW REVERSE SCANS 등의 옵션을 가진 인덱스를 생성할 수 있습니다.

Tip

SYSADM, DBADM 권한 또는 테이블에 대한 CONTROL, INDEX 특권이 필요할 수 있습니다.

Tip

새로운 스키마명을 이용하여 인덱스명을 지정하려면, 데이터베이스에 대한 IMPLICIT_SCHEMA 특권이 필요합니다.

Tip

기존의 스키마명을 이용하여 인덱스명을 지정하려면, 스키마에 대한 CREATIN 특권이 필요합니다.

Tip

INCLUDE 옵션에서 지정한 컬럼은 인덱스를 구성하는 컬럼이 아니므로, 조건식에서 이 컬럼을 이용하여 검색을 요청할 때, index scan 이 선택되지 않습니다.

1 create index 문의 형식은 다음과 같습니다.

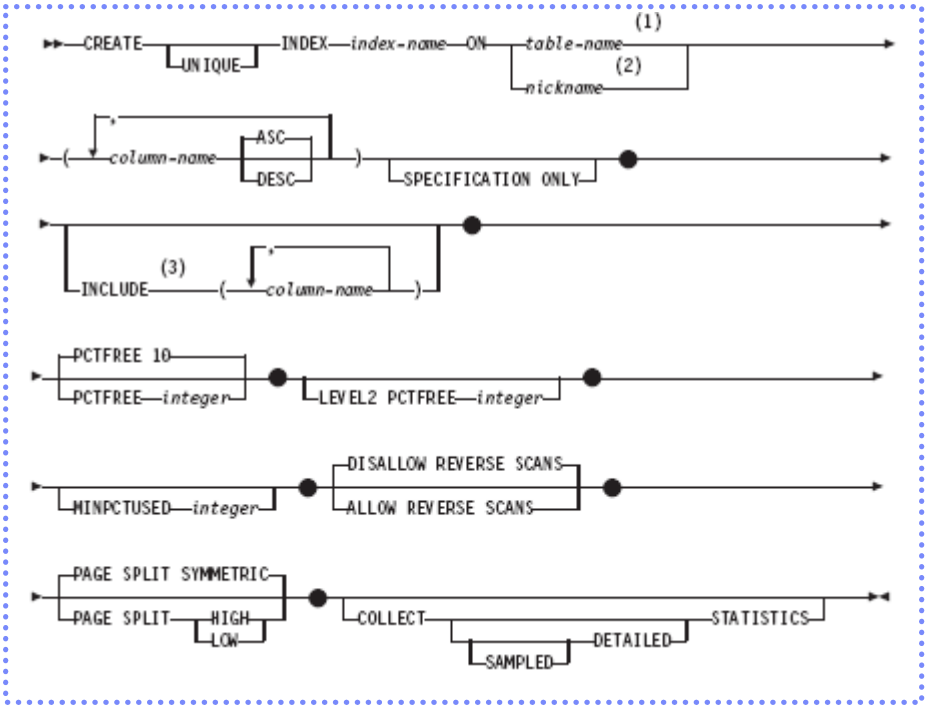


Figure 0821A CREATE INDEX 문

2 옵션에 대한 설명은 다음과 같습니다.

모 드	설 명
UNIQUE	고유 인덱스로 생성합니다.
<인덱스명>	임의의 고유한 이름으로 지정합니다.
ON <테이블명>	인덱스가 생성될 테이블명을 지정합니다.
<컬럼명>	인덱스를 구성하는 컬럼명을 지정합니다. 한 개 이상인 경우에는 ,(컴마)로 구분합니다.
ASC / DESC	컬럼별 정렬 순서를 오름차순(ASC) 또는 내림차순(DESC)로 지정합니다. 기본값은 ASC 입니다.
INCLUDE (<컬럼명>)	인덱스의 리프 페이지에 RID와 함께 저장되는 컬럼을 지정합니다. 한 개 이상인 경우에는 ,(컴마)로 구분합니다.
CLUSTER	클러스터 인덱스로 지정합니다.
PCTFREE	인덱스의 각 페이지의 여유 공간 비율로 백분율로 지정합니다. 기본값은 10 입니다.
MINPCTUSED	인덱스의 리프 페이지의 최소 사용 공간 비율로 백분율로 지정합니다. 50 이하로 지정하는 것이 성능에 유리합니다.
ALLOW REVERSE SCANS	역방향 액세스를 허용합니다.

TOPIC 08 22 시퀀스

Point

데이터베이스 지원에서 제공되는 자동 생성 일련 번호를 시퀀스라고 합니다. CREATE SEQUENCE 문, ALTER SEQUENCE 문, DROP SEQUENCE 문으로 관리합니다.

Tip

데이블의 identity 컬럼과 기능상 동일합니다. identity 컬럼은 해당 테이블에서만 유효하며, 시퀀스는 데이터베이스 수준에서 운영됩니다.

1 create sequence 문의 형식은 다음과 같습니다.

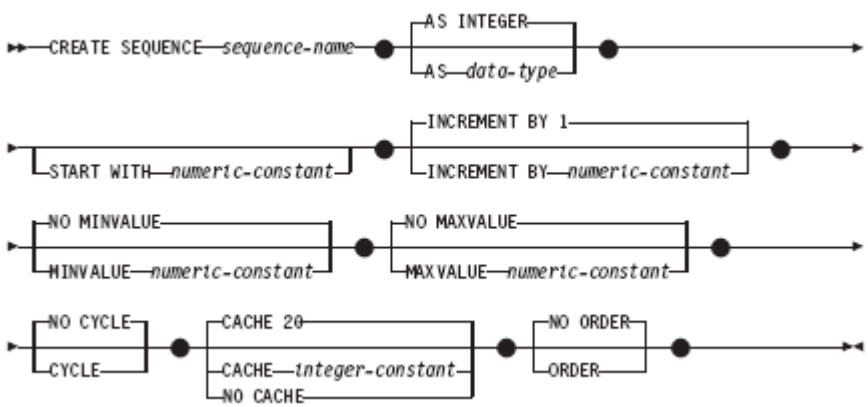


Figure 0822A ••• CREATE SEQUENCE 문

Tip

데이터베이스가 비활성화되면, 사용되지 않고 캐쉬에 남아있던 자동 생성 값은 유실됩니다.

2 주요한 옵션은 다음과 같습니다.

옵션	설명
<시퀀스명>	임의의 고유한 이름으로 지정합니다.
AS <데이터유형>	SAMLLINT, INT, BIGINT, DECIMAL 중에서 원하는 데이터 유형을 선택합니다.
START WITH	양수 또는 음수의 시작값을 지정합니다.
INCREMENT BY	양수 또는 음수의 증가값을 지정합니다.
MINVALUE	양수 또는 음수의 최소값을 지정합니다.
MAXVALUE	양수 또는 음수의 최대값을 지정합니다.
CYCLE	최대값 또는 최소값에 도달하면 최소값 또는 최대값을 생성합니다.
CACHE	지정된 개수의 생성값을 미리 캐쉬에 보관하여 성능에 유리합니다.

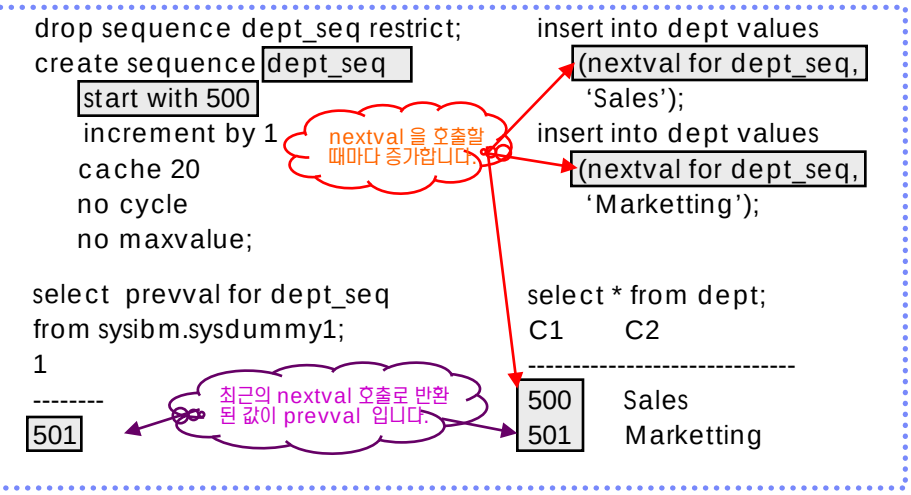


Figure 0822B ••• PREVVAL 과 NEXTVAL

TOPIC 08 23 트리거

Point

특정 테이블에 INSERT, UPDATE, DELETE 문이 실행될 때 자동으로 실행되는 일련의 작업들을 정의합니다. 트리거의 이벤트 유형은 BEFORE, AFTER, INSTEAD OF가 있으며, CREATE TRIGGER 문과 DROP TRIGGER 문으로 관리합니다.

Tip

트리거가 구현하는 비즈니스 규칙은 데이터베이스 오브젝트에 대한 일련의 변경 작업 또는 예외 처리 로직입니다.

Tip

트리거는 특정 조건을 만족하는 경우에만 SQL문의 요청을 허용하기 위한 용도로 사용되기도 합니다.

1 한 데이터베이스의 특정 테이블에 대한 변경 작업이 다른 테이블에 영향을 미칠 수 있습니다. 응용프로그램의 로직에서 이러한 비즈니스 규칙을 구현하면, 비즈니스 규칙이 변경될 때마다 응용프로그램의 로직을 수정해야 합니다. 트리거는 특정 테이블 또는 뷰에 대한 변경 작업이 요청될 때마다 자동으로 실행되어야 하는 일련의 작업들을 데이터베이스 수준에서 정의합니다. 비즈니스 규칙이 변경되되어도 데이터베이스에 존재하는 트리거의 정의만 변경하면 되므로, 모든 응용프로그램은 추가적인 로직의 변경 없이 새로운 비즈니스 규칙을 적용할 수 있습니다.

Figure 0823A ••• 비즈니스 규칙

2 트리거에 정의된 비즈니스 로직을 실행하는 시점에 의해 3가지 유형으로 분류됩니다.

유 형	설 명
AFTER 트리거	특정 테이블에 대해 요청된 INSERT, UPDATE, DELETE 문을 먼저 실행하고, 정의된 일련의 작업들을 실행합니다.
BEFORE 트리거	정의된 일련의 작업들을 먼저 실행하고, 특정 테이블에 대해 요청된 INSERT, UPDATE, DELETE 문을 실행합니다.

3 트리거는 발생시키는 SQL문의 유형에 의해 3가지 유형으로 분류됩니다.

유 형	설 명
INSERT 트리거	특정 테이블에 대해 INSERT문이 요청된 경우에 실행됩니다.
UPDATE 트리거	특정 테이블에 대해 UPDATE문이 요청된 경우에 실행됩니다. 특정 컬럼을 지정할 수도 있습니다.
DELETE 트리거	특정 테이블에 대해 DELETE문이 요청된 경우에 실행됩니다.

4 트리거의 기준이 되는 대상은 2 가지로 분류됩니다.

유 형	설 명
테이블 트리거	특정 테이블에 INSERT, UPDATE, DELETE 문이 요청될 때 요청된 SQL문과 정의된 일련의 작업들을 실행합니다.
뷰 트리거 INSTEAD OF 트리거	특정 뷰에 INSERT, UPDATE, DELETE 문을 요청하면, 요청된 SQL문 대신에 정의된 일련의 작업들을 실행합니다.

TOPIC 08-24 CREATE TRIGGER 문

Point

트리거를 생성하는 SQL문입니다. 단일 SQL문 또는 SQL/PL을 이용하여 트리거가 실행할 로직을 구현합니다. INSERT, UPDATE, DELETE 트리거는 개별적으로 생성하고, 트리거의 정의에 변경이 있을 때는 재 생성합니다.

Tip
SYSADM, DBADM 권한 또는 테이블에 대한 CONTROL 특권이 필요할 수 있습니다

Tip
IMPLICIT_SCHEMA 데이터베이스 특권, 스키마에 대한 ALTERIN, CREATEIN 특권이 필요합니다.

Tip
BEFORE, AFTER 트리거를 정의할 때는 테이블에 대한 ALTER 특권이 필요합니다.

Tip
INSTEAD OF 트리거를 정의할 때는 뷰에 대한 CONTROL 특권이 필요합니다.

Tip
TRANSITION 변수 또는 테이블을 사용할 때는 테이블에 대한 SELECT 특권이 필요합니다.

Tip
transition 행 또는 테이블은 트리거의 비즈니스 로직에서 이용할 수 있습니다. DELETE, UPDATE 트리거에서 사용됩니다.

Tip
UPDATE 옵션은 모든 컬럼의 변경에 적용됩니다. OF 옵션은 지정한 컬럼의 UPDATE 시에만 적용됩니다.

Tip
트리거 본문에 정의되는 비즈니스 로직은 기본적으로 단일 SQL문입니다. 여러 개의 SQL문과 논리적인 로직을 구현하려면 BEGIN~END 블록을 이용하여SQL/PL을 이용합니다.

1 create trigger 문의 형식은 다음과 같습니다.

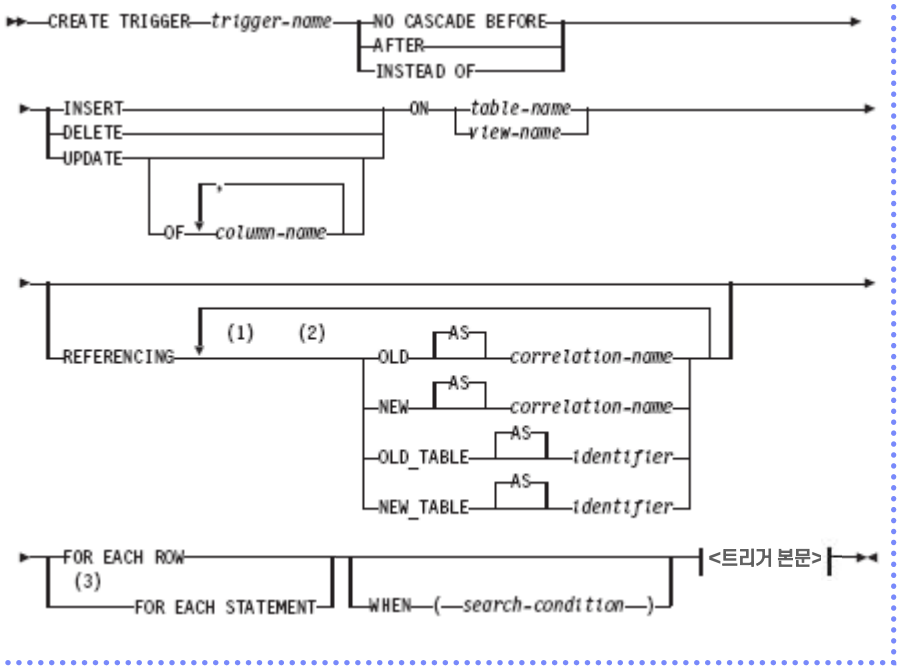


Figure 0824A CREATE TRIGGER 문

2 옵션에 대한 설명은 다음과 같습니다.

모 드	설 명
<트리거명>	임의의 고유한 이름으로 지정합니다.
NO CASCADE BEFORE	BEFORE 트리거를 생성합니다.
AFTER	AFTER 트리거를 생성합니다.
INSTEAD OF	INSTEAD OF 트리거를 생성합니다.
INSERT	INSERT 트리거를 생성합니다.
DELETE	DELETE 트리거를 생성합니다.
UPDATE OF <컬럼명>	UPDATE 트리거를 생성합니다.
REFERENCING	transition 테이블 또는 행의 이름을 지정합니다.
OLD AS <id> OLD_TABLE AS <id>	요청된 SQL문이 실행되기 전의 행 또는 테이블의 값을 보관하고 있는 transition 행 또는 테이블 id 입니다.
NEW AS <id> NEW_TABLE AS <id>	요청된 SQL문이 실행된 후의 행 또는 테이블의 값을 보관하고 있는 transition 행 또는 테이블 id 입니다.
FOR EACH ROW	조건에 맞는 모든 행에 개별적으로 트리거가 적용됩니다.
FOR EACH STATEMENT	조건에 맞는 행의 개수에 상관없이 한 번만 실행됩니다.
WHEN <조건식>	<조건식>을 만족하는 경우에만 본문의 로직을 실행합니다.
<트리거 본문>	SQL/PL 을 이용하여 로직을 작성합니다.

TOPIC 08 25 BEFORE 트리거

Point

특정 테이블에 대한 INSERT, UPDATE, DELETE 문을 실행하기 전에 먼저 정의된 일련의 작업들을 실행합니다. CREATE TRIGGER 문에서 NO CASCADE BEFORE 옵션을 사용하여 생성합니다.

Tip

INSERT, UPDATE, DELETE 문에 대한 트리거를 한 개 이상 생성할 수 있습니다.

1 create trigger 문에서 NO CASCADE BEFORE 옵션으로 생성합니다.

```
CREATE TABLE kes.class (
        number      smallint    not null
      , name        varchar(20)
      , length      smallint
);

CREATE TABLE kes.test (
        id          int   not null
      , number      smallint
      , test_date   date
      , start_time  time
      , seat        smallint
      , score       smallint
);

INSERT INTO kes.class VALUES (1,'DB2',60);

CREATE TRIGGER pre9
NO CASCADE BEFORE INSERT ON kes.test
REFERENCING NEW AS N
FOR EACH ROW
MODE DB2SQL
WHEN (N.START_TIME < '09:00:00')
      SIGNAL SQLSTATE '70003'
      ('09:00 이전에는 좌석을 예약할 수 없습니다!');

INSERT INTO kes.test VALUES (1,1,'2006-04-19','08:30:00',2,NULL);

CREATE TRIGGER aft5
NO CASCADE BEFORE INSERT ON kes.test
REFERENCING NEW AS N
FOR EACH ROW
MODE DB2SQL
WHEN (N.START_TIME +
      (SELECT SMALLINT(LENGTH)
       FROM test
       WHERE NUMBER = N.NUMBER) MINUTES > '17:00:00')
       SIGNAL SQLSTATE '70004'
       ('16:00 이후에는 좌석을 예약할 수 없습니다!');

INSERT INTO kes.test VALUES (3,1,'2006-04-19','16:30:00',1,NULL);
INSERT INTO kes.test VALUES (2,1,'2006-04-19','10:00:00',1,NULL);

SELECT    id, number, start_time
FROM      kes.test;
```

실패 : SQL0438N 응용프로그램이 진단 텍스트 "09:00 이전에는 좌석을 예약할 수 없습니다!"과(와)함께 오류를 표시했습니다. SQLSTATE=70003

실패 : SQL0438N 응용프로그램이 진단 텍스트 "16:00 이후에는 좌석을 예약할 수 없습니다!"과(와) 함께 오류를 표시했습니다. SQLSTATE=70004

```
ID          NUMBER START_TIME
----------- ------ ----------
          2      1 10:00:00
```

Figure 0825A ··· BEFORE 트리거

TOPIC 08-26 AFTER 트리거

Point

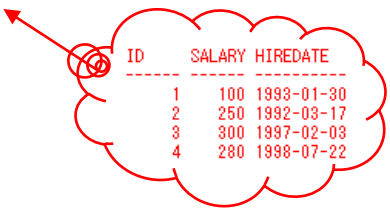
특정 테이블에 대해 요청된 INSERT, UPDATE, DELETE 문을 먼저 실행하고, 정의된 비즈니스 로직을 실행합니다. 최대 16 레벨까지 다른 트리거를 연쇄적으로 수행할 수 있습니다. CREATE TRIGGER 문에서 AFTER 옵션을 사용하여 생성합니다.

Tip

INSERT, UPDATE, DELETE 문에 대한 트리거를 한 개 이상 생성할 수 있습니다.

1 AFTER 트리거를 생성하는 예는 다음과 같습니다.

```
CREATE TABLE kes.empl (
       id          smallint    not null
     , name        varchar(30)  not null
     , sex         char(1)
     , mydept      smallint
     , salary      smallint
     , email       varchar(30)  not null
     , hiredate    date
);

INSERT INTO kes.empl VALUES
       (1,'KES','F',1,100,'kes@kr.ibm.com','1993-01-30')
     , (2,'KHY','F',3,250,'khy@kr.ibm.com','1992-03-17')
     , (3,'JHS','F',2,300,'jhs@kr.ibm.com','1997-02-03')
     , (4,'JJY','M',2,280,'jjy@kr.ibm.com','1998-07-22');

CREATE TABLE kes.log (
       id          smallint
     , osalary     smallint
     , nsalary     smallint
     , ts          timestamp);

CREATE TRIGGER kes.empl_trig01
     AFTER UPDATE OF salary ON kes.empl
     REFERENCING OLD AS o NEW AS n
     FOR EACH ROW
     MODE DB2SQL
         WHEN ( n.salary > o.salary * 1.2 )
             INSERT INTO kes.log
             VALUES(o.id,o.salary,n.salary,current timestamp );

UPDATE  kes.empl
SET     salary = salary + 30
WHERE   hiredate < '2000-01-01' ;

SELECT     id, salary, hiredate
FROM       kes.empl
ORDER BY id ;

SELECT  id, osalary, nsalary
FROM    kes.log;
```



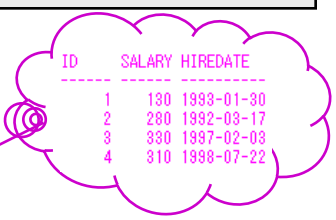


Figure 0826A ••• AFTER 트리거

TOPIC 08-27 INSTEAD OF 트리거

Point

뷰를 대상으로 정의된 트리거로 테이블에 대한 변경 SQL문을 뷰를 통해서 관리하거나, 뷰에 직접 허용되지 않는 변경 SQL문을 트리거 로직으로 대신 실행할 때도 사용됩니다. CREATE TRIGGER 문에서 INSTEAD OF 옵션을 사용하여 생성합니다.

Tip

특정 뷰에 대한 INSERT, UPDATE, DELETE 문을 처리하는 트리거는 한 개씩만 생성할 수 있습니다.

Tip

FOR EACH STATEMENT 옵션은 사용할 수 없습니다.

1 create trigger 문에서 INSTEAD OF 옵션으로 생성합니다.

```
CREATE TABLE kes.empl (
        id              smallint    not null
      , name            varchar(30)  not null
      , sex             char(1)
      , mydept          smallint
      , salary          smallint
      , email           varchar(30)  not null
      , hiredate        date
);

CREATE VIEW kes.empl_v AS
SELECT          id, name, sex
FROM            kes.empl;

CREATE TRIGGER kes.empl_v_insert
    INSTEAD OF INSERT ON kes.empl_v
    REFERENCING NEW AS N
    FOR EACH ROW
    MODE DB2SQL
        INSERT INTO kes.empl VALUES (
                n.id
              , n.name
              , CASE RTRIM(LTRIM(UPPER(n.sex)))
                    WHEN  'M' THEN 'M'
                    WHEN  'F' THEN 'F'
                    ELSE NULL
                END
              , NULL
              , NULL
              , LOWER(n.name)||'@kr.ibm.com'
              , CURRENT DATE – 2 days);

INSERT INTO kes.empl_v VALUES (1,'KES','');
INSERT INTO kes.empl_v VALUES (2,'KHY','f');
INSERT INTO kes.empl_v VALUES (3,'JHS','F');
INSERT INTO kes.empl_v VALUES (4,'JJY','m');
INSERT INTO kes.empl_v VALUES (5,'XXX','X');

SELECT    id
        , substr(name,1,3) name
        , sex
        . email
FROM    kes.empl_v
ORDER BY id;
```

sex 컬럼의 값이 'M', 'm', 'F', 'f' 인 경우에 대문자로 변환하여 입력합니다. '' 값인 경우에는 NULL 값으로 입력합니다.

email 컬럼과 hiredate 컬럼에는 name컬럼의 값과 시스템 날짜를 이용한 값이 입력됩니다.

```
ID     NAME  SEX EMAIL
------ ----- --- ---------------
     1 KES   -   kes@kr.ibm.com
     2 KHY   F   khy@kr.ibm.com
     3 JHS   F   jhs@kr.ibm.com
     4 JJY   M   jjy@kr.ibm.com
     5 XXX   -   xxx@kr.ibm.com
```

Figure 0827A ··· INSTEAD OF 트리거

TOPIC 08-28 사용자 정의 데이터 유형

Point

사용자가 추가적으로 정의할 수 있는 데이터 유형입니다. 일반적으로 도량형과 관련된 데이터를 저장할 때 이용합니다. CREATE DISTINCT TYPE 문과 DROP DISTINCT TYPE 문으로 관리합니다.

1 단위가 서로 다른 의미를 가지는 두 개의 값을 단순히 값으로만 비교하는 것은 잘못된 결과를 만들게 됩니다.

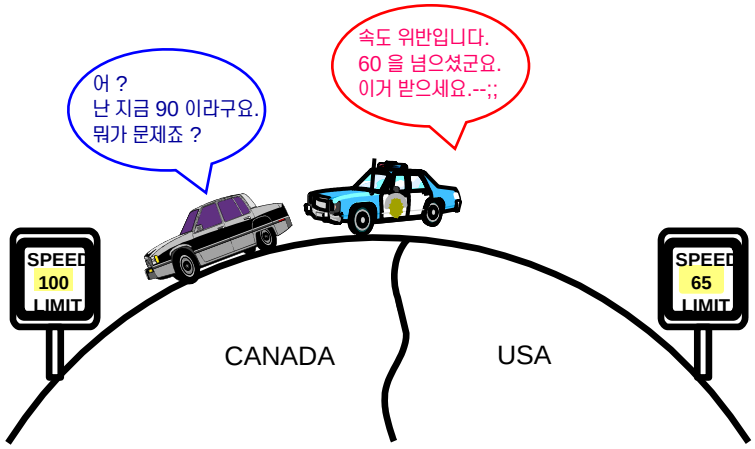


Figure 0828A • • • UDT 의 필요성

Tip

with comparisos 옵션은 새로 생성한 UDT 형으로 형 변환을 할 수 있는 형변환(CAST) 함수를 기본적으로 제공합니다.

2 단위가 다른 두 값을 직접 비교하는 것을 방지하기 위해 사용자가 create distinct type 문으로 새로운 데이터 유형을 생성할 수 있습니다. 유형이 다른 두 데이터는 서로 직접 비교될 수 없습니다.

```
CREATE DISTINCT TYPE dollar AS INTEGER WITH COMPARISONS ;
CREATE DISTINCT TYPE won AS INTEGER WITH COMPARISONS ;

CREATE TABLE kes.person VALUES
  ( f_name          VARCHAR (30)
  , money_d         dollar NOT NULL
  , money_w         won NOT NULL ) ;

INSERT INTO kes.person
VALUES (KES', 80, 70),('JHS',80,100);

SELECT f_name FROM  kes.person
WHERE money_d < money_w;

SELECT f_name FROM  kes.person
WHERE money_d < dollar(money_w)
OR      won(money_d) < money_w;
```

money_d 와 money_w 컬럼을 다른 유형이므로 직접 비교가 불가능합니다.

money_w 컬럼을 dollar 유형으로 형변환하였으므로 비교가 가능합니다.

money_d 컬럼을 won 유형으로 형변환하였으므로 비교가 가능합니다.

Figure 0828B • • • 다른 데이터 유형간의 비교 실패

TOPIC 08 29 사용자 정의 함수

Point

사용자가 직접 구현하여 엔진에 추가적으로 정의한 함수입니다. 사용자 정의 함수는 기존의 내장 SQL 함수와 동일한 방법으로 SQL문에서 사용됩니다. CREATE FUNCTION문과 DROP FUNCTION 문으로 관리합니다.

1 DB2에서 사용하는 함수는 생성 주체에 따라 2 가지 유형으로 구분됩니다.

유 형	설 명
빌트인	엔진에 의해 기본적으로 제공되는 함수입니다. 내장 함수라고 합니다.
사용자 정의	사용자가 새롭게 작성하여 엔진에 추가한 함수입니다.

2 적용되는 대상과 반환하는 값의 유형에 따라 함수는 4 가지 유형으로 구분됩니다.

유 형	설 명
스칼라	조건에 맞는 모든 행에 대해서 각각 함수를 실행하여 결과를 반환합니다. 문자 함수, 수학 함수, 날짜 함수등이 있습니다.
컬럼	조건에 맞는 행들을 지정된 값에 의해 그룹으로 분류하고, 각 그룹별로 함수를 적용시킨 결과값을 반환합니다. 집계 함수가 해당됩니다.
행	조건에 맞는 결과 행을 Row 형태로 반환하는 함수입니다.
테이블	조건에 맞는 결과 집합을 table 구조로 반환합니다. SELECT 문의 FROM 절에서 사용되며, 일반 테이블과 동일한 방법으로 액세스할 수 있습니다. 스냅샷 함수가 해당됩니다.

3 사용자 정의 함수의 유형은 작성하는 언어와 반환하는 결과의 유형에 따라 7 가지로 분류됩니다.

유 형	설 명
외부 스칼라	C, Java 등의 프로그래밍 언어로 작성된 라이브러리를 이용하며, 스칼라 값을 반환합니다.
외부 테이블	C, Java 등의 프로그래밍 언어로 작성된 라이브러리를 이용하며, 테이블을 반환합니다.
OLE DB 외부 테이블	OLE DB 공급자로부터 데이터를 액세스하기 위한 함수입니다.
소스 함수	기존의 내장, 외부, SQL, 소스 함수를 이용하여 작성된 함수입니다.
SQL 스칼라	SQL/PL 로 작성되며, 스칼라값을 반환합니다.
SQL 행	SQL/PL 로 작성되며, 행을 반환합니다.
SQL 테이블	SQL/PL 로 작성되며, 테이블을 반환합니다.

4 사용자 정의 함수는 SQL/PL, C, Java, OLE 등을 이용하여 생성합니다.

5 사용자 정의 함수에 대한 정보는 SYSCAT.ROUTINES 뷰를 이용해서 확인합니다.

```
$ db2 "select * from syscat.routines"
```

6 db2look 명령어로 SQL 사용자 정의 함수에 대한 DDL문을 추출할 수 있습니다.

```
$ db2look –d <DB명> –e  -o <출력파일명>
```

Tip

기존의 함수에서 UDT 유형으로 정의된 컬럼을 사용될 수 없습니다. 기존의 함수를 소스 함수로 하는 새로운 UDF가 필요합니다.

Tip

Java 언어를 이용하면, JAR 형태로 UDF를 관리할 수 있습니다.

Tip

사용자 정의 함수를 작성한 언어에 따라 라이브러리를 생성하는 방법이 다릅니다. SQL/PL로 작성하는 경우에는 별도의 생성 과정 없이 create function 문으로 생성합니다.

TOPIC 08-30 CREATE FUNCTION 문

Point

SQL/PL을 이용하여 작성한 사용자 정의 함수의 라이브러리를 생성하고 시그너처를 등록하는 SQL문입니다. C, Java, OLE 등으로 적성된 사용자 정의 함수는 시그너처만 등록되며, 개별적인 방법으로 함수의 라이브러리를 생성해야 합니다.

Tip

SYSADM, DBADM 권한 또는 테이블에 대한 CONTROL, SELECT 특권이 필요할 수 있습니다.

Tip

새로운 스키마명을 이용하여 인덱스명을 지정하려면, 데이터베이스에 대한 IMPLICIT_SCHEMA 특권이 필요합니다.

Tip

기존의 스키마명을 이용하여 인덱스명을 지정하려면, 스키마에 대한 CREATIN 특권이 필요합니다.

1 create function 문의 형식은 다음과 같습니다.

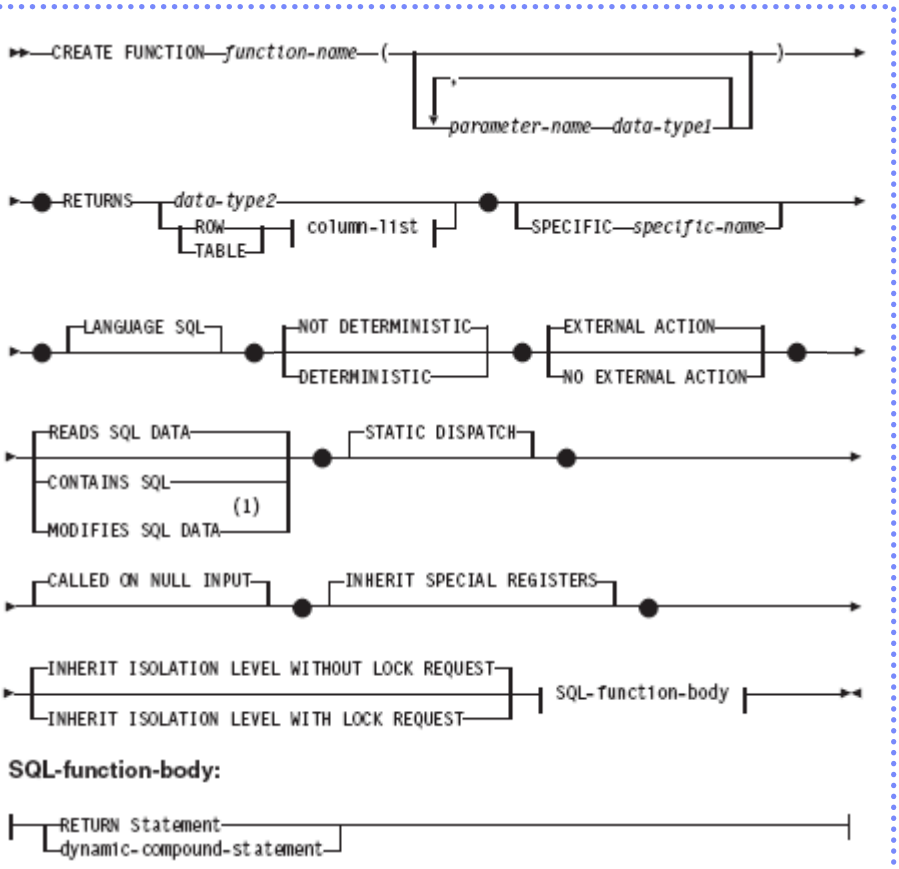


Figure 0830A ··· SQL 사용자 정의 함수를 위한 CREATE FUNCTION 문

Tip

함수의 시그너처가 다르면 동일한 이름으로 함수를 생성할 수 있습니다. 동일한 이름으로 함수를 오버로딩할 때는 SPECIFIC 옵션으로 고유한 함수명을 지정하는 것이 권장됩니다.

Tip

사용자 정의 함수의 <SPECIFIC명>은 <함수명>과 동일하게 지정할 수 있습니다. 옵션을 지정하지 않으면, SQLyymmddhhmmssxxx 형식으로 엔진이 자동 생성합니다.

Tip

SPECIFIC 옵션은 drop 문 또는 다른 함수에서 참조되는 이름으로만 사용됩니다. SQL문에서 실제로 함수를 사용할 때는 <스키마명>.<함수명>을 사용해야 합니다.

2 옵션에 대한 설명은 다음과 같습니다.

모 드	설 명
<함수명>	함수의 이름을 지정합니다.
<인수이름> <데이터유형>	입력 인수의 이름과 데이터 유형을 지정합니다.
RETURNS <데이터유형>	반환되는 값의 데이터 유형입니다.
SPECIFIC <고유함수명>	임의의 고유한 이름으로 지정합니다.
LANGUAGE SQL	SQL/PL을 이용하여 로직을 작성합니다.
DETERMINISTIC	동일한 입력 인수에 대해서 항상 동일한 값을 반환합니다.
CONTAINS SQL	데이터의 조회, 변경을 위한 SQL문을 포함하지 않습니다.
READS SQL DATA	데이터를 변경하는 SQL문을 포함할 수 없습니다.
MODIFIES SQL DATA	지원되는 모든 SQL문을 사용합니다.
CALLED ON NULL INPUT	입력 인수의 값이 NULL인 경우에도 호출됩니다.
<SQL 함수 본문>	단일 SQL문 또는 SQL/PL 블록을 이용하여 구현합니다.

TOPIC 08 31 SQL 사용자 정의 함수

Point

단일 SQL문 또는 SQL/PL을 이용하여 작성한 사용자 정의 함수를 SQL 사용자 정의 함수라고 합니다. CREATE FUNCTION 문으로 함수의 라이브러리가 생성되고, 함수의 시그너처도 등록됩니다. 내장 함수와 동일한 방법으로 SQL문에서 사용합니다.

Tip

단일한 SQL문으로 표현되지 않는 로직을 구현하려면, SQL/PL 의 BEGIN ~ END 블록을 이용합니다.

Tip

SQL/PL에서 ; (세미콜론)은 문장의 구분자로 사용됩니다. create function 문의 끝을 구별할 때는 @ 또는 ! 등의 문자를 사용합니다.

1 SQL 사용자 정의 함수를 위한 로직을 작성하여 임의의 <파일명>으로 저장합니다.

```
$ cat <파일명>
CREATE FUNCTION todate (x varchar(8))
RETURNS date
SPECIFIC TODATE01
LANGUAGE SQL
CONTAINS SQL
NO EXTERNAL ACTION
DETERMINISTIC
RETURN date(SUBSTR(X,1,4)||'-'||SUBSTR(X,5,2)||'-'||SUBSTR(X,7,2))

@
CREATE FUNCTION tan (X DOUBLE)
RETURNS DOUBLE
    LANGUAGE SQL
    CONTAINS SQL
    NO EXTERNAL ACTION
    DETERMINISTIC
    RETURN SIN(X)/COS(X) @

CREATE FUNCTION kes.percent(number int, rate int)
RETURNS decimal(6,2)
F1: BEGIN ATOMIC
   RETURN (number * rate) / 100;
END@
```

Tip

SQL 사용자 정의 함수는 엔진에 내장된 특별한 컴파일러를 이용하여 라이브러리를 작성합니다. 별도의 C 컴파일러는 필요하지 않습니다.

2 create function 문을 이용하여 SQL 사용자 정의 함수를 생성하고, 등록합니다.

```
$ db2 connect to <데이터베이스명>
$ db2 -td@ -svf  <파일명>
```

Tip

UDF를 참조하려면 EXECUTE 권한이 필요합니다. UDF명은 대소문자를 구분하지 않습니다.

3 SQL문에서 기존의 내장 함수와 동일한 방법으로 사용자 정의 함수를 참조합니다.

```
$ db2  –x  " values(TODATE('20040101'))"
2004-01-01

$ db2 -x "SELECT id, salary, kes.percent(salary,5) FROM kes.empl"
    1    130     6.00
    2    280    14.00
    3    330    16.00
    4    310    15.00
```

TOPIC 08 32 저장 프로시저

Point

서버에 저장된 프로그램 로직입니다. 클라이언트에서 CALL 문으로 서버의 저장 프로시저를 호출하면, 서버에서 로직이 실행되어 결과만 클라이언트로 반환됩니다. CREATE PROCEDURE 문과 DROP PROCEDURE 문으로 관리합니다.

1 저장 프로시저는 클라이언트 머신의 응용프로그램에서 실행해야 하는 로직을 서버의 데이터베이스에 저장하여 서버 머신에서 직접 실행함으로써 클라언트와 서버 간의 데이터 전송량을 줄이고, 성능을 향상시킵니다.

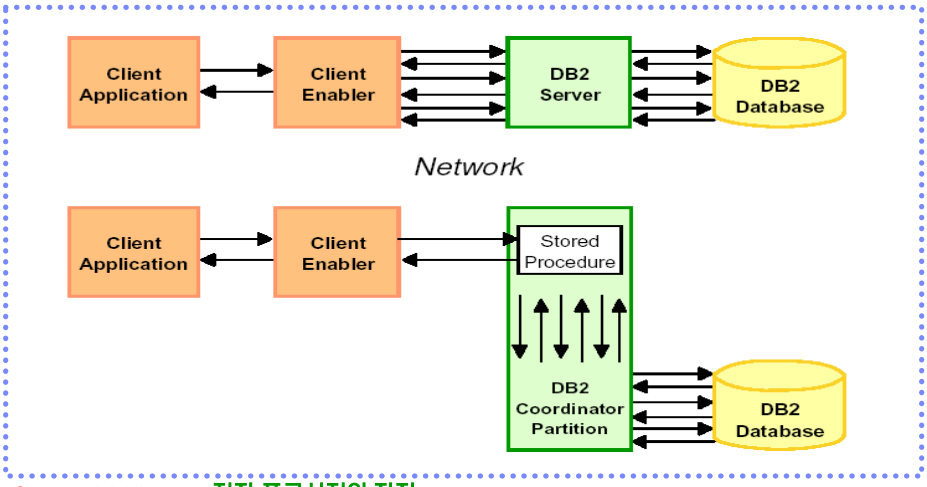


Figure 0832A • • • 저장 프로시저의 장점

2 공용 로직을 저장 프로시저로 만들어서 사용하면 관리가 용이합니다. 클라이언트 응용프로그램의 개별적인 코딩으로 인한 오류와 소스를 반복적으로 작성해야 하는 부담을 줄일 수 있습니다. 프로시저의 로직이 변경되면, 서버의 저장 프로시저만 재생성하면 됩니다.

3 저장 프로시저는 실행시에 서버의 자원을 사용하여 실행됩니다. 일반적으로 클라이언트 머신보다 서버 머신의 사양이 좋으므로, 동일한 로직을 실행할 때 실행 시간이 단축될 수 있습니다.

4 서버 머신의 OS 에 의존적인 로직의 구현이 가능합니다. 클라이언트가 Windows 이고, 서버가 UNIX인 경우에 UNIX 에서만 지원되는 기능을 프로시저의 로직에 포함시킬 수 있습니다.

5 저장 프로시저의 유형은 작성하는 언어에 의해 2 가지로 분류됩니다.

유 형	설 명
외부 소스	ESQL, C, Java 등의 프로그래밍 언어로 작성된 라이브러리를 이용하며, 스칼라 값 또는 결과 집합을 반환합니다.
SQL	SQL/PL 또는 PL/SQL로 작성되며, 스칼라 값 또는 결과 집합을 반환합니다.

Tip
- PL/SQL은 DB2 9.7이후 지원되는 프로시저 언어입니다.

Tip
- 저장 프로시저를 작성한 언어에 따라 라이브러리를 생성하는 방법이 다릅니다. SQL/PL 또는 PL/SQL로 작성하는 경우에는 별도의 생성 과정 없이 create proceudre 문으로 생성합니다.

6 저장 프로시저는 SQL/PL, PL/SQL, ESQL, C, Java 등의 언어를 이용하여 생성합니다.

7 저장 프로시저에 대한 정보는 SYSCAT.ROUTINES 뷰를 이용해서 확인합니다.

```
$ db2 "select * from syscat.routines"
```

8 db2look 명령어로 SQL 저장 프로시저에 대한 DDL문을 추출할 수 있습니다.

```
$ db2look –d <DB명> –e  -o <출력파일명>
```

TOPIC 08 33 CREATE PROCEDURE 문

Point

SQL/PL 저장 프로시저의 라이브러리를 생성하고 시그너처를 등록하는 SQL문입니다. ESQL, C, Java 등으로 적성된 저장 프로시저는 시그너처만 등록되며, 개별적인 방법으로 저장 프로시저의 라이브러리를 생성해야 합니다.

Tip

SYSADM, DBADM 권한 또는 데이터베이스에 대한 BINDADD 특권이 필요할 수 있습니다.

1 create procedure 문의 형식은 다음과 같습니다.

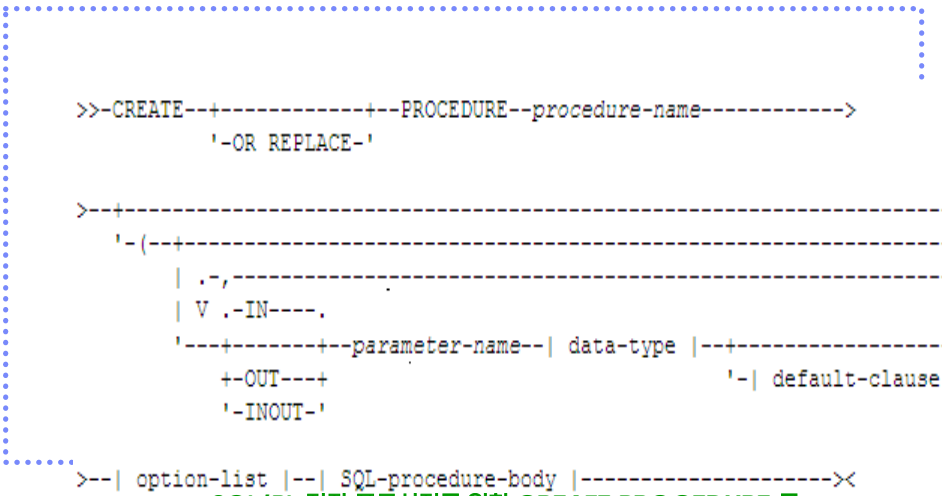


Figure 0833A SQL/PL 저장 프로시저를 위한 CREATE PROCEDURE 문

Tip

사용자 정의 함수의 <SPECIFIC명>은 <함수명>과 동일하게 지정할 수 있습니다. 옵션을 지정하지 않으면, SQLyymmddhhmmssxxx 형식으로 엔진이 자동 생성합니다.

2 옵션에 대한 설명은 다음과 같습니다.

모 드	설 명
<프로시저명>	프로시저의 이름을 지정합니다.
<인수 입출력 유형>	인수의 유형은 IN, OUT, INOUT 으로 지정합니다.
<인수 이름>	인수의 이름을 지정합니다.
<데이터유형>	인수의 데이터 유형을 지정합니다.
SPECIFIC	임의의 고유한 이름으로 지정합니다.
DYNAMIC RESULT SETS	반환할 결과 집합의 개수를 지정합니다.
CONTAINS SQL	데이터의 조회, 변경을 위한 SQL문을 포함하지 않습니다.
READS SQL DATA	데이터를 변경하는 SQL문을 포함할 수 없습니다.
MODIFIES SQL DATA	지원되는 모든 SQL문을 사용합니다.
DETERMINISTIC	동일한 입력 인수에 대해서 항상 동일한 값을 반환합니다.
CALLED ON NULL INPUT	입력 인수의 값이 NULL인 경우에도 호출됩니다.
LANGUAGE SQL	SQL/PL 로 작성한 SP입니다.
NO EXTERNAL ACTION	데이터베이스 시스템이 관리하지 않는 외부 오브젝트에 대한 상태를 변경시키는 로직의 포함 여부를 지정합니다.
<SQL 함수 본문>	SQL/PL 블록을 이용하여 구현합니다.

TOPIC 08 34 SQL/PL 프로시저

Point

SQL/PL 언어를 이용하여 저장 프로시저를 생성합니다. CALL 문을 이용하여 호출합니다.

Tip

SQL/PL에서 ; (세미콜론)은 문장의 구분자로 사용됩니다. create function 문의 끝을 구별할 때는 @ 또는 ! 등의 문자를 사용합니다.

1 SQL 저장 프로시저를 위한 로직을 작성하여 임의의 <파일명>으로 저장합니다.

```
$ cat <파일명>
CREATE PROCEDURE myproc (IN deptNumber CHAR(3),
                         OUT medianSalary DOUBLE)
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INTEGER;
  DECLARE SQLSTATE CHAR(5);
  DECLARE v_numRecords INT DEFAULT 1;
  DECLARE v_counter INT DEFAULT 0;

  DECLARE c1 CURSOR FOR
       SELECT CAST(salary AS DOUBLE) FROM employee
       WHERE workdept = deptNumber
       ORDER BY salary;
  DECLARE EXIT HANDLER FOR NOT FOUND
       SET medianSalary = 6666;

  SET medianSalary = 0;
  SELECT COUNT(*) INTO v_numRecords FROM employee
  WHERE workdept = deptNumber;
  OPEN c1;
  WHILE v_counter < (v_numRecords / 2 + 1) DO
       FETCH c1 INTO medianSalary;
       SET v_counter = v_counter + 1;
  END WHILE;
  CLOSE c1;
END @
```

2 create procedure 문을 이용하여 SQL 저장 프로시저를 생성하고, 등록합니다.

```
$ db2 connect to <데이터베이스명>
$ db2 -td@ -svf  <파일명>
```

Tip

SP를 호출하려면 EXECUTE 권한이 필요합니다. SP명은 대소문자를 구분하지 않습니다.

3 CALL 문을 이용하여 저장 프로시저를 호출합니다.

```
$ db2  "call myproc('A00',?)"
Value of output parameters
--------------------------
Parameter Name  : MEDIANSALARY
Parameter Value : +4.65000000000000E+004
Return Status = 0
```

Tip

CLP에서 SP를 호출하려면, OUT 유형의 인수값에는 출력용 변수명 대신에 ?(물음표)를 이용합니다.

TOPIC 08 35 PL/SQL 프로시저

Point

PL/SQL 언어를 이용하여 저장 프로시저를 생성합니다. CALL 문을 이용하여 호출합니다.

Tip

9.7 이후로는 PL/SQL을 이용하여 저장 프로시저를 생성할 수 있습니다.

1 PL/SQL을 이용하여 저장 프로시저를 생성하고자 하는 경우에는 Registry 변수에 compatibility vector를 '800'로 설정한 후, 데이터베이스를 생성합니다.

```
$ db2set compatibility Vector = 800
```

2 create procedure 문을 이용하여 SQL 저장 프로시저를 개발합니다.

```
$ cat script.db2
set sqlcompat plsql/

CREATE OR REPLACE FUNCTION emp_comp (
    p_sal          NUMBER,
    p_comm         NUMBER )
RETURN NUMBER
IS
BEGIN
   RETURN (p_sal + NVL(p_comm, 0)) * 24;
END emp_comp
/

CREATE OR REPLACE PROCEDURE update_comp(p_name IN
    VARCHAR) AS
BEGIN
   UPDATE emp SET tot_comp = emp_comp(salary, comm)
    WHERE name = p_name;
END update_comp
/
```

작성된 구문이 PL/SQL구문임을 알려주는 기능

Tip

작성된 스크립트에서 구분자가 '/' 또는 ';'인 경우에는 -td/-vf로 생성합니다.

3 DB2CLP 창에서 컴파일 및 생성합니다.

```
$ db2 –td/ -vf script.db2
```

4 CLP창에서 프로시저를 호출할 때에는 CALL을 사용합니다.

```
$ db2 call update_comp('Curly')
```

UNIT 09

데이터 이동

DB2는 데이터베이스 테이블에 저장된 데이터를 여러 가지 유형의 파일로 저장시키는 EXPORT 유틸리티를 제공합니다. 또한, 파일의 데이터를 테이블에 추가시키는 IMPORT와 LOAD 유틸리티를 제공합니다. LOAD 유틸리티는 대량의 데이터를 고속으로 처리하는데 사용됩니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 데이터 파일의 유형
- EXPORT 유틸리티
- EXPORT 명령어
- IMPORT 유틸리티
- IMPORT 명령어
- LOAD 유틸리티
- LOAD 명령어
- LOAD QUERY 명령어
- LOAD 단계
- BUILD 단계
- DELETE 단계
- 백업 보류 상태
- 점검 보류 상태
- LOAD 시나리오
- Cursor Load

TOPIC 09 01 데이터 파일의 유형

Point

데이블로부터 데이터를 파일로 저장하거나, 파일의 데이터를 추가 입력하는 유틸리티에서 사용되는 파일의 유형은 ASC, DEL, IXF 등이 있습니다. IXF는 데이터와 컬럼에 대한 메타정보를 가지고 있으므로 목표 테이블을 생성할 수도 있습니다.

Tip

Host DB2로부터 데이터 추출 시에는 IXF포맷을 지원합니다.

1 EXPORT, IMPORT, LOAD 명령어는 테이블과 파일간의 데이터 이동을 지원합니다.

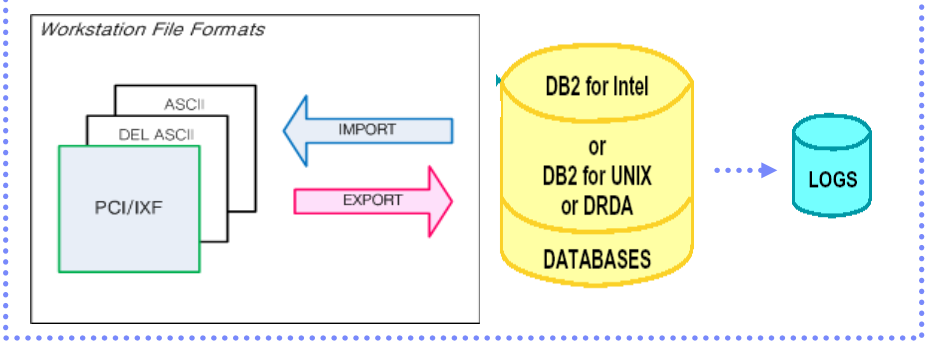


Figure 0901A 데이터 이동에 사용되는 파일의 유형

2 5가지 유형의 데이터 파일과 CURSOR를 지원합니다.

유 형	설 명
ASC	Non-delimited ASCii 파일로, IMPORT, LOAD 에 사용됩니다.
DEL	DELimited ascii 파일로, EXPORT, IMPORT, LOAD 에 사용됩니다.
WSF	DB2 9.7에서는 더 이상 지원하지 않습니다.
IXF	Integrated eXchange Format으로 EXPORT, IMPORT, LOAD 에 사용됩니다.
CURSOR	SELECT문의 결과 집합을 저장한 구조체로 LOAD에 사용됩니다.

Tip

데이터 파일의 유형과 확장자는 일치하지 않아도 됩니다.

3 ASC파일 유형은 에디터로 편집이 가능한 파일이며, 각 컬럼에 대응되는 데이터의 값들은 그 시작 바이트와 종료 바이트의 위치가 동일합니다.

```
$ cat <데이터 파일명>.asc
AAAA      312       2006-01-01
BB        4538      2006-02-01
```

4 DEL 파일 유형도 에디터로 편집이 가능한 파일이며, 각 컬럼에 대응되는 데이터의 값들은 , (컴마 부호)와 " (쌍따옴표 부호) 등의 구분자에 의해 구별됩니다.

```
$ cat <데이터 파일명>.del
"AAAA", 312,"2006-01-01"
"BB", 45,"2006-02-01"
```

Tip

CURSOR는 유틸리티가 실행되는 동안 엔진에 의해 임시로 메모리 또는 시스템 임시 테이블스페이스에 생성되는 파일과 유사한 개념입니다.

5 IXF 파일 유형은 데이터와 그 데이터에 대한 속성을 함께 가진 파일입니다. 에디터로 편집은 할 수 없으며, 새로운 테이블을 생성하고 데이터를 입력할 때 사용됩니다.

6 CURSOR 유형은 소스 테이블에 SQL 쿼리문을 이용하여 조건에 맞는 결과 집합을 추출하여 데이터 파일을 생성하지 않은 채로 LOAD 유틸리티의 입력으로 사용하는 방법입니다.

TOPIC 09 02 EXPORT 유틸리티

Point

EXPORT 명령어를 이용하여 테이블의 데이터를 파일로 저장할 수 있습니다. 지원되는 파일의 유형은 DEL, WSF, IXF 입니다. 원격 데이터베이스에 대해서도 실행할 수 있으며, 데이터 파일은 클라이언트에 생성됩니다.

1 EXPORT 명령어에서 데이터를 추출할 SELECT문과 출력 파일명을 지정하면, SELECT문의 실행으로 생성된 결과 집합을 지정한 출력 파일로 저장합니다.

```
EXPORT TO kes.del OF DEL
MESSAGES kes.msgs
SELECT * FROM kes.empl ;

EXPORT TO kes.wsf OF WSF
MESSAGES kes.msgs
SELECT * FROM artists;

EXPORT TO kes.ixf OF IXF
MESSAGES kes.msgs
SELECT * FROM artists;
```

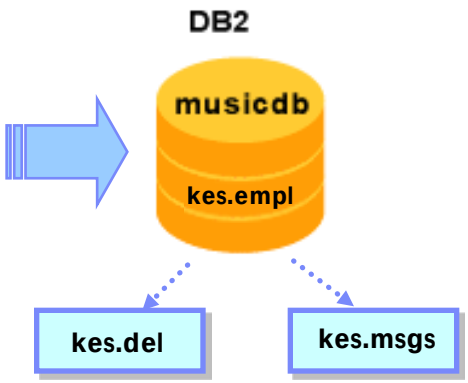


Figure 0902A DEL 유형의 파일로 데이터 내보내기

Tip

DB2 9.7이후 WSF형식은 더 이상 지원되지 않습니다.

2 데이터베이스에 접속하고 export 명령어로 DEL 유형의 파일에 데이터를 저장합니다.

```
$ db2 connect to <데이터베이스명>
$ db2 "export to <출력파일명> of del messages <메시지파일명> <select문>"
```

3 생성된 DEL 유형의 데이터 파일을 확인합니다.

```
$ cat <출력파일명>
```

Tip

메시지 파일에 기록된 오류 코드 중에서 SQL0012W, SQL0347W, SQL0360W, SQL0437W, SQL1824W 은 경고 메시지입니다. 다른 오류가 발생하면, 유틸리티는 실패합니다.

4 메시지 파일을 확인하여 오류가 있었는지 확인합니다.

```
$ cat <메시지파일명>
```

5 WSF 유형의 파일은 Lotus 1-2-3과 Symphony 제품이 사용하는 파일의 형식입니다.

```
$ db2 "export to <출력파일명> of ixf <SELECT문>"
```

Tip

255 글자 이상의 VARCHAR 또는 LOB 등의 LONG 유형의 컬럼이 포함된 경우에는 IXF 유형을 사용하도록 합니다.

6 IXF 유형의 파일로 데이터를 저장하면, 데이터와 컬럼의 속성 정보가 함께 저장되므로, 동일한 구조의 테이블을 생성하고 데이터도 함께 입력할 때 이용됩니다.

```
$ db2 "export to <출력파일명> of ixf <SELECT문>"
```

7 EXPORT 유틸리티는 ASC 유형의 출력 파일을 지원하지 않으므로 , CLP를 이용하여 원하는 SELECT문을 실행하고, 그 결과를 출력 파일로 저장하는 간접적인 방법을 사용합니다.

```
$ db2 -x -o "<SELECT문>" > <출력파일명>
```

TOPIC 09 03 EXPORT 명령어

Point

SELECT문을 이용하여 데이터를 조회하여 결과를 파일로 저장하는 명령어입니다. 지원되는 파일의 유형은 DEL, WSF, IXF 이며, 데이터 파일은 클라이언트에 생성됩니다.

Tip

SYSADM, DBADM 권한 또는 테이블에 대한 CONTROL, SELECT 특권을 가진 사용자가 실행합니다.

1 EXPORT 명령어의 형식은 다음과 같습니다.

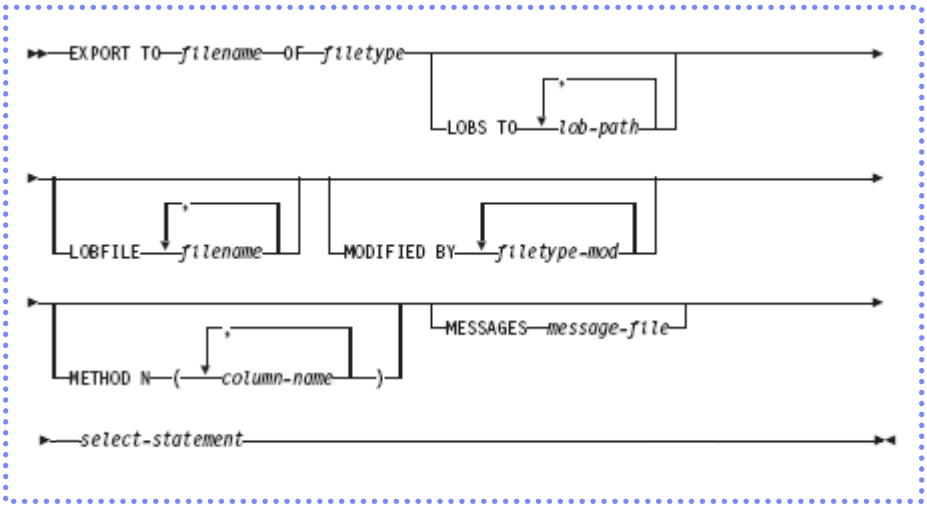


Figure 0903A ••• EXPORT 명령어

Tip

출력 파일의 유형은 OF 옵션으로 판별합니다. 입력 파일의 확장자는 무의미합니다.

Tip

METHOD N 옵션은 WSF, IXF 유형에서만 지원됩니다.

2 옵션에 대한 설명은 다음과 같습니다.

모 드	설 명
TO <파일명>	출력 데이터 파일명을 지정합니다.
OF <파일의 유형>	ASC, DEL, WSF, IXF 중의 한 가지를 지정합니다.
LOBS TO <디렉토리명>	LOB 컬럼이 저장될 디렉토리명을 지정합니다.
LOBFILE <파일명>	LOB 컬럼이 저장될 파일명을 지정합니다. 3자리 숫자의 확장자가 자동적으로 생성됩니다.
MODIFIED BY <파일형식수정자>	다양한 옵션으로 데이터 파일의 형식을 제어합니다.
METHOD N	컬럼의 이름으로 데이터를 지정합니다.
MESSAGES	작업 결과를 기록하는 메시지 파일명을 지정합니다.
<SELECT문>	데이터를 추출할 SELECT문을 지정합니다.

Tip

DB2 9.7이후 WSF형식은 더 이상 지원되지 않습니다.

3 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0008303.html 에서 MODIFIED BY 옵션에 사용되는 <파일형식수정자> 정보를 확인합니다.

TOPIC 09 04 IMPORT 유틸리티

Point

IMPORT 명령어를 이용하여 입력 파일의 데이터를 테이블에 추가할 수 있습니다. 지원되는 파일의 유형은 ASC, DEL, WSF, IXF 입니다. 원격 데이터베이스에 대해서도 실행할 수 있으며, 데이터 파일은 클라이언트에 있어야 됩니다.

Tip

INSERT, UPDATE, CREATE TABLE 등을 실행할 때와 동일하게 데이터베이스 로깅, 트리거 실행, 고유 인덱스 점검, 외부 키 점검, 컬럼 제약 조건 점검 등이 실행됩니다.

1 데이터 파일을 준비하여 IMPORT 명령어를 실행하면 파일의 데이터를 INSERT 문으로 테이블에 추가하거나 UPDATE 문으로 갱신합니다. IXF 유형을 사용하면 CREATE TABLE 문으로 테이블을 생성하고, INSERT 문으로 데이터를 추가합니다.

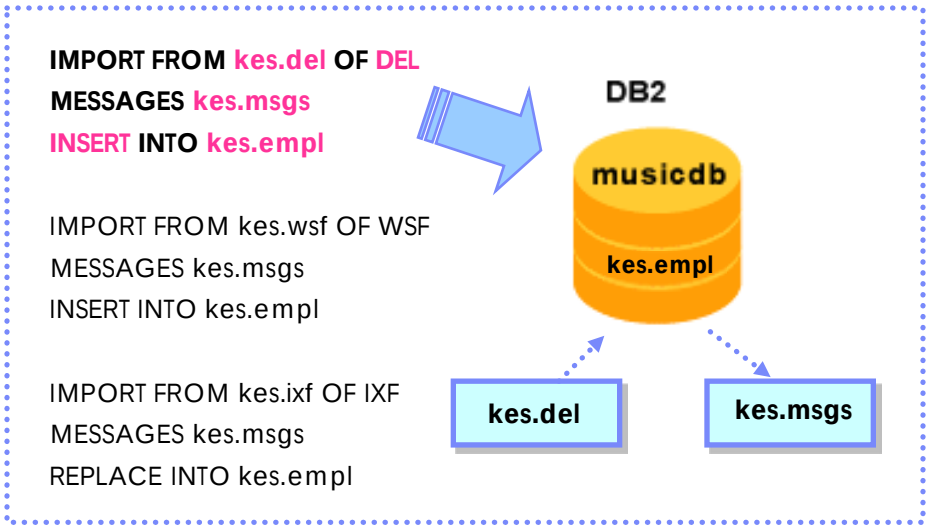


Figure 0904A • • • DEL 유형의 파일에서 데이터 가져오기

Tip

DB2 9.7이후 WSF형식은 더 이상 지원되지 않습니다.

2 ASC 유형의 입력 파일에서 데이터를 테이블로 저장합니다. 반드시 METHOD L 옵션을 이용하여 컬럼에 대응하는 필드의 위치를 지정해야 합니다.

```
$ db2 "import from <입력파일명> of asc METHOD L (<시작위치 1> <종료위치 1>, <시작위치 2> <종료위치 2>, ... , <시작위치 N> <종료위치 N>) messages <메시지파일명> <실행모드> into <목표테이블명>"
```

Tip

ASC, DEL 유형의 데이터 파일은 에디터를 이용하여 편집할 수 있습니다.

3 DEL 유형의 입력 파일에서 데이터를 테이블로 저장합니다.

```
$ db2 "import from <입력파일명> of del messages <메시지파일명> <실행모드> into <목표테이블명>"
```

Tip

Linux 또는 Unix에서 REPLACE 모드로 입력 파일의 이름을 /dev/null 로 지정하면 목표 테이블의 기존 데이터가 로깅 없이 삭제됩니다.

4 WSF 유형의 파일은 Lotus 1-2-3과 Symphony 제품이 사용하는 파일의 형식입니다. 목표 테이블에 데이터를 추가할 수 있습니다.

```
$ db2 "import from <입력파일명> of wsf messages <메시지파일명> <실행모드> into <목표테이블명>"
```

Tip

IMPORT 유틸리티가 실패하면, 마지막 COMMITCOUNT 지점으로 롤백됩니다. REPLACE 모드에서 COMMITCOUNT 옵션이 없다면, 모든 데이터는 삭제됩니다.

5 IXF 유형의 파일을 이용하면 목표 테이블을 생성하고 데이터를 추가할 수 있습니다. 목표 테이블이 이미 존재하면 데이터만 추가할 수도 있습니다.

```
$ db2 "import from <입력파일명> of ixf messages <메시지파일명> <실행모드> into <목표테이블명>"
```

TOPIC 09 05 IMPORT 명령어

Point

입력 파일의 데이터를 INSERT, UPDATE, CREATE TABLE 문을 이용하여 기존 또는 새로운 테이블로 저장하는 명령어입니다. ASC, DEL, WSF, IXF 유형이 지원되며, 데이터 파일은 클라이언트에 있어야 합니다.

Tip

SYSADM, DBADM 권한 또는 테이블에 대한 CONTROL, INSERT 특권을 가진 사용자가 실행합니다.

1 IMPORT 명령어의 형식은 다음과 같습니다.

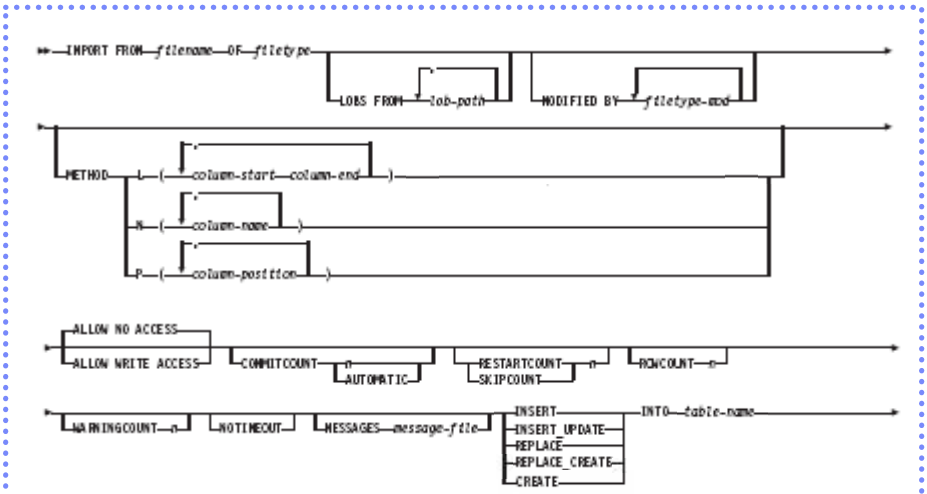


Figure 0905A IMPORT 명령어

Tip

입력 파일의 유형은 OF 옵션으로 판별합니다. 입력 파일의 확장자는 무의미합니다.

Tip

COMMITCOUNT 옵션으로 데이터가 큰 경우에 과다한 로그 파일 사용을 방지하는 것이 좋습니다.

Tip

RESTARTCOUNT n 옵션과 SKIPCOUNT n 옵션은 동일합니다.

Tip

RESTARTCOUNT m 옵션과 ROWCOUNT n 옵션을 함께 사용하면, m+1 번째부터 n 건의 데이터를 처리합니다.

Tip

INSERT_UPDATE 모드를 사용하려면, 반드시 목표 테이블에 기본 키가 정의되어 있어야 합니다.

Tip

CRETAE, CREATE_REPLACE 옵션은 IXF 유형에서 지원됩니다.

2 옵션에 대한 설명은 다음과 같습니다.

모 드	설 명
FROM <입력파일명>	입력 데이터 파일명을 지정합니다.
OF <파일의 유형>	ASC, DEL, WSF, IXF 중의 한 가지를 지정합니다.
METHOD L / N / P	필드의 시작과 종료 위치 바이트 옵셋, 컬럼의 이름, 필드 번호로 컬럼별 데이터를 지정합니다.
COMMITCOUNT	n 건을 처리할 때마다 COMMIT을 실행합니다.
RESTARTCOUNT n	n+1 번째 데이터부터 처리합니다.
ROWCOUNT n	n 건의 데이터만 처리합니다.
WARNINGCOUNT n	n 개 이상의 경고가 발생하면 처리를 중단합니다.
MESSAGES	작업 결과를 기록하는 메시지 파일명을 지정합니다.
INSERT	기존의 목표 테이블에 데이터를 추가합니다.
INSERT_UPDATE	기존의 목표 테이블에 데이터를 추가 또는 갱신합니다.
REPLACE	기존의 목표 테이블의 데이터를 로깅 없이 truncate한 후, INSERT문으로 데이터를 추가합니다.
CREATE	목표 테이블을 생성하고 데이터를 입력합니다.
CREATE_REPLACE	REPLACE 또는 CREATE 옵션과 동일합니다.
INTO <테이블명>	목표테이블명을 지정합니다.

3 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0008304.html 에서 MODIFIED BY 옵션에 사용되는 <파일형식수정자> 정보를 확인합니다.

TOPIC 0906 LOAD 유틸리티

Point

파일로부터 대량의 데이터를 테이블에 고속으로 저장하는 유틸리티입니다. load 명령어와 set integrity 명령어가 사용됩니다. 지원되는 파일의 유형은 ASC, DEL, IXF 이며, 데이터 파일은 서버 또는 클라이언트에 존재할 수 있습니다.

Tip

입력된 데이터는 데이터베이스 로그 파일에 기록되지 않고, 목표 테이블과 관련된 트리거도 실행되지 않습니다.

1 데이터 파일을 준비하여 LOAD 명령어를 실행하면, 입력된 데이터는 목표 테이블과 인덱스에 반영되고, 고유 인덱스를 위반한 데이터는 목표 테이블에 입력되지 않습니다. LOAD 명령어는 내부적으로 LOAD, BUILD, DELETE 과정을 실행합니다.

단 계	설 명
LOAD	입력 파일의 데이터를 테이블스페이스 컨테이너로 직접 복사합니다. 컬럼과 데이터 유형이 호환되지 않거나, NULL을 허용하지 않는 컬럼에 NULL 값을 제공한 행은 복사되지 않습니다. 인덱스에 대한 정보를 함께 수집합니다.
BUILD	LOAD 단계에서 수집한 인덱스 정보를 이용하여 기존의 인덱스를 갱신하거나 재 생성합니다.
DELETE	고유 인덱스를 위반한 데이터를 점검하여 목표 테이블에서 제거하여 미리 정의된 예외 테이블에 입력합니다.

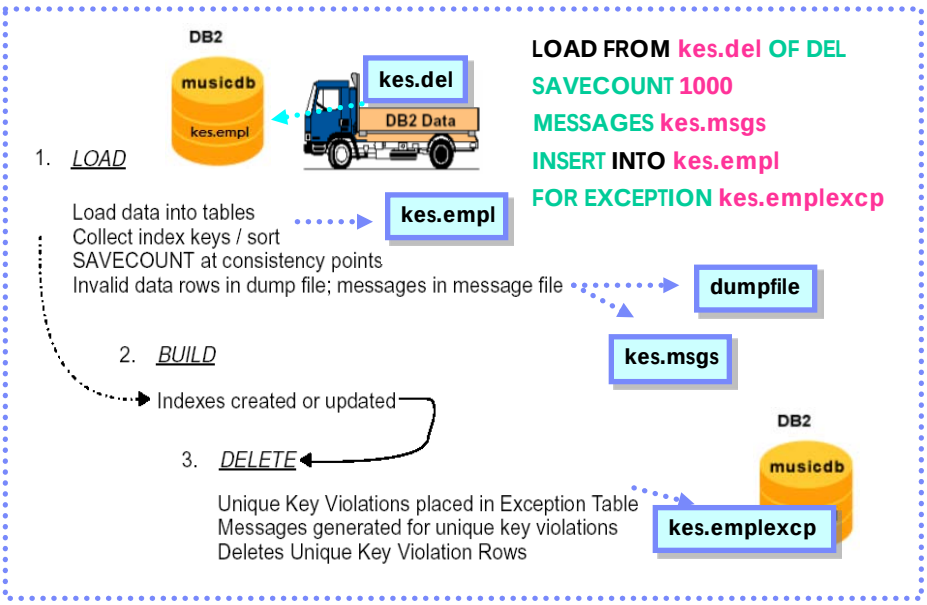


Figure 0906A LOAD 명령어의 3단계

Tip

예외 테이블을 사용하는 것을 권장합니다. 예외 테이블은 목표테이블과 다른 테이블스페이스에 생성하도록 합니다.

2 성공적으로 입력되지 못한 데이터는 예외 테이블에 저장합니다. 예외 테이블은 목표 테이블과 동일한 구조로 생성하며, 예외 데이터로 처리된 이유를 저장하기 위해 2개의 컬럼이 추가로 필요합니다.

```
$ db2 "create table <예외테이블> like <목표테이블>"
$ db2 "alter table <예외테이블> add column ts timestamp add column
    msg clob(32K)"
```

3 아카이브 로깅에서 load 명령어가 완료된 후에 목표 테이블이 속한 테이블스페이스는 '백업 보류 (Backup Pending)' 상태가 될 수 있습니다. backup db 명령어로 해결합니다.

4 load 명령어가 완료된 후에 목표 테이블에 외부키 또는 컬럼 제약 조건이 있으면, 목표 테이블은 '점검 보류 (Check Pending)' 상태가 됩니다. set integrity 명령어로 해결합니다.

TOPIC 0907 LOAD 명령어

Point

LOAD 유틸리티에서 사용되는 명령어로 데이터 LOAD, 인덱스 BUILD, 고유 인덱스 위반 행 DELETE 등의 작업을 처리합니다. ASC, DEL, IXF 유형의 데이터 파일을 지원하며, 데이터 파일은 서버 또는 클라이언트에 존재할 수 있습니다.

Tip
SYSADM, DBADM 또는 LOAD 권한과 테이블에 대한 INSERT, DELETE 특권이 필요합니다.

Tip
데이터의 일관성을 보장을 위해 기본적으로 목표 테이블에 배타적 잠금을 적용합니다.

Tip
OF CURSOR 옵션을 이용하면, CURSOR 를 선언하여 입력 파일을 대신합니다.

Tip
데이터 파일이 클라이언트에 있으면 CLIENT 옵션을 사용합니다.

Tip
입력 파일의 유형은 OF 옵션으로 판별합니다. 입력 파일의 확장자는 무의미합니다.

Tip
입력 데이터 파일이 큰 경우에는 SAVECOUNT 옵션을 사용하면, 복사 중에 실패해도 마지막 복사 완료 지점부터 계속 복사할 수 있습니다.

Tip
REPLACE 옵션을 사용한 LOAD 는 TERMINATE 옵션으로 종료했을 때, 목표 테이블의 모든 데이터는 truncate 되어 남아있지 않습니다.

Tip
고유 인덱스, 외부키, 컬럼 제약 조건 등을 위반한 행을 예외테이블에 저장하여 확인하는 것이 권장됩니다.

Tip
아카이브 로깅 모드를 사용할 때는 COPY 옵션을 지정하도록 합니다.

1 LOAD 명령어의 형식은 다음과 같습니다.

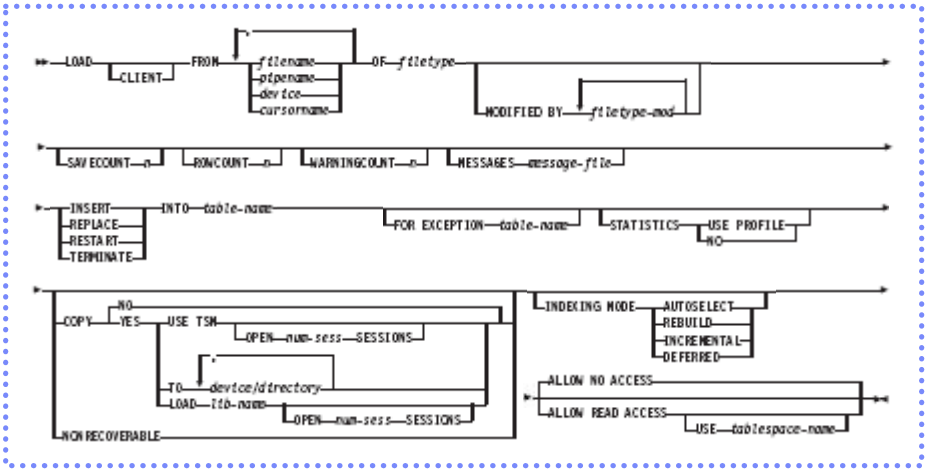


Figure 0907A LOAD 명령어

2 옵션에 대한 설명은 다음과 같습니다.

옵션	설명
CLIENT	클라이언트의 데이터 파일을 사용할 수 있습니다.
FROM <입력파일명>	입력 데이터 파일명을 지정합니다.
OF <파일의 유형>	ASC, DEL, IXF 중의 한 가지를 지정합니다.
METHOD L / N / P	데이터 파일의 필드를 구별하는 방법을 지정합니다.
SAVECOUNT n	n 건을 복사할 때마다 복사 완료 정보를 보관합니다.
ROWCOUNT n	n 건의 데이터만 처리합니다.
WARNINGCOUNT n	n 개 이상의 경고가 발생하면 처리를 중단합니다.
MESSAGES <파일명>	작업 결과를 기록하는 메시지 파일명을 지정합니다.
INSERT	목표 테이블에 데이터를 추가합니다.
REPLACE	목표 테이블의 데이터를 교체합니다.
RESTART / TERMINATE	중단된 LOAD 작업을 다시 시작하거나 종료합니다.
INTO <테이블명>	목표테이블명을 지정합니다.
FOR EXCEPTION <테이블명>	예외테이블명을 지정합니다.
COPY <모드>	백업 실행 여부를 지정합니다.
INDEXING MODE <모드>	인덱스 재생성 모드를 지정합니다.
ALLOW READ ACCESS	load 명령어가 실행되기 이전의 데이터에 대한 읽기 액세스를 허용합니다.

3 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0008305.html 에서 MODIFIED BY 옵션에 사용되는 <파일형식수정자> 정보를 확인합니다.

TOPIC 0908 LOAD QUERY 명령어

Point

로드 유틸리티가 로드 조작 중에 데이터베이스 일관성을 위해 변경하는 테이블의 상태 값을 확인하는 명령어로 LOAD 명령어의 현재 실행 단계 및 LOAD 단계에서 복사 완료된 데이터의 건수와 재생성 또는 갱신이 완료된 인덱스의 개수도 확인할 수 있습니다.

Tip

권한 또는 특권이 필요하지 않습니다.

1 LOAD QUERY 명령어의 형식은 다음과 같습니다.

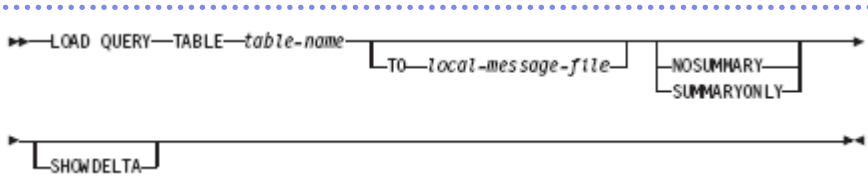


Figure 0908A ··· LOAD QUERY 명령어

2 옵션에 대한 설명은 다음과 같습니다.

옵션	설명
<테이블명>	load 명령어가 실행 중인 목표 테이블명을 지정합니다.
TO <메시지 파일명>	load 명령어가 현재까지 실행한 상황을 저장할 메시지 파일명을 지정합니다.
NOSUMMARY	load 명령어가 처리한 데이터 건수에 대한 요약 정보 (읽은 행 수, 건너뛴 행 수, 입력된 행 수, 거부된 행 수, 삭제된 행 수, 커미트된 행 수, 경고 개수)를 제공하지 않습니다.
SUMMARYONLY	load 명령어가 처리한 데이터 건수에 대한 요약 정보만 제공합니다.
SHOW DELTA	최근의 load query 명령어 실행 이후에 변경된 정보만 제공합니다.

Tip

load 명령어를 실행하고 있는 세션에서 실행하지 말고 새로운 세션에서 실행합니다.

3 데이터베이스에 접속한 후에 load query 명령어를 실행합니다.

```
$ db2 connect to <DB명>
$ db2 load query table <목표테이블명> TO <메시지파일명>
$ db2 load query table <목표테이블명> NOSUMMARY
$ db2 load query table <목표테이블명> SUMMARYONLY
$ db2 load query table <목표테이블명> summaryonly SHOWDELTA
```

```
$ db2 load query table kes.empl summaryonly

Number of rows read         - 453376
Number of rows skipped      - 0
Number of rows loaded       - 453376
Number of rows rejected     - 0
Number of rows deleted      - 0
Number of rows committed    - 408439
Number of warnings          - 0

Tablestate:
  Load in Progress
```

Figure 0908B ··· LOAD QUERY 명령어의 SUMMARYONLY 옵션

TOPIC 09-09 LOAD 단계

Point

LOAD 유틸리티의 첫 번째 과정으로 입력 파일의 데이터를 테이블스페이스 컨테이너로 직접 복사합니다. 컨테이너에 추가된 데이터는 데이터베이스 로그 파일에 기록되지 않으며, 데이터의 추가로 인한 INSERT 트리거도 실행되지 않습니다.

Tip

데이터는 SQL문을 이용해서 입력되는 것이 아니라, 디스크 수준에서 직접 복사됩니다.

Tip

MESSAGES 옵션을 이용하면, 특정 행이 유효하지 못한 데이터로 분류된 이유를 확인할 수 있습니다.

Tip

MODIFIED BY DUMPFILE 옵션을 이용하면 지정한 파일명으로 덤프 파일이 생성되어 유효하지 못한 데이터를 저장합니다. LOAD 작업이 완료된 후, 덤프 파일을 입력 파일로 이용하여 추가적인 LOAD 작업을 할 수 있습니다.

Tip

Not NULL 컬럼에 데이터파일로부터 값이 입력되지 않는 경우에는 Usedefaults 옵션을 사용하면 칼럼 Default 값이 자동 입력됩니다.

Tip

MODIFIED BY DUMPFILE 옵션은 ASC, DEL 유형의 입력 파일에만 사용할 수 있습니다. 덤프 파일명은 확장자를 한 개만 가질 수 있습니다.

1 입력 파일에 대해 다음과 같은 작업들을 실행합니다.

작 업	설 명
데이터 파일 확인	데이터베이스 서버의 지정한 디렉토리에서 입력 파일의 존재를 확인합니다. 로드 유틸리티를 실행하는 클라이언트에 데이터 파일이 있는 경우에는 CLIENT 옵션이 필요합니다.
데이터 복사	입력 파일의 데이터를 한 행씩 읽어들여 테이블스페이스 컨테이너로 직접 복사합니다. 데이터가 많은 경우에는 SAVECOUNT 옵션을 이용하여 내부적으로 복사 완료 중간 지점을 기록하게 하면, LOAD 유틸리티가 이 단계에서 실패한 경우에 마지막 복사 완료 중간 지점 이후부터 복사를 계속할 수 있습니다.
데이터 유형 점검	목표 컬럼의 데이터 유형과 대응되는 필드의 데이터 유형이 호환되지 못하는 행은 복사되지 않습니다. 유효하지 못한 데이터라고 표현합니다.
NULL 값 점검	NULL 값을 허용하지 않는 목표 컬럼에 NULL 값이 제공된 행은 복사되지 않습니다. 유효하지 못한 데이터로 처리하며, DUMPFILE 옵션을 이용하여 별도의 덤프 파일에 저장할 수 있습니다.
인덱스 정보 수집	목표 테이블에 존재하는 모든 인덱스에 대한 인덱스 키값과 대응되는 RID에 대한 정보를 미리 수집합니다.

2 목표 테이블에 데이터를 입력하는 모드는 2가지가 있습니다.

모 드	설 명
INSERT	목표 테이블에 입력 파일의 데이터를 추가합니다.
REPLACE	목표 테이블의 기존 데이터를 데이터베이스 로그 파일에 기록하지 않고 truncate한 후에 입력 파일의 데이터로 대체합니다.

3 load query table 명령어로 복사 작업의 진행 정도를 확인할 수 있습니다. 테이블스페이스의 상태는 '로드 진행 중 (Load in Progress)' 가 됩니다.

4 이 단계에서 실패하면, 목표 테이블은 '로드 보류 (Load Pending)' 상태가 됩니다. 로드 보류 상태는 다음의 2가지 모드를 이용하여 load 명령어를 다시 실행하여 해결합니다.

모 드	설 명
RESTART	LOAD 유틸리티를 계속합니다. SAVECOUNT 옵션을 사용했다면 마지막 복사 완료 중간 지점 이후부터 데이터 복사가 계속됩니다.
TERMINATE	LOAD 유틸리티를 취소합니다. INSERT 모드를 사용했다면 테이블은 load 명령어를 실행하기 이전의 상태로 복구되고, REPLACE 모드를 사용했다면 테이블의 데이터는 모두 삭제됩니다.

5 성공적으로 완료되면, 덤프 파일과 메시지 파일을 확인합니다. 유효하지 못한 데이터로 분류되어 테이블에 추가되지 못한 데이터와 그 원인을 확인할 수 있습니다.

```
$ cat <메시지파일명>
$ cat <덤프파일명>
```

TOPIC 09-10 BUILD 단계

Point

LOAD 유틸리티의 두 번째 과정으로 LOAD 단계에서 수집된 인덱스 정보를 이용하여 기존의 인덱스를 갱신하거나 재 생성합니다. LOAD 명령어에서 INDEXING MODE 옵션으로 제어합니다.

Tip

load 명령어를 완료한 후에 인덱스를 생성하는 것보다 인덱스를 미리 생성하여 load 명령어의 BUILD 단계에서 처리하는 것이 일반적으로 성능이 좋습니다.

Tip

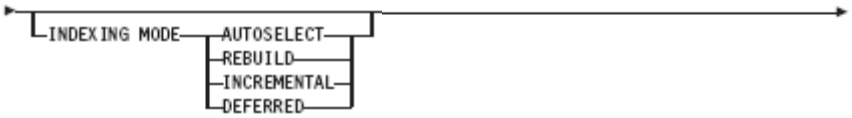
이 단계에서는 Index를 Build하기 위해 Temp공간을 사용합니다. Table의 크기에 비례하여, 그리고 동시에 수행되는 로드 작업 수를 감안하여 충분한 크기의 Temp공간을 미리 확보하십시오.

1 LOAD 단계에서 수집된 인덱스의 정보를 이용하여 새로 추가된 데이터를 인덱스에 반영하는 방법은 2 가지가 있습니다.

작 업	설 명
인덱스 갱신	목표 테이블에 존재하는 모든 인덱스에 새로 추가된 행에 대한 인덱스 데이터를 추가합니다.
인덱스 재생성	목표 테이블에 존재하는 모든 인덱스를 제거하고 다시 생성합니다.

2 인덱스의 재생성 여부는 4가지의 INDEXING MODE 옵션에 의해 결정됩니다.

작 업	설 명
AUTO	엔진이 UPDATE 와 REBUILD 모드 중에서 선택합니다.
UPDATE	목표 테이블에 존재하는 모든 인덱스에 추가된 데이터의 인덱스 정보를 추가합니다.
REBUILD	목표 테이블에 존재하는 모든 인덱스를 재 생성합니다.
DEFERRED	목표 테이블에 존재하는 모든 인덱스의 재생성 작업을 실행하지 않고 보류합니다. INDEXREC 데이터베이스 구성 변수의 값에 의해 재생성 시점이 결정됩니다.




```
load from kes.del of del messages kes.msgs insert into kes.empl
INDEXING MODE REBUILD ;
```

Figure 0910A LOAD 명령어의 INDEXING MODE 옵션

3 load query table 명령어로 인덱스 재생성 또는 갱신 작업의 진행 정도를 확인할 수 있습니다. 테이블스페이스의 상태는 '로드 진행 중 (Load in Progress)' 가 됩니다.

4 이 단계에서 실패하면, 목표 테이블은 '로드 보류 (Load Pending)' 상태가 됩니다. 로드 보류 상태는 다음의 2가지 모드를 이용하여 해결합니다.

모 드	설 명
RESTART	LOAD 유틸리티를 계속합니다. 마지막으로 실패한 인덱스의 재생성 또는 갱신 작업부터 다시 시작합니다.
TERMINATE	LOAD 유틸리티를 취소합니다. INSERT 모드를 사용했다면 테이블은 load 명령어를 실행하기 이전의 상태로 복구되고, REPLACE 모드를 사용했다면 테이블의 데이터는 모두 삭제됩니다.

5 성공적으로 완료되면, 메시지 파일을 확인합니다.

```
$ cat <메시지파일명>
```

TOPIC 09 11 DELETE 단계

Point

LOAD 유틸리티의 세 번째 과정으로 LOAD 단계에서 추가된 데이터 중에서 고유 인덱스의 규정을 위반한 중복 행들을 점검하여 예외 테이블로 입력합니다.

1 LOAD 단계에서 복사된 데이터에 대해서 다음과 같이 점검합니다.

작 업	설 명
기본키 확인	테이블에 일차키가 정의되어 있으면, 고유 인덱스가 존재합니다.
고유키 확인	테이블에 고유키가 정의되어 있으면, 고유 인덱스가 존재합니다.
고유 인덱스 확인	테이블에 일차키 또는 고유키가 정의되어 있지 않아도, 고유 인덱스가 존재할 수 있습니다.
고유 인덱스 위반 점검	LOAD 단계에서 복사된 데이터 중에서 고유 인덱스의 규정을 위반한 행을 점검합니다. 복사된 순서대로 점검하므로, 중복된 데이터는 최초의 한 건을 제외하고 모두 위반행으로 처리됩니다.
위반 데이터 제거	고유 인덱스 규정을 위반한 데이터를 테이블스페이스 컨테이너에서 제거합니다.
예외 테이블 입력	EXCEPTION 옵션으로 예외 테이블을 제공하면, 위반한 행은 미리 정의된 예외 테이블에 입력됩니다. 예외 테이블을 조회하여 데이터가 입력되지 못한 이유를 확인할 수 있습니다.

Tip

예외 테이블을 지정하지 않아도 이 단계는 실행되지만, 중복행에 대한 정보는 확인할 수 없습니다.

2 FOR EXCEPTION 옵션으로 고유 인덱스를 위반한 행을 저장할 예외 테이블명을 지정합니다.


```
load from kes.del of del messages kes.msgs insert into kes.empl
FOR EXCEPTION kes.emplexcp ;
```

Figure 0911A LOAD 명령어의 FOR EXCEPTION 옵션

3 load query table 명령어로 고유 인덱스 위반 행 점검 작업의 진행 정도를 확인할 수 있습니다. 테이블스페이스의 상태는 '삭제 진행 중 (Delete in Progress)' 가 됩니다.

4 이 단계에서 실패하면, 목표 테이블은 '삭제 보류 (Delete Pending)' 상태가 됩니다. 삭제 보류 상태는 RESTART 또는 TERMINATE 옵션으로 해결합니다.

모 드	설 명
RESTART	LOAD 유틸리티를 계속합니다. 마지막으로 실패한 DELETE 작업부터 다시 시작합니다.
TERMINATE	LOAD 유틸리티를 취소합니다. INSERT 모드를 사용했다면 테이블은 load 명령어를 실행하기 이전의 상태로 복구되고, REPLACE 모드를 사용했다면 테이블의 데이터는 모두 삭제됩니다.

5 성공적으로 완료되면, 예외 테이블을 이용하여 고유 인덱스를 위반한 행을 확인합니다.

```
$ db2 "select * from <예외테이블명>"
```

TOPIC 09 12 백업 보류 상태

Point

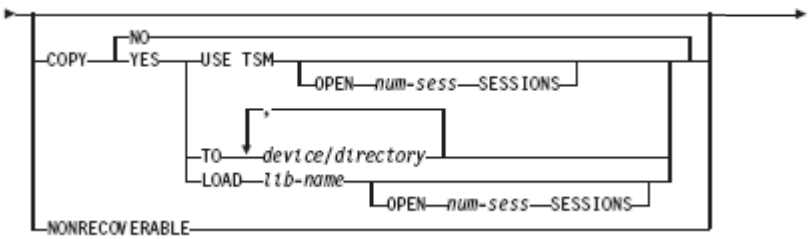
아카이브 로깅 모드에서 LOAD 명령어를 실행하면, 목표 테이블이 속한 테이블스페이스가 '백업 보류' 상태가 됩니다. LOAD 명령어의 COPY 옵션으로 LOAD 중에 백업 이미지를 생성하거나 BACKUP DB 명령어로 테이블스페이스의 백업을 실행합니다.

Tip

순환 로깅 모드를 사용했다면, '백업 보류 상태' 가 발생하지 않습니다.

1 load 명령어의 COPY 옵션으로 제어합니다. 기본 옵션은 COPY NO 입니다.

모 드	설 명
COPY YES	load 명령어를 실행하면서 추가된 데이터에 대한 백업 이미지를 생성합니다.
COPY NO	load 명령어를 실행하면서 추가된 데이터에 대한 백업 이미지를 생성하지 않으므로, load 명령어가 완료되면, 목표 테이블이 속한 테이블스페이스는 '백업 보류' 상태가 됩니다.
NONRECOVERABLE	로드 작업을 '복구 불가능' 상태로 표시합니다. 롤포워드 복구 시에 '복구 불가능'으로 표시된 로드 작업은 무시되고, 해당 시점 이후에 목표 테이블과 관련된 모든 트랜잭션은 무시됩니다. 롤포워드 복구가 완료되면, drop table 문으로 목표 테이블을 제거해야 합니다.




```
load from kes.del of del insert into kes.empl  COPY YES TO /back;
load from kes.del of del insert into kes.empl  COPY NO;
load from kes.del of del insert into kes.empl  ;
load from kes.del of del insert into kes.empl  NONRECOVERABLE;
```

Figure 0912A ... LOAD 명령어의 COPY 옵션

2 COPY YES 옵션을 지정하면, load 명령어를 실행하는 동안 추가된 데이터에 대한 백업 이미지를 생성하며, 완료한 후에 '백업 보류' 상태가 되지 않습니다.

3 COPY 옵션이 기본값인 NO 였다면, load 명령어를 완료한 후에 목표 테이블이 속한 테이블스페이스는 '백업 보류 (Backup Pending)' 상태가 됩니다. backup db 명령어에서 TABLESPACE 옵션을 이용하여 해당 테이블스페이스를 백업하여 해결합니다.

```
$ db2 backup db online tablespace <목표테이블스페이스명>
```

4 NONRECOVERABLE 옵션을 지정하면, load 명령어를 완료한 후에 '백업 보류' 상태가 되지 않습니다. ROLLFORWARD 복구시에 rollforward db 명령어로 로그 파일을 재적용할 때, 목표 테이블에 LOAD 유틸리티로 추가한 데이터는 복구가 불가능합니다.

TOPIC 09 13 점검 보류 상태

Point

LOAD 명령어는 테이블의 외부키와 점검 제한 조건을 점검하지 않으므로, LOAD 명령어가 완료된 후에 목표 테이블은 '점검 보류 (Check Pending)' 상태가 됩니다. 예외 테이블을 이용하여 SET INTEGRITY 문으로 해결합니다.

Tip

외부키 또는 점검에 대한 점검 제한 조건이 없는 목표테이블은 '점검 보류 상태' 가 되지 않습니다.

1 set integrity 문은 테이블의 외부키와 점검 제한 조건을 만족하지 않는 행을 검출하여 테이블에서 삭제하고 예외 테이블에 저장합니다.

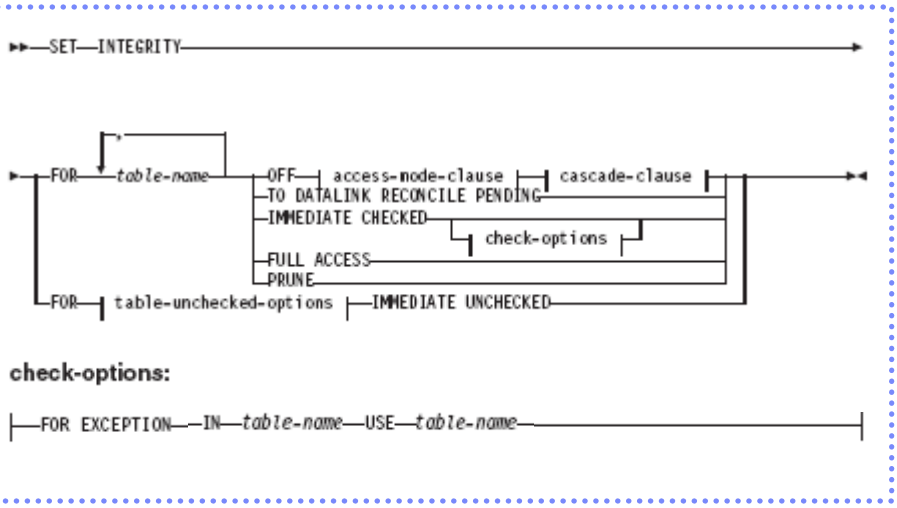


Figure 0913A SET INTEGRITY 문

2 옵션에 대한 설명은 다음과 같습니다.

모 드	설 명
FOR <테이블명>	목표 테이블명을 지정합니다.
IMMEDIATE CHECKED	즉시 점검하도록 요청합니다.
FOR EXCEPTION	예외 테이블이 사용되는 것을 알려줍니다.
FOR <테이블명>	예외 행이 검출된 테이블명입니다.
USE <테이블명>	예외 테이블명입니다.

Tip

예외 테이블은 일반적 용어이지 않고, load 명령어에서 사용하던 것을 그대로 사용하도록 합니다.

3 '점검 보류' 상태에 있는 테이블에 대해 set integrity 문을 실행합니다.

```
db2 "set integrity for <목표테이블명> immediate checked for exception
    in <목표테이블명> use <예외테이블명>"
```

4 예외 테이블을 이용하여 외부키와 점검 제한 조건을 위반한 행을 확인합니다.

```
$ db2 "select * from <예외테이블명>"
```

TOPIC 09 14 LOAD 시나리오

Point

LOAD 명령어로 입력 파일과 예외 테이블을 이용하여 실행합니다. '백업 보류' 상태에서는 BACKUP DB 명령어가 필요하며, '점검 보류' 상태에서는 SET INTEGRITY 명령어가 필요합니다. 실행 후에는 메시지 파일, 덤프 파일, 예외 테이블을 확인합니다.

Tip

load 명령어를 실행할 때는 목표 테이블이 반드시 존재해야 합니다.

Tip

인덱스를 따라 생성하는 것이 좋습니다. 시나리오에서는 alter table 문에 의하여 인덱스가 생성되었습니다.

Tip

kes.dept 테이블과 kes.empl 테이블은 RI 관계를 가지고 있습니다.

1 목표 테이블을 생성하고, 기본 데이터를 입력하는 SQL문을 작성하여, kes.sql 로 저장합니다.

```
$ vi kes.sql
DROP TABLE kes.empl;
DROP TABLE kes.dept;

CREATE TABLE kes.dept (
       id            smallint       not null
     , name          varchar(20)    not null
     , budget        int
) IN  ts01;

ALTER TABLE kes.dept ADD
      CONSTRAINT dept_pk01 PRIMARY KEY (id);

CREATE TABLE kes.empl (
       id            smallint       not null
     , name          varchar(30)    not null
     , sex           char(1)
     , mydept        smallint
     , salary        smallint
     , email         varchar(30)    not null
     , hiredate      date
) in ts02;

ALTER TABLE kes.empl ADD CONSTRAINT empl_pk01
      PRIMARY KEY(id);
ALTER TABLE kes.empl ADD CONSTRAINT empl_uk01
      UNIQUE (email);
ALTER TABLE kes.empl ADD CONSTRAINT empl_fk01
      FOREIGN KEY(mydept) REFERENCES kes.dept ;
ALTER TABLE kes.empl ADD CONSTRAINT empl_cc01
      CHECK (sex = 'M' or sex ='F');

INSERT INTO kes.dept VALUES
       (1,'총무팀',1000), (2,'기술지원팀', 2000) , (3,'POST팀',1500);
INSERT INTO kes.empl VALUES
       (1,'KES','F',1,100,'kes@kr.ibm.com','1993-01-30')
     , (2,'KHY','F',3,250,'khy@kr.ibm.com','1992-03-17')
     , (3,'JHS','F',2,300,'jhs@kr.ibm.com','1997-02-03')
     , (4,'JJY','M',2,280,'jjy@kr.ibm.com','1998-07-22');
```

2 CLP를 이용하여 kes.sql 파일의 SQL문을 실행합니다.

```
$ db2 connect to sample
$ db2 –tvf kes.sql
```

TOPIC 09 14 LOAD 시나리오

Point

LOAD 명령어로 입력 파일과 예외 테이블을 이용하여 실행합니다. '백업 보류' 상태에서는 BACKUP DB 명령어가 필요하며, '점검 보류' 상태에서는 SET INTEGRITY 명령어가 필요합니다. 실행 후에는 메시지 파일, 덤프 파일, 예외 테이블을 확인합니다.

Tip

1 행의 데이터는 문제가 없습니다.

Tip

2 행의 데이터는 첫 번째 칼럼의 값이 empl_pk01에 위반되므로 입력되지 않습니다. load 명령어의 DELETE 단계에서 예외 테이블인 kes.emplexcp 에 저장됩니다.

Tip

3 행의 데이터는 문제가 없습니다.

Tip

4 행에 있는 데이터는 여섯 번째 칼럼의 값이 empl_uk01에 위반되므로 입력되지 않습니다. load 명령어의 DELETE 단계에서 예외 테이블인 kes.emplexcp 에 저장됩니다.

Tip

5 행에 있는 데이터는 세 번째 칼럼의 값이 empl_cc01에 위반되므로 입력되지 않습니다. set integrity 문을 실행하면 예외 테이블인 kes.emplexcp 에 저장됩니다.

Tip

6 행의 데이터는 문제가 없습니다.

Tip

7 행의 데이터는 다섯 번째 칼럼의 값이 문자형이므로 컬럼의 데이터 유형과 호환되지 않습니다. NULL 값을 허용하는 컬럼이므로, NULL 값으로 변경되어 입력됩니다.

Tip

8 행의 데이터는 두 번째 칼럼의 값이 NULL 값이므로 입력되지 않고, 덤프 파일인 kes.dmp에 저장됩니다.

Tip

9 행에 있는 데이터는 네 번째 칼럼의 값이 empl_fk01에 위반되므로 입력되지 않습니다. set integrity 문을 실행하면 예외 테이블에 kes.emplexcp 에 저장됩니다.

Tip

10 행의 데이터는 첫 번째 칼럼의 값이 문자형이고, NULL 값으로 변경될 수 없으므로 입력되지 않고,덤프 파일인 kes.dmp에 저장됩니다.

3 입력 데이터를 준비하여 DEL 유형의 /work/kes.del 파일로 저장합니다.

```
$ vi /work/kes.del
11,"이문세","M", 1, 100, "lms@kr.ibm.com" , "2002-03-07"
11,"김경호","M", 2, 200, "kkh@kr.ibm.com", "2001-04-25"
13,"이기찬","M", 1, 300, "lkc@kr.ibm.com", "2002-02-19"
14,"김현정","F", 3, 400, "lkc@kr.ibm.com", "2002-07-17"
15,"김건모","m" ,2, 500, "kkm@kr.ibm.com", "2001-08-02"
16,"제이","F", 1, 120, "j@kr.ibm.com", "2000-05-08"
17,"양희은","F", 2, "130", "yhe@kr.ibm.com", "2002-10-20"
18,  ,"M", 2, 140, "god@kr.ibm.com", "2001-11-29"
19,"신화","M", 4 , 150, "sh@kr.ibm.com", "2001-04-07"
"20","엄정화","F", 1, 160, "ejw@kr.ibm.com", "2001-04-28"
```

4 추가되지 못한 행들을 확인하기 위해 예외 테이블인 kes.emplexcp를 미리 생성합니다.

```
$ db2 "CREATE TABLE kes.emplexcp LIKE kes.empl"
$ db2 "ALTER TABLE kes.emplexcp ADD COLUMN ts timestamp"
$ db2 "ALTER TABLE kes.emplexcp ADD COLUMN msg clob(32K)"
```

5 목표 테이블의 인덱스를 확인합니다.

```
$ db2 DESCRIBE INDEXES FOR TABLE kes.empl SHOW DETAIL
```

인덱스 스키마	인덱스 이름	규칙	고유한 컬럼수	컬럼 이름
KES	EMPL_PK01	P	1	+ID
KES	EMPL_UK01	U	1	+EMAIL

6 목표 테이블의 현재 데이터를 확인합니다.

```
$ db2 "SELECT id, name, mydept, sex, email FROM kes.empl"
```

ID	NAME	MYDEPT	SEX	EMAIL
1	KES	1	F	kes@kr.ibm.com
2	KHY	3	F	khy@kr.ibm.com
3	JHS	2	F	jhs@kr.ibm.com
4	JJY	2	M	jjy@kr.ibm.com

4 레코드가 선택됨.

```
$ db2 "SELECT id, name, mydept, sex, email FROM kes.emplexcp"
```

ID	NAME	MYDEPT	SEX	EMAIL

0 레코드가 선택됨.

TOPIC 09 14 LOAD 시나리오

Point

LOAD 명령어로 입력 파일과 예외 테이블을 이용하여 실행합니다. '백업 보류' 상태에서는 BACKUP DB 명령어가 필요하며, '점검 보류' 상태에서는 SET INTEGRITY 명령어가 필요합니다. 실행 후에는 메시지 파일, 덤프 파일, 예외 테이블을 확인합니다.

Tip

덤프 파일과 메시지 파일은 load 명령어가 실행되면서 생성됩니다.

7 kes.del 파일에서 kes.empl 테이블로 데이터를 저장할 load 명령어를 작성하여 kes.db2라는 파일에 저장합니다.

```
$ vi kes.db2
LOAD FROM /work/kes.del OF DEL
MODIFIED BY dumpfile=/work/kes.dmp
SAVECOUNT 10000
MESSAGES /work/kes.msgs
INSERT INTO kes.empl
FOR EXCEPTION kes.emplexcp;
```

8 CLP를 이용하여 kes.db2 파일의 load 명령어를 실행합니다.

```
$ db2 –stvf kes.db2
```

Tip

load query 명령어는 load 명령어가 실행되고 있는 세션이 아닌 새로운 세션에서 실행합니다.

Tip

OS shell 에서 while 문을 이용하여 일정한 시간 간격으로 load query 의 결과를 확인합니다.

9 다른 세션을 열고, load query 명령어를 이용하여 load 명령어의 진행 상태를 확인합니다.

```
$ db2 connect to sample
```

```
   데이터베이스 연결 정보

 데이터베이스 서버                        = DB2/AIX64 8.2.3
 SQL 권한 부여 ID                         = POST01
 로컬 데이터베이스 별명                   = SAMPLE
```

```
$ db2 load query table kes.empl summaryonly
```

```
------------------------------------------------------------------
SQL3532I  로드 유틸리티가 현재 "LOAD" 단계에 있습니다.

읽은 행 수                               = 10
건너뛴 행 수                             = 0
로드된 행 수                             = 8
거부된 행 수                             = 2
삭제된 행 수                             = 0
커미트된 행 수                           = 0
경고 수                                  = 3

테이블 상태:
  점검 보류
  로드 진행
------------------------------------------------------------------
SQL3532I  로드 유틸리티가 현재 "BUILD" 단계에 있습니다.
SQL3533I  로드 유틸리티가 현재 "2"의 인덱스 "2"을(를) 작성 중입니다.
------------------------------------------------------------------
SQL3532I  로드 유틸리티가 현재 "DELETE" 단계에 있습니다.
SQL3534I  로드 삭제 단계는 거의 "100"퍼센트 완료되었습니다.
------------------------------------------------------------------
SQL3532I  로드 유틸리티가 현재 "UNKNOWN" 단계에 있습니다.
테이블 상태:
  점검 보류
  로드 진행
```

TOPIC 09 14 LOAD 시나리오

Point

LOAD 명령어로 입력 파일과 예외 테이블을 이용하여 실행합니다. '백업 보류' 상태에서는 BACKUP DB 명령어가 필요하며, '점검 보류' 상태에서는 SET INTEGRITY 명령어가 필요합니다. 실행 후에는 메시지 파일, 덤프 파일, 예외 테이블을 확인합니다.

Tip

LOAD 단계에서는 10 개의 행 중에서 8 개의 행이 입력되었습니다.

- 7 행의 5번째 컬럼의 값이 NULL로 변환되어 입력되었습니다.
- 8 행의 두 번째 컬럼의 값이 누락되어 입력되지 못했습니다.
- 10 행의 첫 번째 컬럼의 값이 데이터 유형이 맞지 않고, NULL 값도 허용되지 않으므로 입력되지 못했습니다.
- 8 행과 10 행의 데이터는 덤프파일인 kes.dmp에 저장되었습니다.
- 거부된 행 수는 2건으로 8행과 10 행입니다.

Tip

BUILD 단계에서는 2 개의 인덱스가 다시 생성되었습니다.

- KES.EMPL_PK01
- KES.EMPL_UK01

Tip

DELETE 단계에서는 8 개의 행 중에서 2 개의 행이 삭제되었습니다.

- 2 행의 첫 번째 필드의 값이 empl_pk01에 위반되므로 삭제되었습니다.
- 4 행의 여섯 번째 필드의 값이 empl_uk01에 위반되므로 삭제되었습니다.
- 2 행과 4 행의 데이터는 예외테이블인 kes.emplexcp에 저장되었습니다.
- 삭제된 행수는 2건으로 2행과 4행입니다.

Tip

load 명령어로 입력된 행 수는 0건으로 1, 3, 5, 6, 7, 9 행의 데이터가 목표 테이블인 kes.empl 테이블에 입력되었습니다.

10 load 명령어가 완료되면, 메시지 파일을 확인하여 유효하지 못한 데이터로 분류되어 테이블에 추가되지 못한 데이터와 그 이유를 확인합니다.

```
$ cat kes.msgs

SQL3109N  유틸리티가 파일 "/data/post01/work/kes.del"에서 데이터를
          로드하기 시작합니다.

SQL3500W  유틸리티가 "04/22/2006 01:50:01.243151"에서 "LOAD" 단계를
          시작 중입니다.

SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "0".
SQL3520W  일관성 지점 로드에 성공했습니다.

SQL3117W  행 "F0-7", 컬럼 "5"의 필드 값을 SMALLINT 값으로 변환할 수
          없습니다.  널(NULL)이 로드되었습니다.
SQL3116W  행 "F0-8", 컬럼 "2"의 필드 값이 누락되었으나, 목표 컬럼이
          널(NULL) 입력 가능하지 않습니다.
SQL3185W  입력 파일의 "F0-8" 행에서 데이터를 처리하는 동안 이전의
          오류가 발생했습니다.
SQL3120W  행 "F0-10", 컬럼 "1"의 필드 값을 INTEGER 값으로 변환할 수
          는 없지만, 목표 컬럼이 널(NULL) 입력 가능하지 않습니다.
          행이 로드되지 않았습니다.
SQL3185W  입력 파일의 "F0-10" 행에서 데이터를 처리하는 동안 이전의
          오류가 발생했습니다.

SQL3227W  레코드 토큰 "F0-7"은(는) 사용자 레코드 번호 "7"을(를) 참조합
SQL3227W  레코드 토큰 "F0-8"은(는) 사용자 레코드 번호 "8"을(를) 참조합
SQL3227W  레코드 토큰 "F0-10"은(는) 사용자 레코드 번호 "10"을(를) 참조

SQL3110N  유틸리티가 처리를 완료했습니다. 입력 파일에서 "10"개의 행
          을 읽었습니다.
SQL3519W  일관성 지점 로드 시작. 입력 레코드 계수 = "10".
SQL3520W  일관성 지점 로드에 성공했습니다.
SQL3515W  유틸리티가 "04/22/2006 01:50:01.481270"에 "LOAD" 단계를 완료

SQL3500W  유틸리티가 "04/22/2006 01:50:01.493816"에서 "BUILD" 단계
          를 시작 중입니다.
SQL3213I  인덱싱 모드는 "REBUILD"입니다.
SQL3515W  유틸리티가 "04/22/2006 01:50:01.720362"에 "BUILD" 단계를
          완료했습니다.

SQL3500W  유틸리티가 "04/22/2006 01:50:02.192915"에서 "DELETE" 단계
          를 시작 중입니다.
SQL3509W  유틸리티가 테이블에서 "2"개의 행을 삭제했습니다.
SQL3515W  유틸리티가 "04/22/2006 01:50:02.261254"에 "DELETE" 단계를
          완료했습니다.

SQL3107W  메시지 파일에 적어도 하나의 경고 메시지가 있습니다.

읽은 행 수                                  = 10
건너뛴 행 수                                = 0
로드된 행 수                                = 8
거부된 행 수                                = 2
삭제된 행 수                                = 2
커미트된 행 수                              = 10
```

TOPIC 09-14 LOAD 시나리오

Point

LOAD 명령어로 입력 파일과 예외 테이블을 이용하여 실행합니다. '백업 보류' 상태에서는 BACKUP DB 명령어가 필요하며, '점검 보류' 상태에서는 SET INTEGRITY 명령어가 필요합니다. 실행 후에는 메시지 파일, 덤프 파일, 예외 테이블을 확인합니다.

Tip

load 명령어에서 거부된 경우 8 행과 10 행의 데이터는 덤프 파일인 kes.dmp에서 확인됩니다.

Tip

load 명령어에서 거부된 경우 2 행과 4 행의 데이터는 예외 테이블인 kes.emplexcp에서 확인됩니다.

Tip

12번 또는 13번은 load 명령어가 실패한 경우에만 실행합니다.

Tip

REPLACE 옵션을 지정한 경우에는 모든 데이터가 삭제됩니다.

11 덤프 파일을 확인하면 유효하지 못한 데이터가 저장된 것을 확인합니다.

```
$ cat kes.dmp.load.000
18,    ,"M", 2, 140, "god@kr.ibm.com", "2001-11-29"
"20","엄정화","F", 1, 160, "ejw@kr.ibm.com", "2001-04-28"
```

12 예외 테이블을 이용하여 고유 인덱스를 위반한 행이 입력되었는지 확인합니다.

```
$ db2 "SELECT SMALLINT(id) id, SUBSTR(name,1,6) name, mydept, sex,
  SUBSTR(email,1,15) email, SUBSTR(msg,1,30) msg FROM
  kes.emplexcp"

ID     NAME   MYDEPT SEX EMAIL           MSG
------ ------ ------ --- --------------- ------------------------------
    14 김현정      3 F   lkc@kr.ibm.com  00001I0000500002
    11 김경호      2 M   kkh@kr.ibm.com  00001I0000500001
2 레코드가 선택됨.
```

13 load 명령어를 실행 도중에 오류가 발생하면, RESTART 옵션으로 마지막 실패 지점 이후부터 계속할 수 있습니다. kes.db2 파일에서 INSERT옵션을 RESTART 옵션으로 변경하여 다시 실행합니다.

```
$ vi kes.db2
LOAD FROM /work/kes.del OF DEL
SAVECOUNT 10000
MODIFIED BY dumpfile /work/kes.dmp
MESSAGES /work/kes.msgs
RESTART INTO kes.empl
FOR EXCEPTION kes.emplexcp;
$ db2 –stvf kes.db2
```

14 load 명령어를 실행 도중에 오류가 발생하면, TERMINATE 옵션으로 load 명령어를 취소할 수 있습니다. kes.db2 파일에서 INSERT옵션을 TERMINATE 옵션으로 변경하여 다시 실행합니다. 테이블의 데이터는 load 명령어를 실행하기 전으로 복구됩니다.

```
$ vi kes.db2
LOAD FROM /work/kes.del OF DEL
SAVECOUNT 10000
MODIFIED BY dumpfile /work/kes.dmp
MESSAGES /work/kes.msgs
TERMINATE INTO kes.empl
FOR EXCEPTION kes.emplexcp;
$ db2 –svtf kes.db2
```

TOPIC 09 14 LOAD 시나리오

Point

LOAD 명령어로 입력 파일과 예외 테이블을 이용하여 실행합니다. '백업 보류' 상태에서는 BACKUP DB 명령어가 필요하며, '점검 보류' 상태에서는 SET INTEGRITY 명령어가 필요합니다. 실행 후에는 메시지 파일, 덤프 파일, 예외 테이블을 확인합니다.

Tip

아카이브 로깅 모드이고, 기존 백업 COPY NO 옵션이 사용된 경우에만 해당 테이블스페이스가 백업 보류 상태가 됩니다.

Tip

테이블스페이스가 정상 상태인 경우에는 백업이 필요하지 않습니다.

15 목표 테이블이 속한 테이블스페이스의 상태를 확인합니다.

```
$ db2 "SELECT SUBSTR(tbspace,1,18) tablespace FROM syscat.tables
  WHERE tabschema = 'KES' AND tabname = 'EMPL' "

TABLESPACE
------------------
TS02
1 레코드가 선택됨.
```

```
$ db2 LIST TABLESPACES

테이블 스페이스 ID                    = 4
이름                                  = TS02
유형                                  = 데이터베이스 관리 스페이스
내용                                  = 임의의 데이터
상태                                  = 0x0020
  세부사항 설명:
    백업 보류
```

16 '백업 보류' 상태를 해결하려면, BACKUP DB 명령어로 테이블스페이스 ts02를 백업합니다.

```
$ db2 "BACKUP DB sample TABLESPACE(ts02) ONLINE "
백업이 완료되었습니다. 이 백업 이미지에 대한 시간소인은 20060422011035
```

17 BACKUP DB 명령어를 실행한 세션의 데이터베이스 접속은 자동적으로 해제됩니다. 다시 데이터베이스에 접속합니다.

```
$ db2 CONNECT TO sample

   데이터베이스 연결 정보

데이터베이스 서버                    = DB2/AIX64 8.2.3
SQL 권한 부여 ID                     = POST01
로컬 데이터베이스 별명               = SAMPLE
```

18 목표 테이블의 데이터를 확인합니다.

```
$ db2 "SELECT * FROM kes.empl"

ID          NAME                          SEX MYDEPT SALARY EMAIL
----------- ----------------------------- --- ------ ------ ----------
SQL0668N  이유 코드 "1"(으)로 인해 테이블 "KES.EMPL"에서 조작이 허용되
않습니다.  SQLSTATE=57016
```

TOPIC 09 14 LOAD 시나리오

Point

LOAD 명령어로 입력 파일과 예외 테이블을 이용하여 실행합니다. '백업 보류' 상태에서는 BACKUP DB 명령어가 필요하며, '점검 보류' 상태에서는 SET INTEGRITY 명령어가 필요합니다. 실행 후에는 메시지 파일, 덤프 파일, 예외 테이블을 확인합니다.

Tip

외부키 또는 점검에 대한 점검 제한 조건이 있는 경우에만, 테이블이 '점검 보류' 상태가 됩니다.

Tip

SET INTEGRITY 명령어로 LOAD 명령어가 입력했던 6 개의 행 중에서 2 개의 행이 삭제되었습니다.

- 5 행의 세 번째 필드의 값이 empl_cc01에 위반되므로 삭제되었습니다.
- 9 행의 네 번째 필드의 값이 empl_fk01에 위반되므로 삭제되었습니다.
- 삭제된 행 수는 2건으로 5행과 9행입니다.
- kes.empl 테이블에 남은 행 수는 4 건으로 1, 3, 6, 7 행의 데이터가 목표 테이블인 kes.empl에 남았습니다.

Tip

SET INTEGRITY 명령으로 삭제된 행인 5행과 9행의 데이터가 예외 테이블인 kes.emplexcp에 추가되었습니다. LOAD 명령어에서 삭제된 행인 2행과 4행의 데이터를 포함해 총 4건의 데이터가 예외 테이블인 kes.emplexcp에서 확인됩니다.

19 SQL0668N 은 '점검 보류' 상태를 의미합니다. 정확한 에러 메시지를 확인합니다.

```
$ db2 "? SQL0668N"
SQL0668N 이유 코드 "<reason-code>"(으)로 인해 테이블
         "<table-name>"에서 조작이 허용되지 않습니다.
설명:
 테이블 "<table-name>"에 대한 액세스가 제한됩니다.  원인은
"<reason-code>"에 기초합니다.
1 테이블이 점검 보류 상태입니다.  테이블의 무결성이 적용되지
않았으므로 테이블 내용이 유효하지 않을 수 있습니다.  종속
테이블이 점검 보류 상태이면 점검 보류 상태가 아닌 상위 테이블이나
기본 테이블에서 조작을 수행할 경우에도 이 오류를 수신할 수
있습니다.
```

20 시스템 카탈로그에서 테이블에 대한 정보를 확인하면, 테이블에 대한 제한 조건이 점검되지 않았다는 것을 확인할 수 있습니다.

```
$ db2 "SELECT const_checked FROM syscat.tables WHERE
  tabschema = 'KES' AND tabname = 'EMPL' "
CONST_CHECKED
--------------------------------
NNYYYYYYYYYYYYYYYYYYYYYYYYYYYY
1 레코드가 선택됨.
```

21 '점검 보류' 상태를 해결하려면, SET INTEGRITY 명령어로 외부키와 점검 제한 조건을 위반한 행을 점검합니다. 위반한 행은 예외 테이블에 저장됩니다.

```
$ db2 "SET INTEGRITY FOR kes.empl IMMEDIATE CHECKED FOR
  EXCEPTION IN kes.empl USE kes.emplexcp"
SQL3602W  데이터 점검 처리시 제한조건 위반이 발견되어 예외 테이블로
이동시켰습니다.  SQLSTATE=01603
```

22 예외 테이블을 이용하여 외부키와 점검 제한 조건을 위반한 행이 입력되었는지 확인합니다.

```
$ db2 "SELECT SMALLINT(id) id, SUBSTR(name,1,6) name, mydept, sex,
  SUBSTR(email,1,15) email, SUBSTR(msg,1,30) msg FROM
  kes.emplexcp"
ID     NAME   MYDEPT SEX EMAIL           MSG
------ ------ ------ --- --------------- ------------------------------
    14 김현정      3 F   lkc@kr.ibm.com  00001I0000500002
    11 김경호      2 M   kkh@kr.ibm.com  00001I0000500001
    15 김건모      2 m   kkm@kr.ibm.com  00001K00018KES.EMPL.EMPL_CC01
    19 신화        4 M   sh@kr.ibm.com   00001F00018KES.EMPL.EMPL_FK01
4 레코드가 선택됨.
```

TOPIC 09 14 LOAD 시나리오

Point

LOAD 명령어로 입력 파일과 예외 테이블을 이용하여 실행합니다. '백업 보류' 상태에서는 BACKUP DB 명령어가 필요하며, '점검 보류' 상태에서는 SET INTEGRITY 명령어가 필요합니다. 실행 후에는 메시지 파일, 덤프 파일, 예외 테이블을 확인합니다.

Tip

kes.del 파일에 있던 10 건의 데이터 중에서 1, 3, 6, 7 행의 데이터만 입력되었습니다. kes.empl 테이블에 원래 있던 4 건을 포함하여 8 건의 데이터가 목표 테이블인 kes.empl 에서 확인됩니다.

23 목표 테이블의 최종 데이터를 확인합니다.

```
$ db2 "SELECT id, name, mydept, sex, email FROM kes.empl ORDER
  BY id"

ID          NAME                           MYDEPT SEX EMAIL
----------- ------------------------------ ------ --- ----------------
          1 KES                                 1 F   kes@kr.ibm.com
          2 KHY                                 3 F   khy@kr.ibm.com
          3 JHS                                 2 F   jhs@kr.ibm.com
          4 JJY                                 2 M   jjy@kr.ibm.com
         11 이문세                              1 M   lms@kr.ibm.com
         13 이기찬                              1 M   lkc@kr.ibm.com
         16 제이                                1 F   j@kr.ibm.com
         17 양희은                              2 F   yhe@kr.ibm.com
8 레코드가 선택됨.
```

TOPIC 09 15 Cursor Load

Point

사용자 정의 커서로부터 데이터를 입력 받아 로드하는 기능을 Cursor Load라고 합니다. 데이터 이동 적재를 위해 중간 SAM 파일을 생성하지 않고 직접 로드할 수 있어 사용자 편의성을 제공합니다.

Tip

SELECT구문은 Join을 포함하여 어떤 종류의 조회 문도 사용될 수 있습니다. 테이블도 동일 데이터베이스 내 테이블이나 또는 다른 데이터베이스에 Federation된 Nickname 오브젝트도 사용 가능합니다.

Tip

하나의 작업(cursorload.sql)에서는 100개까지 Cursor를 선언할 수 있으며, 임의의 이름을 부여할 수 있습니다.

Tip

이기종 데이터의 Federation을 위해Infosphere Federation Server 제품을 설치하십시오.

Tip

Federation구성 방법은 Infosphere Federation Server 매뉴얼을 참조하십시오.

1 먼저 사용자 정의 커서를 선언한 후, Cursor로부터 데이터를 로드합니다.

```
$cat cursorload.sql

DECLARE cur1 CURSOR for select * from  ORA10.CUSTOMER WITH UR;
LOAD FROM cur1 OF CURSOR INSERT INTO DB2.CUSTOMER;
DECLARE cur2 CURSOR for select * from  ORA10.ORDERS WITH UR;
LOAD FROM cur2 OF CURSOR INSERT INTO DB2.ORDERS;
DECLARE cur3 CURSOR for select * from  ORA10.CUSTOMER WITH UR;
LOAD FROM cur3 OF CURSOR INSERT INTO DB2.CUSTOMER;
DECLARE cur4 CURSOR for select * from  ORA10.CUSTOMER WITH UR;
LOAD FROM cur4 OF CURSOR INSERT INTO DB2.CUSTOMER;

$db2 –stvf cursorload.sql
```

2 이 기종 데이터베이스로부터 데이터를 이관하고자 하는 경우에는 Federation을 구성하고, 미리 사용될 테이블들에 대해 아래와 같이 Nickname을 정의합니다.

```
$cat createnick.sql

CREATE NICKNAME ORA10.CUSTOMER for ora10.DBOWN.CUSTOMER;
CREATE NICKNAME ORA10.ORDER for ora10.DBOWN.ORDERS;
CREATE NICKNAME ORA10.PRODUCT for ora10.DBOWN.PRODUCT;
CREATE NICKNAME ORA10.LINEITEM for ora10.DBOWN.LINEITEM;

$db2 –stvf createnick.sql
```

UNIT 10

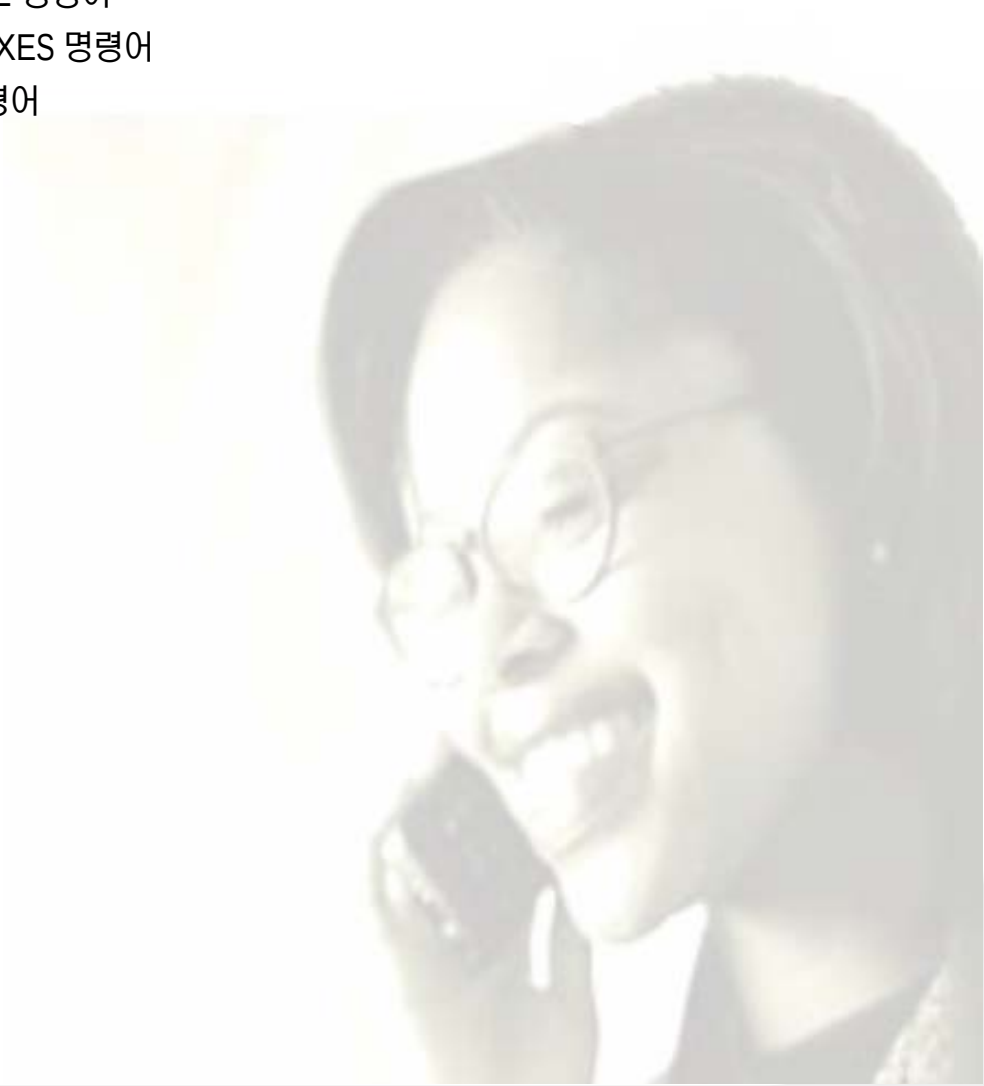
데이터베이스 운영

데이터베이스 시스템을 운영할 때는 업무 유형에 따라 일정한 주기로 REORGCHK, REORG, RUNSTATS등의 명령어를 실행하여, 테이블과 인덱스의 현재 저장 상태를 최적화하고, REBIND 등의 명령어를 실행하여 항상 최적의 액세스 플랜을 유지하는 것이 중요합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 일반적인 운영 방법
- REORGCHK 명령어
- 재구성 필요 여부 판별 방법
- REORG TABLE 명령어
- REORG INDEXES 명령어
- RUNSTATS 명령어
- 액세스 플랜 갱신
- REBIND 명령어



TOPIC 10 01 일반적인 운영 방법

Point

데이터베이스 시스템을 운영할 때는 일정한 주기로 RUNSTATS, REORGCHK, REORG, REBIND 등의 명령어들을 실행하여 항상 효율적인 액세스 플랜이 적용될 수 있도록 하는 것이 중요합니다.

Tip

인덱스 추가 또는 대량의 데이터 로드 등의 작업 후에는 RUNSTATS 명령어로 통계 자료를 갱신하고, REBIND 명령어로 패키지를 갱신하는 것이 권장됩니다.

1 REORGCHK, REORG TABLE, REORG INDEXES, RUNSTATS, REBIND 등의 유틸리티를 이용하여 데이터베이스의 구조가 최적화되었는지 정기적으로 점검합니다.

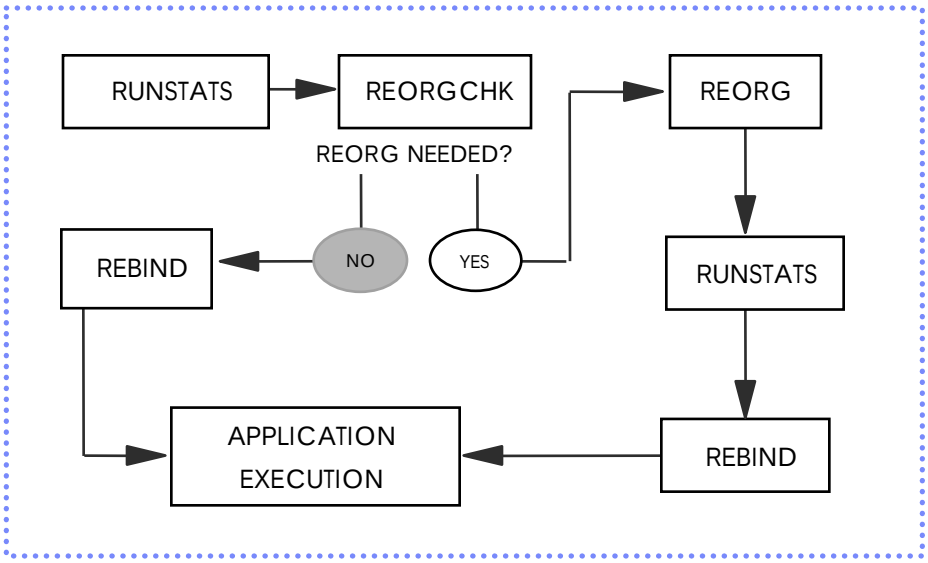


Figure 1001A 운영 유틸리티를 이용한 데이터 운영

2 DB2 사용자로 로그인하여 데이터베이스에 접속합니다.

```
$ login <DB2 사용자>
$ db2 connect to <데이터베이스명>
```

Tip

테이블명 또는 인덱스명을 명시할 때는 <스키마명>.<오브젝트명> 의 형태를 사용하도록 합니다.

3 REORGCHK 명령어는 테이블과 인덱스의 재구성 필요 여부를 판별합니다.

```
$ db2 reorgchk on table <스키마명>.<테이블명>
```

4 재구성이 필요하다고 판별되면 REORG TABLE 명령어와 REORG INDEX 명령어를 이용하여 테이블과 인덱스를 재구성합니다.

```
$ db2 reorg table <스키마명>.<테이블명>
$ db2 reorg index <스키마명>.<인덱스명>
```

5 REORG 명령어를 실행한 후에는 반드시 RUNSTATS 명령을 이용하여 시스템 카탈로그의 통계 자료를 갱신합니다.

```
$ db2 runstats on table <스키마명>.<테이블명> and indexes all
```

6 시스템의 통계 자료가 갱신되면, 더 적합한 액세스 플랜이 생성되도록 모든 패키지를 갱신합니다.

```
$ db2rbind <데이터베이스명> /l  <로그파일명> all
```

TOPIC 10 02 REORGCHK 명령어

Point

REORGCHK 명령어는 8가지의 판별식을 사용하여 테이블과 인덱스의 재구성 필요 여부를 판단합니다. 각 판별식을 적용한 결과가 기준치를 벗어난 경우에는 * 가 표시되어 테이블 또는 인덱스의 재구성이 필요함을 알려줍니다.

Tip

SYSADM, DBADM 권한 또는 테이블에 대한 CONTROL 특권을 가진 사용자가 실행합니다.

Tip

REORGCHK 를 실행하기 전에는 RUNSTATS를 실행하는 것이 권장됩니다. CURRENT STATISTICS 옵션은 최근에 RUNSTATS를 이미 실행한 경우에만 사용하도록 합니다.

1 REORGCHK 명령어는 8가지 판별식으로 테이블과 인덱스의 재구성 필요 여부를 판별합니다.

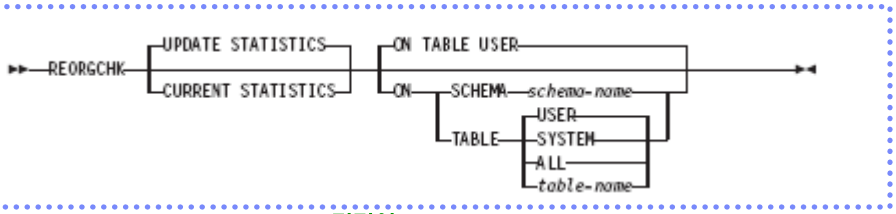


Figure 1002A ••• REORGCHK 명령어

2 현재의 통계 자료를 유지한 상태로 재구성 필요 여부를 점검합니다. 옵션을 지정하지 않으면, 기본값인 UPDATE STATISTICS 가 적용되므로 자동으로 RUNSTATS 명령어가 실행됩니다.

```
$ db2 reorgchk CURRENT STATISTICS on table <스키마명>.<테이블명>
```

3 특정한 <스키마명>을 가지는 테이블들에 대해서 재구성 필요 여부를 점검합니다.

```
$ db2 reorgchk ON SCHEMA  <스키마명>
```

4 시스템 카탈로그 테이블에 대해서만 재구성 필요 여부를 점검합니다.

```
$ db2 reorgchk ON TABLE SYSTEM
```

5 모든 테이블에 대해서 재구성 필요 여부를 점검합니다.

```
$ db2 reorgchk ON TABLE ALL
```

6 특정 테이블에 대해서 재구성 필요 여부를 점검합니다.

```
$ db2 reorgchk ON TABLE <스키마명>.<테이블명>
```

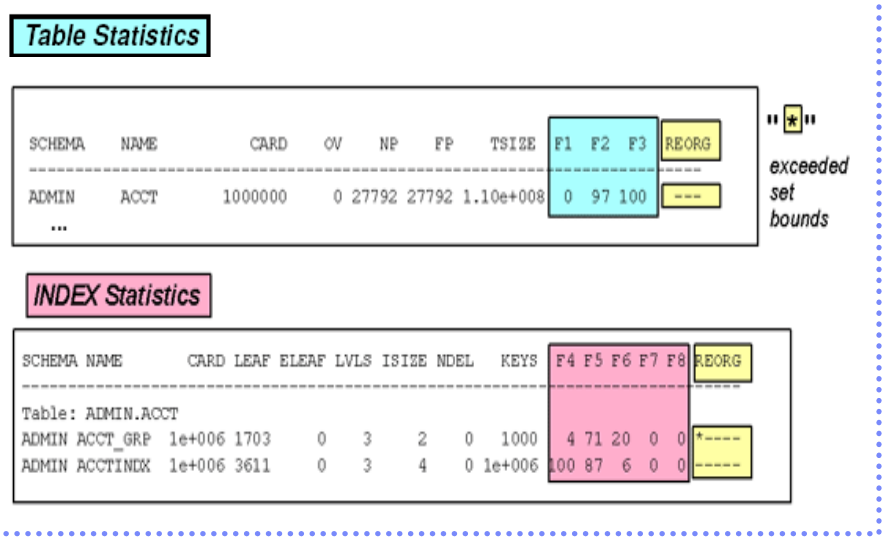


Figure 1002B ••• REORGCHK 명령어의 실행 결과

TOPIC 10 03 재구성 필요 여부 판별 방법

Point

총 8가지의 REORGCHK 판별식 중에서 F1~F3 까지의 판별식은 테이블의 재구성 여부를 판별하는데 사용되며, F4~F8 까지의 판별식은 각 인덱스에 대한 재구성 여부를 판별하는데 사용됩니다.

Tip

- 재구성을 실행하더라도 모든 인덱스에 대한 클러스터링 비율이 좋아질 수는 없으므로, 가장 중요한 인덱스를 기준으로 재구성하는 것이 좋습니다.

1 테이블에 대한 3가지 판별식을 이용하여 테이블의 재구성 필요 여부를 판별합니다.

Table Statistics:

F1: 100*OVERFLOW/CARD < 5
F2: 100*TSIZE / ((FPAGES-NPARTITIONS) * (TABLEPAGESIZE-68)) > 70
MDC 테이블) 100*TSIZE / ((ACTBLK-FULLKEYCARD) * EXTENTSIZE * (TABLEPAGESIZE-68)) > 70
F3: 100*NPAGES/FPAGES > 80
MDC 테이블) 100 * activeblocks / ((fpages / ExtentSize) - 1)

Figure 1003A ••• 테이블 재구성 필요 여부 판별식

판별식	설 명
F1	오버플로우된 행의 건수는 전체 건수의 5% 미만이어야 합니다.
F2	할당된 전체 공간에서 비어있는 공간의 총합은 30% 이하이어야 합니다.
F3	완전히 비어있는 페이지의 수는 전체 페이지 수의 20% 이하이어야 합니다.

2 인덱스에 대한 5가지 판별식을 이용하여 인덱스의 재구성 필요 여부를 판별합니다.

Index Statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
다중파티션), AVGPARTITION_CLUSTERRATIO or normalized AVGPARTITION _CLUSTERFACTOR > 80
F5: 100*(KEYS*(LEAF_RECSIZE+LEAF_RECSIZE_OVERHEAD)+ (INDCARD-KEYS)*DUPKEYSIZE) / ((NLEAF-NUM_EMPTY_LEAFS-1)* (INDEXPAGESIZE-LEAF_PAGE_OVERHEAD)) > MIN(50,(100 - PCTFREE))
다중 파티션)100*(KEYS*(LEAF_RECSIZE+LEAF_RECSIZE_OVERHEAD)+ (INDCARD-KEYS)*DUPKEYSIZE) / ((NLEAF-NUM_EMPTY_LEAFS - NPARTITIONS)* (INDEXPAGESIZE-LEAF_PAGE_OVERHEAD)) > MIN(50,(100 - PCTFREE))
F6: (100-PCTFREE) * ((FLOOR((100 - LEVEL2PCTFREE) / 100 * (INDEXPAGESIZE - NLEAF_PAGE_OVERHEAD)/(NLEAF_RECSIZE + NLEAF_RECSIZE_OVERHEAD)))* (FLOOR((100 - MIN(10, LEVEL2PCTFREE))/100*(INDEXPAGESIZE - NLEAF_PAGE_OVERHEAD)/ (NLEAF_RECSIZE + NLEAF_RECSIZE_OVERHEAD)) ** (NLEVELS - 3)) * (INDEXPAGESIZE - LEAF_PAGE_OVERHEAD))/(KEYS*(LEAF_RECSIZE+ LEAF_RECSIZE_OVERHEAD)+ (INDCARD - KEYS) * DUPKEYSIZE)) < 100
F7: 100 * (NUMRIDS_DELETED / (NUMRIDS_DELETED + INDCARD)) < 20
F8: 100 * (NUM_EMPTY_LEAFS/NLEAF) < 20

Figure 1003B ••• 인덱스 재구성 필요 여부 판별식

판별식	설 명
F4	인덱스의 클러스터링 비율은 80% 를 초과해야 합니다.
F5	인덱스에 할당된 공간에서 비어 있는 공간의 총합은 50% 이하이어야 합니다.
F6	재구성 후에 인덱스 트리의 레벨이 한 단계 낮아질 수 있는지를 판별합니다.
F7	삭제된 키가 존재하는 페이지(non-pseudo-empty pages)에서 삭제된 키(pseudo-deleted RIDs)가 차지하는 총 공간은 20% 미만이어야 합니다.
F8	전체 리프 페이지 중에서 완전히 비어 있는 리프 페이지(pseudo-empty leaf pages)의 비율은 20% 미만이어야 합니다.

TOPIC 10 04 REORG TABLE 명령어

Point

REORGCHK 수행 결과 테이블 재구성을 권장 받으면, REORG TABLE 명령어를 이용하여 테이블의 데이터가 연속적인 페이지에 저장될 수 있도록 테이블을 재구성합니다. Inplace옵션을 사용하면 Online 모드로 수행됩니다.

Tip

SYSADM, SYSCTRL, SYSMAINT, DBADM 권한 또는 테이블에 대한 CONTROL 특권을 가진 사용자가 실행합니다.

Tip

REORG TABLE 을 실행한 후에는 RUNSTATS, REBIND 를 실행하는 것이 권장됩니다.

Tip

작업 시에는 현재 데이터의 약 2.5배의 공간이 필요하므로 USE 옵션을 이용하여 충분한 크기의 임시 테이블 공간을 확보하는 것이 좋습니다.

Tip

임시 테이블스페이스의 페이지 크기는 재구성하는 테이블과 동일해야 합니다.

Tip

INPLACE 옵션을 사용하면, 임시 테이블 공간을 지정할 수 없습니다.

1 REORG TABLE 명령어로 테이블의 데이터를 물리적으로 재배치하고, 인덱스를 재 생성합니다.

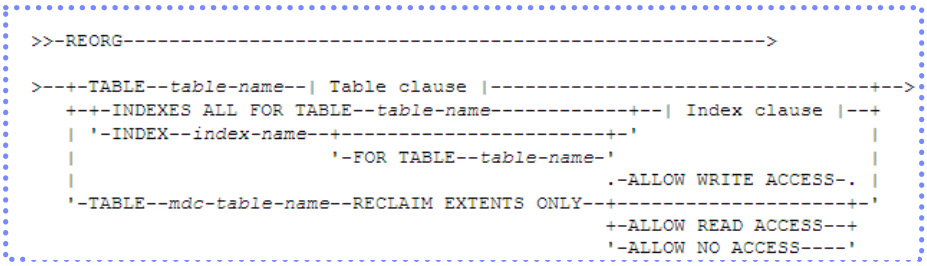


Figure 1004A • • • REORG TABLE 명령어

2 테이블은 기본적으로 클러스터 인덱스를 기준으로 재구성됩니다. INDEX 옵션을 지정하면 특정 인덱스의 정렬 순서를 기준으로 재구성합니다. 클러스터 인덱스가 없거나 INDEX 옵션을 지정하지 않으면, 데이터는 임의의 순서로 재구성됩니다.

```
$ db2 reorg table <스키마명>.<테이블명> index <스키마명>.<인덱스명>
```

3 재구성할 테이블의 복사본을 저장할 임시 테이블스페이스를 지정합니다. 지정하지 않으면 테이블이 저장된 테이블스페이스에 복사본을 생성하게 됩니다.

```
$ db2 reorg table <스키마명>.<테이블명> USE <시스템 임시 테이블스페이스명>
```

4 테이블을 재구성하는 동안 다른 사용자는 테이블에 대한 액세스 범위를 제한 받습니다. 액세스 허용 범위는 ALLOW 옵션의 2가지의 모드로 조절합니다.

모 드	설 명
ALLOW NO ACCESS	테이블을 액세스할 수 없습니다.
ALLOW READ ACCESS	읽기 액세스만 허용됩니다. 기본 모드입니다.
ALLOW WRITE ACCESS	읽기와 쓰기 액세스가 모드 허용됩니다. INPLACE 옵션을 사용한 온라인 재구성에서만 허용됩니다.

```
$ db2 reorg table  <스키마명>.<테이블명>
$ db2 reorg table <스키마명>.<테이블명>  ALLOW NO ACCESS
```

5 INPLACE 옵션을 사용하여 온라인 모드로 테이블을 재구성하면, 재구성이 진행되는 동안에 테이블의 쓰기 액세스를 허용할 수 있습니다. START, STOP, PAUSE, RESUME 옵션을 이용하여 온라인 재구성 작업을 중단시키고 옵션을 변경한 후에 다시 계속할 수 있습니다.

```
$ db2 reorg table <스키마명>.<테이블명> INPLACE ALLOW READ ACCESS
    START
$ db2 reorg table <스키마명>.<테이블명> INPLACE PAUSE
$ db2 reorg table <스키마명>.<테이블명> inplace ALLOW WRITE ACCESS
    NOTRUNCATE RESUME
$ db2 reorg table <스키마명>.<테이블명> INPLACE STOP
```

TOPIC 10 05 REORG INDEXES 명령어

Point

REORGCHK 수행 결과 인덱스 재구성을 권장 받으면, REORG INDEXES 명령어를 이용하여 인덱스의 데이터가 연속적인 페이지에 저장될 수 있도록 인덱스만 재구성할 수 있습니다.

Tip

SYSADM, SYSCTRL, SYSMAINT, DBADM 권한 또는 테이블에 대한 CONTROL 특권을 가진 사용자가 실행합니다.

Tip

REORG INDEXES 를 실행한 후에는 RUNSTATS, REBIND 를 실행하는 것이 권장됩니다.

1 REORG INDEXES 명령어로 인덱스를 재 생성하거나, 데이터가 삭제된 공간을 반환합니다.

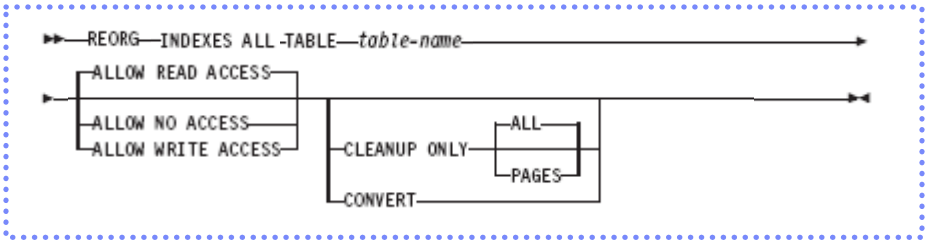


Figure 1005A REORG INDEXES 명령어

2 인덱스를 재구성하는 동안 다른 사용자는 테이블에 대한 액세스 범위를 제한 받습니다. 액세스 허용 범위는 ALLOW 옵션의 3가지의 모드로 조절합니다.

모 드	설 명
ALLOW NO ACCESS	테이블을 액세스할 수 없습니다.
ALLOW READ ACCESS	읽기 액세스만 허용됩니다. 기본 모드입니다.
ALLOW WRITE ACCESS	읽기와 쓰기 액세스가 모드 허용됩니다.

```
$ db2 reorg indexes all for table <스키마명>.<테이블명>
$ db2 reorg indexes all for table <스키마명>.<테이블명>  ALLOW NO
    ACCESS
$ db2 reorg indexes all for table <스키마명>.<테이블명>  ALLOW WRITE
    ACCESS
```

Tip

한 페이지의 모든 데이터가 삭제되어 완전이 빈 페이지를 'committed pseudo empty page' 라고 하며, SYSCAT.INDEXES 카탈로그 뷰의 NUM_EMPTY_LEAFS 컬럼의 값으로 확인할 수 있습니다.

3 인덱스를 재구성하는 작업은 실제로 인덱스를 재 생성하는 작업입니다. CLEANUP ONLY 옵션으로 인덱스를 재 생성하지 않고, 데이터가 삭제된 영역들을 동일한 테이블의 다른 인덱스가 사용할 수 있도록 할 수 있습니다. CLEANUP ONLY 옵션은 2 가지 모드를 지원합니다.

Tip

리프 페이지에서 삭제된 인덱스 데이터를 'committed pseudo deleted key' 라고 하며, 카탈로그 뷰인 SYSCAT.INDEXES 의 NUMRIDS_DELETED 컬럼의 값으로 확인할 수 있습니다.

모 드	설 명
PAGES	한 페이지의 모든 데이터가 삭제되어 완전히 비어있는 페이지를 인덱스 트리에서 제거합니다.
ALL	완전히 비어있는 페이지를 인덱스 트리에서 제거하고, 일부 영역이 비어 있는 페이지는 인접 페이지와 병합하여 한 페이지를 빈 페이지로 만든 후 인덱스 트리에서 제거합니다. 기본 모드입니다.

```
$ db2 reorg indexes all for table <스키마명>.<테이블명>  CLEANUP ONLY
$ db2 reorg indexes all for table <스키마명>.<테이블명>  CLEANUP ONLY
    PAGES
```

Tip

버전8 은 기본적으로 type-2 인덱스를 생성합니다. 버전7 의 데이터베이스를 마이그레이션하여 사용하면, 기존 인덱스는 type-1 이 됩니다.

4 CONVERT 옵션으로 type-1 인덱스를 type-2 인덱스로 변환할 수 있습니다.

```
$ db2 reorg indexes all for table <스키마명>.<테이블명>  CONVERT
```

TOPIC 10 06 RUNSTATS 명령어

Point

DB2는 비용 기반의 옵티마이저를 이용하므로, RUNSTATS명령어를 이용하여 통계정보를 최근 것으로 유지하는 것이 중요합니다. 통계정보는 SYSSTAT 스키마를 가진 시스템 카탈로그 뷰에 그 정보가 저장됩니다.

Tip

SYSADM, SYSCTRL, SYSMAINT, DBADM, LOAD 권한 또는 테이블의 CONTROL 특권을 가진 사용자가 실행합니다.

Tip

RUNSTATS 로 통계 자료를 갱신한 후에는 REBIND를 실행하는 것이 권장됩니다.

Tip

SYSSTAT로 시작되는 뷰의 값이 -1로 된 항목은 통계 자료가 갱신되지 않았음을 의미합니다.

Tip

runstats가 실행되는 동안 목표 테이블에 INSERT 가 발생하면 INDEX와의 불일치 경고 메시지가 반환될 수 있지만, 이는 무시해도 좋습니다.

Tip

수집된 대표적인 통계 자료는 다음과 같은 정보를 포함합니다.

- 테이블의 페이지 수
- 비어 있지 않은 페이지의 수
- 페이지 오버플로우의 정도
- 테이블의 행의 수
- 컬럼 내의 구별되는 값의 수
- 색인의 클러스터링 정도
- 색인 레벨의 수
- 각각의 색인에서 리프 페이지의 수
- 자주 사용된 컬럼 값의 발생 수
- 컬럼값의 분산
- 사용자 정의 함수에 대한 비용 예측

Tip

Sampling 중 BERNOULLI는 레코드단위인데 반해 SYSTEM은 페이지 단위로 샘플링하여 성능이 보다 우수합니다.

1 RUNSTATS 명령어로 수집된 데이터베이스에 대한 통계 자료를 시스템 카탈로그에 반영합니다.

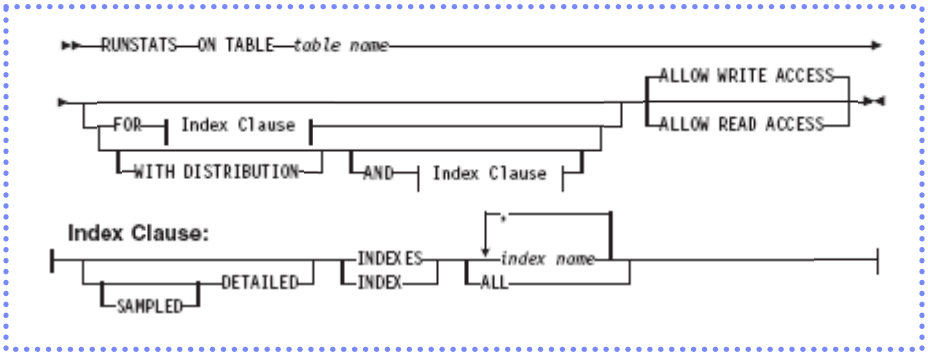


Figure 1006A RUNSTATS 명령어

2 특정 테이블에 대한 통계 자료를 갱신합니다.

```
$ db2 runstats on table <스키마명>.<테이블명>
```

3 특정 테이블의 인덱스에 대한 통계 자료만 갱신할 때에는 FOR INDEXES ALL 또는 FOR INDEX 옵션을 사용합니다.

```
$ db2 runstats on table <스키마명>.<테이블명> FOR INDEXES ALL
$ db2 runstats on table <스키마명>.<테이블명> FOR INDEX <스키마명>.<인덱스명>
```

4 특정 테이블과 인덱스에 대한 통계 자료를 모두 갱신할 때에는 AND INDEX 옵션 또는 AND INDEXES ALL 옵션을 사용합니다.

```
$ db2 runstats on table <스키마명>.<테이블명> AND INDEXES ALL
$ db2 runstats on table <스키마명>.<테이블명> AND INDEX <스키마명>.<인덱스명>
```

5 DETAILED 옵션을 이용하면 인덱스에 대한 상세한 통계 자료를 갱신합니다.

```
$ db2 runstats on table <스키마명>.<테이블명> for DETAILED indexes all
$ db2 runstats on table <스키마명>.<테이블명> and DETAILED indexes all
```

6 WITH DISTRIBUTION 옵션으로 기본 통계 자료와 컬럼값의 분포에 대한 통계 자료를 함께 수집하여 시스템 카탈로그에 반영할 수 있습니다.

```
$ db2 runstats on table <스키마명>.<테이블명> WITH DISTRIBUTION
```

7 ALLOW READ ACCESS 옵션은 통계 자료를 갱신하는 동안 쓰기 액세스를 차단합니다.

```
$ db2 runstats on table <스키마명>.<테이블명> ALLOW READ ACCESS
```

8 대용량 테이블의 경우, 일부의 Page만 Sampling하여 통계정보를 수집합니다. (예제 1%)

```
$ db2 runstats on table <스키마명>.<테이블명> AND INDEXES ALL
  TABLESAMPLE SYSTEM (1)"
```

TOPIC 10 07 액세스 플랜 갱신

Point

이전에 바인드 된 응용프로그램의 패키지에 저장된 액세스 플랜을 갱신하는 작업을 리바인드라고 합니다. RUNSTATS 명령어에 의해 갱신된 새로운 통계 자료를 사용하거나 추가적으로 작성된 인덱스를 이용하려면 리바인드 작업을 실행합니다.

Tip

동적 SQL에 대한 액세스 플랜은 시스템 통계 정보가 갱신되면, 실행 시에 자동으로 다시 작성됩니다. 정적 SQL은 BIND 또는 REBIND 명령어를 이용하여 명시적으로 액세스 플랜을 갱신하지 않으면, 기존의 액세스 플랜을 계속 유지하게 되므로 주의해야 합니다.

1 SQL문에 대한 액세스 플랜은 시스템 카탈로그의 패키지에 저장됩니다. list packages 명령어를 이용하여 패키지에 대한 현재 상태를 확인하면, VALID 항목에 패키지의 상태에 따라 Y(정상), N(무효), X(작동 불능) 중에 한 가지가 표시됩니다.

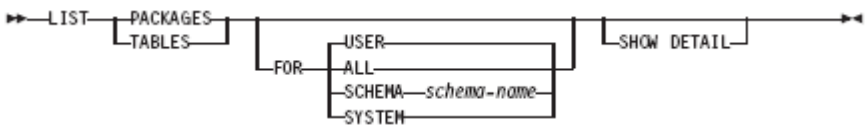


Figure 1007A ••• LIST PACKAGES 명령어

2 패키지 상태는 3가지로 분류되며, 정상 상태가 아닌 경우에는 리바인드가 필요합니다.

상 태	설 명
NORMAL	정상 상태입니다. 리바인드는 필요하지 않지만, RUNSTATS 명령을 수행한 후 갱신된 통계를 사용하거나 새로 작성된 인덱스를 이용하고자 한다면 명시적으로 리바인드를 실행할 수 있습니다.
INVALID	패키지에 포함된 SQL문에서 참조했던 테이블, 뷰, 트리거 등의의 데이터베이스 오브젝트가 삭제되면, 패키지는 무효 상태가 되어 리바인드 후에 사용할 수 있습니다. 리바인드는 해당 패키지가 다음 번에 실행될 때 데이터베이스 엔진에 의해 자동으로 실행됩니다.
INOPERATIVE	참조하고 있던 UDF가 삭제되면 패키지는 작동 불능 상태가 됩니다. 이 경우에는 반드시 BIND 명령어 또는 REBIND 명령어를 이용하여 사용자가 명시적으로 리바인드를 해야 합니다.

3 list packaes 명령어를 이용하여 패키지에 대한 현재 상태를 확인하면, VALID 항목에 패키지의 상태에 따라 Y(정상), N(무효), X(작동 불능) 중에 한 가지가 표시됩니다.

```
$ db2 list packages
```

```
                                Bound      Total                          Isolation
Package    Schema   Version    by         sections    Valid  Format  level     Blocking
---------- -------- ---------- -------- ------------ ------ ------- --------- --------
F4INS      USERA    VER1       SNOWBELL          221 Y      0       CS        U
F4INS      USERA    VER2.0     SNOWBELL          201 Y      0       RS        U
F4INS      USERA    VER2.3     SNOWBELL          201 N      3       CS        U
F4INS      USERA    VER2.5     SNOWBELL          201 Y      0       CS        U
PKG12      USERA               USERA              12 Y      3       RR        B
PKG15      USERA               USERA              42 Y      3       RR        B
SALARY     USERT    YEAR2000   USERT              15 Y      3       CS        N
```

Figure 1007B ••• 패키지의 상태 확인

TOPIC 10 08 REBIND 명령어

Point

명시적으로 리바인드를 요청하는 명령어로써 시스템 카탈로그에 저장되어 있는 기존의 특정한 패키지를 갱신합니다. 패키지에 저장된 액세스 플랜은 새로운 통계 자료를 기준으로 다시 생성됩니다.

Tip

SYSADM, DBADM 권한 또는 스키마에 대한 ALTERIN, 패키지에 대한 BIND 특권을 가진 사용자가 실행합니다.

1 최신의 데이터베이스에 대한 통계 자료를 기반으로 기 작성된 패키지의 액세스 플랜을 갱신합니다.

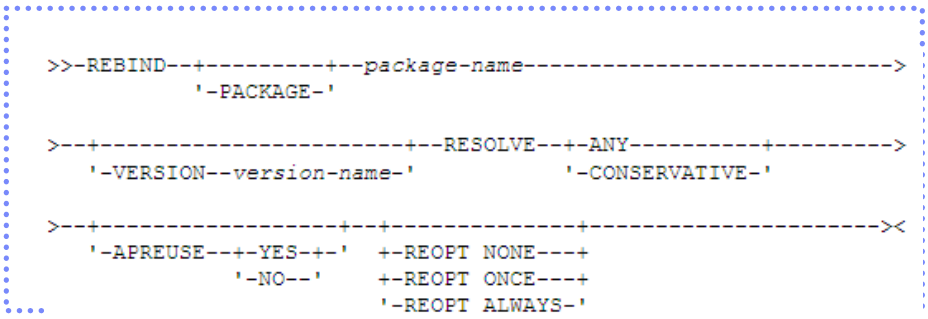


Figure 1008A • • • REBIND 명령어

2 옵션에 대한 설명은 아래와 같습니다.

Tip

RESOLVE CONVERSATIVE옵션은 Inoperative 패키지에 대해서는 지원되지 않습니다.

모 드	설 명
RESOLVE ANY	모든 오브젝트를 참조하여, 비 제한적 바인딩으로 수행됩니다.
RESOLVE CONVERSATIVE	이전 정의된 오브젝트만 참조하여, 제한적 바인딩으로 수행됩니다.
APREUSE YES	패키지 내에서 정적 SQL Plan이 재 사용됩니다
APREUSE NO	패키지 내에서 정적 SQL Plan이 재 사용되지 않습니다.
REOPT NONE	액세스 플랜을 결정할 때 호스트 변수, 매개 변수 표시 문자, 특수 레지스터리 변수에 할당된 실제값을 이용하지 않고, 미리 결정된 기본 추정값 (Default Filtering Factor)를 이용합니다. 결정된 액세스 플랜은 캐쉬에 저장되어 계속 사용됩니다. REOPT 옵션을 지정하지 않는 경우에 기본 모드로 제공됩니다.
REOPT ONCE	쿼리가 처음 실행되었을 때 호스트 변수, 매개 변수 표시 문자, 특수 레지스터리 변수에 할당된 실제값을 이용하여 액세스 플랜을 결정합니다. 결정된 액세스 플랜은 캐쉬에 저장되어 계속 사용됩니다.
REOPT ALWAYS	쿼리가 실행될 때마다 호스트 변수, 매개 변수 표시 문자, 특수 레지스터리 변수에 할당된 실제값을 이용하여 SQL문을 재컴파일하고, 액세스 플랜을 재결정합니다.

3 rebind 명령어를 이용하여 패키지를 갱신합니다.

```
$ db2 rebind <스키마명>.<패키지명> RESOLVE ANY REOPT ONCE
```

4 db2rbind 명령어는 특정 데이터베이스에 있는 모든 패키지의 액세스 플랜을 한꺼번에 갱신합니다. <로그파일명>은 임의로 지정합니다.

```
$ db2rbind <데이터베이스명> /l <로그파일명> all
```

UNIT 11

동시성 제어

동시성이란 다수의 사용자가 동시에 데이터베이스 오브젝트를 액세스할 수 있도록 허용하는 메커니즘입니다. DB2는 테이블과 행 수준의 잠금을 이용하여 동시성과 데이터의 무결성을 함께 보장합니다.

RR, RS, CS, UR, CC 등의 분리 수준으로 데이터 조회에 적용되는 잠금을 조절합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 동시성과 무결성
- 분리 수준
- 분리 수준 지정 방법
- 잠금의 대상
- 테이블 잠금 유형
- 테이블 잠금 지정 방법
- 행 잠금 유형
- 잠금 변환 현상
- 잠금 상승 현상
- 잠금 대기 현상
- 교착 상태

TOPIC 1101 동시성과 무결성

Point

동시성이란 다수의 사용자가 동시에 데이터베이스 오브젝트를 액세스할 수 있도록 허용하는 매커니즘입니다. 데이터베이스 엔진은 동시성을 지원하면서 데이터의 무결성을 보장할 수 있어야 합니다.

Tip

실제 응용프로그램의 구현 시에는 업무 특성에 따라 바람직하지 않은 현상을 의도적으로 허용할 수도 있습니다.

1 여러 사용자가 동시에 데이터베이스를 액세스할 때에는 다음과 같은 데이터베이스의 무결성을 위반하는 바람직하지 않은 현상이 발생될 수 있습니다.

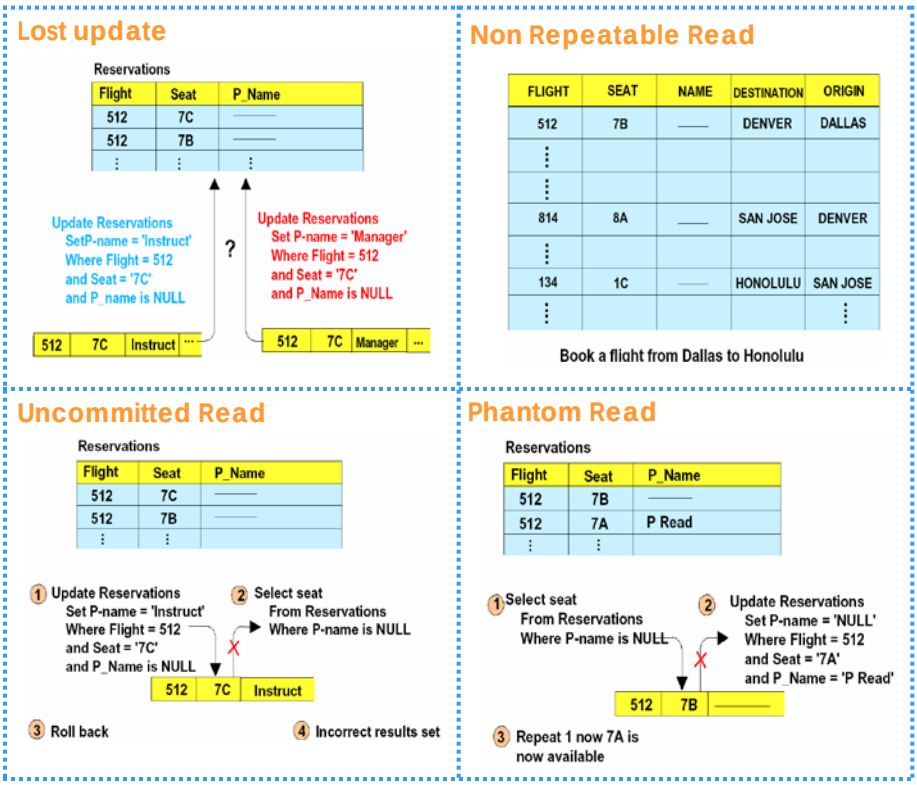


Figure 1101A ••• 동시성을 지원할 때 발생하는 현상

2 A와 B라는 사용자가 동일한 데이터를 액세스한다면, 다음과 같은 현상이 발생할 수 있습니다.

현 상	설 명
갱신 유실 Lost Update	A와 B가 동일한 데이터를 동시에 조회하고 변경하면, A가 변경한 값은 유실되고, B가 변경한 값이 남게 됩니다.
미확약 읽기 Uncommitted Read	A가 추가 또는 갱신한 후에 COMMIT 하지 않은 데이터를 B가 조회할 수 있도록 허용할 때, A가 commit으로 트랜잭션을 종료되었다면, B가 조회한 데이터는 정확한 데이터가 되지만, A가 rollback으로 트랜잭션을 종료한다면, B가 이미 조회한 데이터는 잘못된 데이터가 됩니다.
반복 읽기 불가능 Non Repeatable Read	A가 SELECT문을 실행하여 결과 집합을 조회하고, 동일한 트랜잭션에서 동일한 조건의 SELECT 문을 다시 요청하면 이전의 결과 집합이 그대로 반환되지 않습니다.
팬텀 읽기 Phantom Read	A가 SELECT문을 실행하여 10 건의 결과 집합을 조회하고, 동일한 트랜잭션에서 동일한 조건의 SELECT 문을 다시 요청하면 이전의 결과 집합인 10건은 그대로 반환되고, B가 추가한 1 건의 행이 추가적으로 반환됩니다. B가 새로 추가한 행을 팬텀 행이라고 합니다.

TOPIC 11-02 분리 수준

Point

한 응용프로그램이 SELECT문을 실행하여 결과 집합이 반환되면, 다른 응용프로그램은 해당 결과 집합에 영향을 주는 액세스를 제한 받게 됩니다. 액세스가 제한되는 대상의 범위와 SQL문의 종류는 분리 수준에 의해 조절됩니다.

Tip

- 분리 수준은 SELECT문에 대해서만 적용됩니다. 데이터를 변경시키는 INSERT,UPDATE,DELETE 문에 의해 잠금이 적용된 행은 COMMIT 또는 ROLLBACK이 될 때까지 액세스가 제한됩니다.

1 분리 수준은 다음과 같이 4 가지로 구분됩니다.

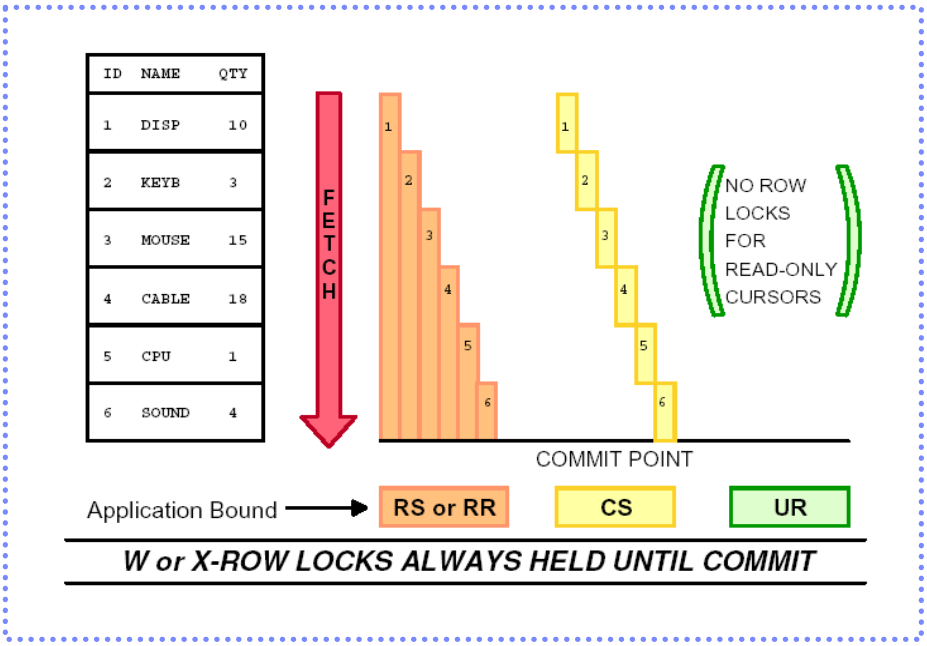


Figure 1102A • • • 분리 수준별 잠금 대상의 범위와 유지 기간

2 분리 수준의 유형별 의미는 다음과 같습니다.

Tip

- CC는 DB2 9.7에서 새롭게 추가되었습니다.

유형	설명
CC	Currently Committed의 약자입니다. 조회하는 UOW는 동시에 변경하는 다른 사용자에 의해 변경되고 있더라도 수행에 방해를 받지 않고 즉시 결과를 리턴 받게 됩니다. DB2 9.7에서 기본 분리 수준입니다.
CS	Cursor Stability의 약자입니다. 한 UOW 에서 동일한 조건의 SELECT문을 여러 번 실행하면 이전의 결과 집합이 반환되는 것을 보장하지 않습니다. 현재 처리 중인 한 개의 행에 대해서만 NS 잠금을 설정합니다. 다른 응용프로그램은 결과 집합의 데이터를 추가, 변경, 삭제할 수 있습니다. DB2 9.5까지 기본 분리수준 입니다.
RR	Repeatable Read 의 약자입니다. 한 UOW 에서 동일한 조건의 SELECT문을 여러 번 실행하면, 이전의 결과 집합과 항상 동일한 결과 집합이 반환되는 것을 보장하므로 반복 읽기가 가능합니다. 반환된 결과 집합의 모든 행과 결과 집합을 추출하기 위해 액세스된 모든 행에 대해 S 잠금을 적용합니다. 다른 응용프로그램은 결과 집합의 데이터를 추가, 변경, 삭제할 수 없습니다.
RS	Read Stability의 약자입니다. 한 UOW 에서 동일한 조건의 SELECT문을 여러 번 실행하면, 이전의 결과 집합을 항상 포함하고, 팬텀 행이 추가적으로 반환되는 것을 허용합니다. 반환된 결과 집합의 모든 행에 NS 잠금을 적용합니다. 다른 응용프로그램은 결과 집합의 데이터를 추가는 할 수 있지만, 변경과 삭제는 할 수 없습니다.
UR	Uncommitted Read의 약자입니다. 한 UOW 에서 동일한 조건의 SELECT문을 여러 번 실행하면 이전의 결과 집합이 반환되는 것을 보장하지 않습니다. 잠금을 전혀 적용하지 않습니다. 다른 응용프로그램은 결과 집합의 데이터를 추가, 변경, 삭제할 수 있습니다. 다른 응용프로그램에서 처리 중인 UOW의 COMMIT 되지 않은 데이터를 조회할 수 있고, COMMIT 하지 않은 데이터가 액세스되는 것을 허용합니다.

TOPIC 1103 분리 수준 지정 방법

Point

분리 수준은 개별 SQL문, 세션, 패키지, DB2 CLI 에서 지정할 수 있습니다. 개별 SQL문에서 WITH 옵션으로 분리 수준을 지정하는 것이 가장 우선적으로 적용됩니다.

Tip

WITH UR 을 지정하면, X 잠금이 설정된 데이터도 액세스할 수 있습니다.

Tip

패키지에 저장된 SELECT문의 분리 수준이 패키지 수준의 분리 수준과 다른 경우에는 해당 SELECT문에만 적용됩니다.

Tip

CLI 구성 파일인 db2cli.ini 파일에 저장되는 TXNISOLATION 키워드의 값은 1(UR), 2(CS), 4(RS), 8(RR) 중에서 한 가지로 지정합니다.

1. <분리 수준>을 지정할 때는 RR, RS, CS, UR 이라는 키워드를 이용합니다. 대소문자는 구별하지 않습니다.

2. 분리 수준의 유형을 지정하는 방법은 다음과 같습니다.

대 상	설 명
SQL문	SQL문의 마지막에 WITH 옵션을 사용하여 지정합니다. 패키지 수준과 세션 수준의 설정값보다 우선적으로 적용됩니다. SELECT 문, SELECT INTO 문, INSERT 문, FOR 문, searched DELETE 문, searched UPDATE 문, DECLARE CURSOR 문 등은 WITH 옵션을 사용할 수 있습니다.
현재 세션	SET CURRENT ISOLATION 문을 이용하여 지정합니다. 패키지에 저장된 동적 SQL문에 대해서 패키지 수준의 설정값보다 우선적으로 적용됩니다.
패키지	PREP 명령어의 ISOLATION 옵션, BIND 명령어의 ISOLATION 옵션으로 지정하며, 패키지에 포함된 모든 SELECT문에 적용됩니다.
DB2 CLI	CHANGE ISOLATION LEVEL 명령어, SQLSetConnectAttr 함수, db2cli.ini 파일의 TXNISOLATION 키워드를 이용하여 지정하며, CLI 와 ODBC 드라이버를 통한 실행시에 적용됩니다.
DB2 CLP	CHANGE ISOLATION LEVEL 명령어를 이용하며, 데이터베이스의 재접속시에 반영됩니다.

3. 다양한 환경에서 분리 수준을 UR로 설정하는 예는 다음과 같습니다.

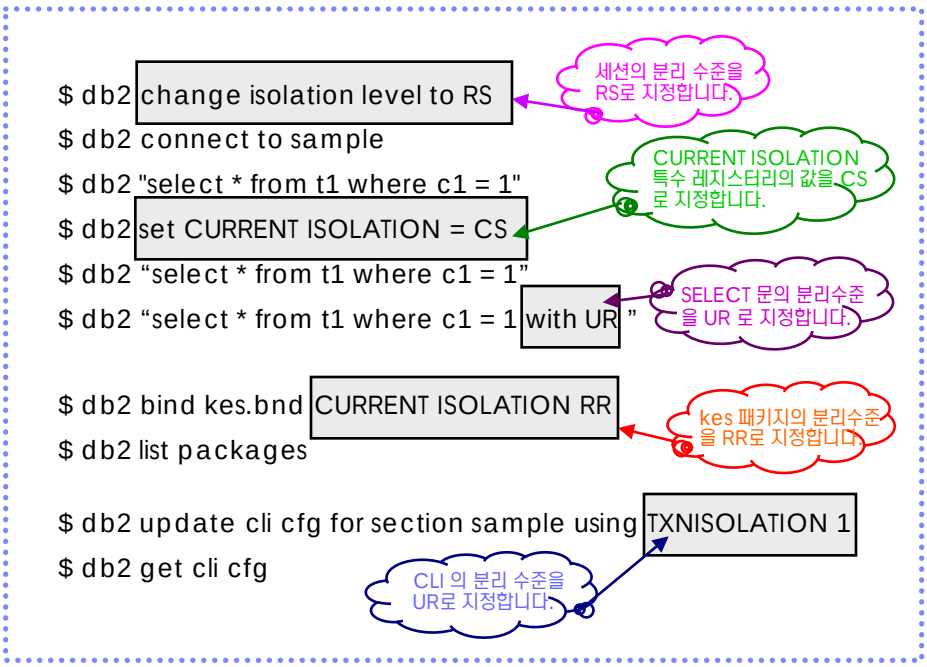


Figure 1103A ••• 다양한 환경에서 분리 수준을 UR로 설정하는 방법

TOPIC 1104 잠금의 대상

Point

QUIESCE 명령어, CONNECT 명령어, LOCK TABLE문으로 인스턴스, 데이터베이스, 테이블스페이스, 테이블에 사용자가 명시적으로 잠금을 적용할 수 있습니다. SQL문을 실행하면 엔진에 의해 테이블과 인덱스의 행에 자동으로 잠금이 적용됩니다.

1 잠금이 적용되는 데이터베이스 오브젝트는 다음과 같습니다.

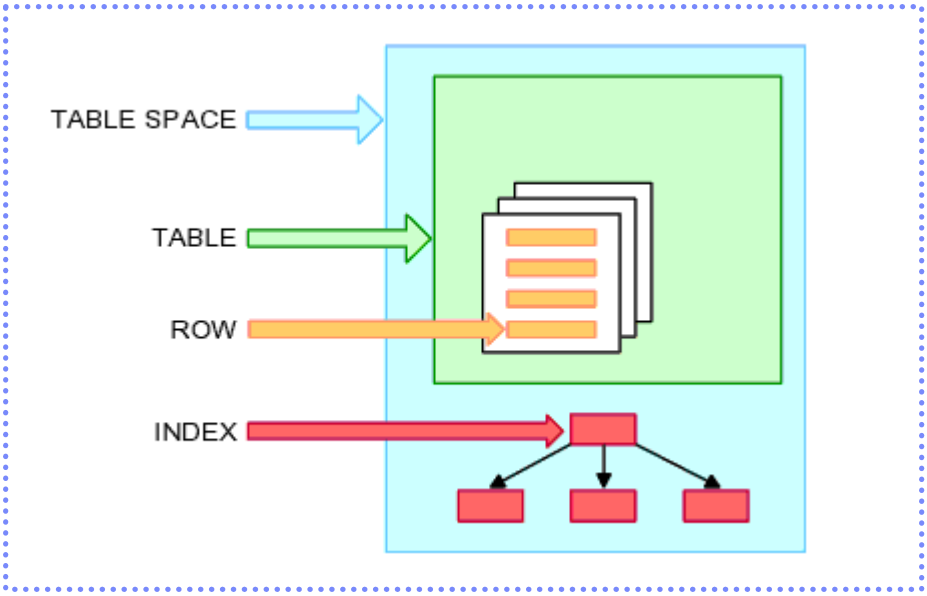


Figure 1104A ••• 잠금이 적용되는 데이터베이스 오브젝트

Tip

인스턴스, 데이터베이스, 테이블스페이스에 대한 잠금은 데이터베이스 관리 차원의 잠금입니다.

Tip

일반적인 응용프로그램에서 SQL문으로 데이터를 액세스하므로 테이블, 행, 인덱스에 대한 잠금이 적용됩니다.

2 잠금을 적용하는 방법은 다음과 같습니다.

대 상	설 명
인스턴스	QUIESCE INSTANCE 명령어를 실행하면, SYSADM / SYSCTRL / SYSMAINT 권한을 가진 사용자와 액세스가 허용된 사용자만 인스턴스와 데이터베이스를 액세스할 수 있습니다.
데이터베이스	CONNECT 문을 이용하여 데이터베이스에 접속할 때 배타적 모드의 접속이 가능합니다. 기본 접속 모드는 공유적 모드이므로 여러 사용자가 동시에 접속할 수 있습니다. QUIESCE DATABASE 명령어를 실행하면, SYSADM / SYSCTRL / SYSMAINT / DBADM 권한을 가진 사용자와 액세스가 허용된 사용자만 데이터베이스를 액세스할 수 있습니다.
테이블스페이스	QUIESCE TABLESPACES FOR TABLE 명령어를 실행하면, LOAD 명령어를 실행하는 동안 해당 테이블스페이스에 잠금을 적용합니다.
테이블	LOCK TABLE 문을 이용하여 배타적 또는 공유적 모드로 테이블 전체에 잠금을 설정할 수 있습니다. 한 테이블의 행 잠금의 비율이 지정한 기준을 초과하면, 내부적으로 행잠금이 테이블 잠금으로 변환될 수도 있습니다.
행	SQL문을 이용하여 데이터를 액세스하면 기본적으로 행 수준의 잠금이 적용됩니다.
인덱스	SQL문을 이용하여 데이터를 액세스할 때, 특정한 인덱스를 이용하여 액세스하게 되면, 해당 인덱스의 행에 잠금이 적용됩니다.

TOPIC 1105 테이블 잠금의 유형

Point

테이블 잠금의 유형에는 IN, IS, IX, SIX, S, U, X, Z 등이 있습니다. 테이블에 잠금이 적용되면, 행에 대한 잠금 정보는 별도로 관리되지 않습니다. 테이블의 모든 행에 대해서 액세스가 유형별로 제한됩니다.

1 테이블 잠금 유형과 잠금 유형간의 호환표은 다음과 같습니다.

IN	Intent None
IS	Intent Share
IX	Intent eXclusive
S IX	Share with Intent eXclusive
S	Share
U	Update
X	eXclusive
Z	superexclusive

MODE OF LOCK A	MODE OF LOCK B							
	IN	IS	S	IX	SIX	U	X	Z
IN	YES	YES	YES	YES	YES	YES	YES	NO
IS	YES	YES	YES	YES	YES	YES	NO	NO
S	YES	YES	YES	NO	NO	YES	NO	NO
IX	YES	YES	NO	YES	NO	NO	NO	NO
SIX	YES	YES	NO	NO	NO	NO	NO	NO
U	YES	YES	YES	NO	NO	NO	NO	NO
X	YES	NO	NO	NO	NO	NO	NO	NO
Z	NO	NO	NO	NO	NO	NO	NO	NO

Figure 1105A ··· 테이블 잠금 모드와 호환표

2 테이블 잠금 유형의 특성은 다음과 같습니다.

모 드	설 명
IN	잠금 소유자는 언커미트된 데이터를 포함하여 오브젝트의 어떤 데이터든지 읽을 수 있지만, 갱신할 수는 없습니다. 다른 동시 응용프로그램은 테이블을 읽거나 갱신할 수 있습니다. 분리 수준이 UR로 설정된 경우에 적용됩니다. IN, IS, S, IX, SIX, U, X 잠금과 호환됩니다.
IS	잠금 소유자는 잠긴 테이블에서 데이터를 읽을 수는 있지만, 이 데이터를 갱신할 수는 없습니다. 다른 응용프로그램은 테이블을 읽거나 갱신할 수 있습니다. IN, IS, S, IX, SIX, U 잠금과 호환됩니다.
IX	잠금 소유자 및 동시 응용프로그램은 데이터를 읽고 갱신할 수 있습니다. 다른 동시 응용프로그램은 테이블을 읽고 갱신할 수 있습니다. IN, IS, IX 잠금과 호환됩니다.
SIX	잠금 소유자는 데이터를 읽고 갱신할 수 있습니다. 다른 동시 응용프로그램은 테이블을 읽을 수 있습니다. 이미 IX 모드가 적용된 테이블에 대해 S 잠금을 적용해야 하는 경우에 사용됩니다. IN, IS 잠금과 호환됩니다.
S	잠금 소유자와 모든 동시 응용프로그램은 잠긴 데이터를 읽을 수는 있지만, 갱신할 수는 없습니다. IN, IS, S, U 잠금과 호환됩니다.
U	잠금 소유자는 데이터를 갱신할 수 있습니다. 다른 UOW는 잠긴 오브젝트에서 데이터를 읽을 수는 있지만, 갱신할 수는 없습니다. FOR UPDATE 옵션이 있는 커서를 이용한 SELECT문을 실행한 경우에 적용됩니다. IN, IS, S 잠금과 호환됩니다.
X	잠금 소유자만 데이터를 읽고 갱신할 수 있습니다. IN 잠금과 호환됩니다.
Z	테이블이 변경되거나 삭제된 경우, 테이블의 인덱스가 작성되거나 삭제되는 경우에 적용되며, 다른 동시 응용프로그램은 테이블을 읽거나 갱신할 수 없습니다.

Tip

분리 수준 UR을 사용하면 Z 잠금을 제외한 모든 잠금을 무시하고 액세스할 수 있습니다.

Tip

IS는 테이블의 일부 행들이 SELECT 문에 의해 S 또는 NS 잠금이 설정되었다는 것을 알려주기 위해 테이블에 적용되는 지시성 잠금 유형입니다.

Tip

IX는 테이블의 일부 행들이 INSERT, UPDATE, DELETE문에 의해 X 또는 W 잠금이 설정되었다는 것을 알려주기 위해 테이블에 적용되는 지시성 잠금 유형입니다.

TOPIC 1106 테이블 잠금 지정 방법

Point

데이터를 액세스하면 기본적으로 행 수준의 잠금이 적용됩니다. 특정 테이블의 행의 수가 많고, 대부분의 행을 액세스한다면, 행 잠금을 관리하기 위한 리소스가 많이 소모되어 응답 시간이 지연될 수 있으므로 명시적으로 테이블 잠금을 지정하는 것이 유리합니다.

Tip

기본 잠금 수준을 테이블 잠금으로 높이면 동시성이 저하시키는 단점이 있지만, 잠금을 관리하기 위한 비용이 줄어들 수 있으므로 실제적인 응답 시간은 빨라질 수 있습니다.

1 데이터의 기본 잠금 수준은 행 잠금입니다. 기본 잠금 수준을 테이블 잠금으로 지정하면, SQL문을 실행할 때 항상 테이블 잠금이 적용됩니다.

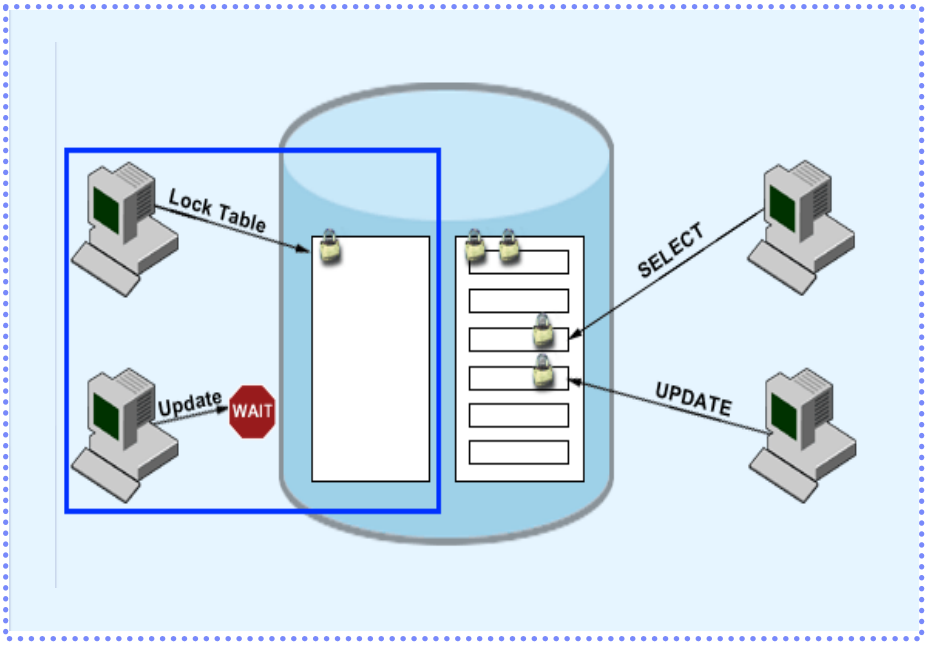


Figure 1106A • • • 명시적인 테이블 잠금

2 alter table 문에서 LOCKSIZE 옵션을 이용하여 지정합니다.

```
$ db2 "alter table <테이블명> LOCKSIZE TABLE"
$ db2 "alter table <테이블명> LOCKSIZE ROW"
```

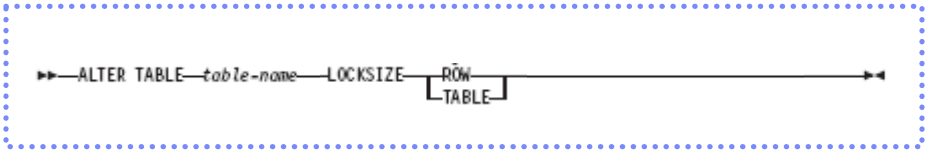


Figure 1106B • • • ALTER TABLE 문의 LOCKSIZE 옵션

3 lock table 문에서 LOCKSIZE 옵션을 이용하여 지정합니다.

```
$ db2 "lock table <테이블명> in SHARE mode"
$ db2 "lock table <테이블명> in EXCLUSIVE mode"
```

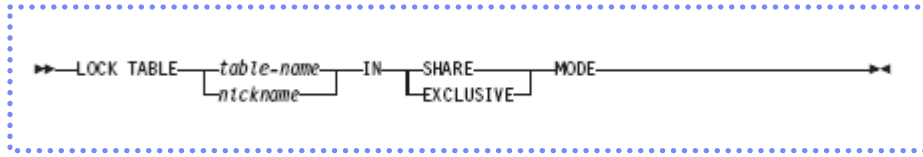


Figure 1106C • • • LOCK TABLE 문

TOPIC 1107 행 잠금 유형

Point

행 잠금의 유형에는 S, U, X, W, NS, NX, NW 등이 있습니다. 행에 잠금이 적용되면, 테이블에는 지시성 잠금이 적용됩니다. 잠금이 적용된 행에 대해서 액세스가 유형별로 제한됩니다.

Tip

S, NS 행 잠금이 적용된 테이블에는 자동으로 IS 테이블 잠금이 적용됩니다.

Tip

U, X, W, NW 행 잠금이 적용된 테이블에는 자동으로 IX 테이블 잠금이 적용됩니다.

1 행 잠금 유형과 잠금 유형간의 호환표은 다음과 같습니다.

Row Lock		Minimum Supporting Table Lock
S	Share	IS
U	Update	IX
X	eXclusive	IX
W	Weak exclusive	
NS	Next key Share	
NW	Next key Weak exclusive	

LOCK A MODE	MODE OF LOCK B					
	S	U	X	W	NS	NW
S	YES	YES	NO	NO	YES	NO
U	YES	NO	NO	NO	YES	NO
X	NO	NO	NO	NO	NO	NO
W	NO	NO	NO	NO	NO	YES
NS	YES	YES	NO	NO	YES	YES
NW	NO	NO	NO	YES	YES	NO

Figure 1107A ••• 행 잠금 모드와 호환표

Tip

분리 수준 UR을 사용하는 응용프로그램만은 어떤 유형의 잠금이라도 무시하고 액세스할 수 있습니다.

Tip

데이터를 조회한 후에 해당 데이터를 변경하는 로직을 가진 응용프로그램을 여러 사용자가 동시에 실행하려면 FOR UPDATE 옵션이 있는 커서를 사용하여 교착 상태가 발생하는 것을 방지하는 것이 좋습니다.

2 행 잠금 유형의 특성은 다음과 같습니다.

모 드	설 명
S	잠금 소유자와 모든 동시 응용프로그램은 잠긴 데이터를 읽을 수는 있지만, 갱신할 수는 없습니다. 분리 수준이 RR인 SELECT문을 실행한 경우에 조건에 맞는 행과 액세스에 사용된 모든 행에 대해 적용됩니다. S, U, NS 잠금과 호환됩니다.
U	잠금 소유자는 데이터를 갱신할 수 있습니다. 다른 UOW는 잠긴 오브젝트에서 데이터를 읽을 수는 있지만, 갱신할 수는 없습니다. FOR UPDATE 옵션이 있는 커서를 이용한 SELECT문을 실행하여 FETCH한 경우에 적용됩니다. S, NS 잠금과 호환됩니다.
X	잠금 소유자는 잠긴 오브젝트의 데이터를 읽고 갱신할 수 있습니다. UPDATE, INSERT, DELETE문을 실행한 경우에 적용됩니다. 다른 응용프로그램들은 X 잠금이 적용된 행에 대한 액세스를 할 수 없습니다.
W	type-2 인덱스가 없는 테이블로 행이 삽입되는 경우에 적용됩니다. 잠금 소유자는 잠긴 행을 변경할 수 있습니다. 중복 값 발견시 중복 값을 커미트했는지 판별하기 위해, 고유한 인덱스로의 삽입 중에도 이 잠금이 사용됩니다. NW 잠금과 호환성이 되는 것을 제외하고 이 잠금은 X 잠금과 유사합니다. NW잠금과 호환됩니다.
NS	잠금 소유자와 모든 동시 응용프로그램은 잠긴 행을 읽을 수는 있지만, 갱신할 수는 없습니다. S 잠금과 유사하며, 분리 수준이 RS 또는 CS인 경우에 적용됩니다. S, U, NS, NW 잠금과 호환됩니다.
NW	행이 인덱스에 삽입되면, NW 잠금은 다음 행에서 획득됩니다. type-2 인덱스의 경우, 다음 행이 현재 RR 스캔에 의해 잠긴 경우에만 발생합니다. 잠금 소유자는 잠긴 행을 읽을 수는 있지만, 갱신할 수는 없습니다. 이 잠금 모드는 W 및 NS 잠금과 호환되는 것을 제외하고는 X 잠금과 유사합니다. W, NS 잠금과 호환됩니다.

TOPIC 1108 잠금 변환 현상

Point

이미 잠금이 적용된 행 또는 테이블에 다시 잠금이 요청되면, 기존의 잠금 정보와 수위를 비교합니다. 기존의 잠금이 더 제한적이라면 새로운 잠금 정보는 무시됩니다. 새로운 잠금이 더 강력하다면 기존의 잠금 정보는 새로운 잠금 정보로 변환됩니다.

Tip

테이블 잠금과 행 잠금의 변환은 개별적으로 적용됩니다.

1 테이블 잠금과 행 잠금의 단계는 다음과 같습니다. 하위 잠금이 적용된 상태에서 상위 잠금이 요청되면, 하위 잠금 정보는 상위 잠금 정보로 갱신됩니다.

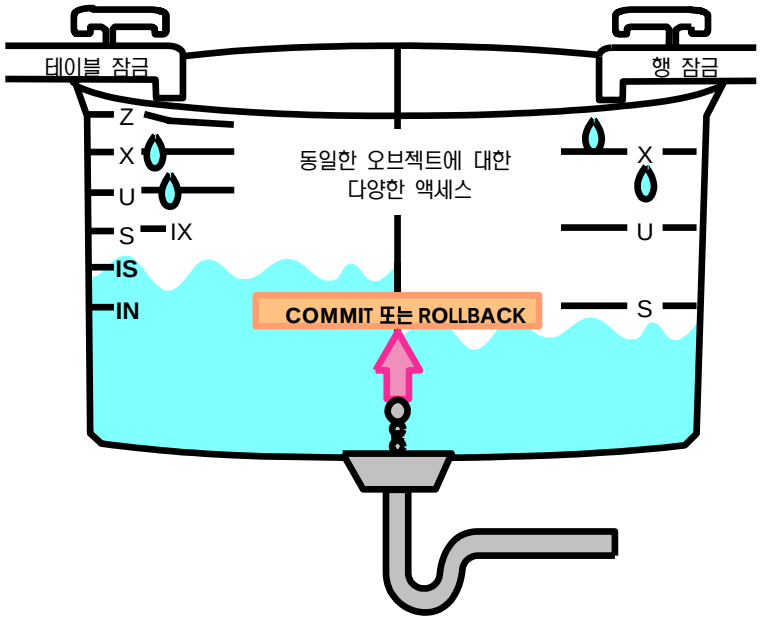


Figure 1108A ••• 잠금 변환

2 SELECT문의 조건문에서 고유한 인덱스가 있는 컬럼을 이용하여 데이터를 조회하면, 해당 행에 대해 NS 또는 S 모드의 잠금이 적용됩니다. 동일한 행에 대해 UPDATE 문을 다시 실행하면, X 잠금이 적용되고, LOCKLIST에 저장된 해당 행에 대한 NS 또는 S 모드의 잠금 정보는 X 모드의 잠금으로 갱신됩니다.

```
SELECT   *
FROM     kes.test_taken
WHERE    seat_no = '1'
AND      date_take = CURRENT DATE
WITH RR;
```

```
UPDATE   kes.test_taken
SET      seat_no = '2'
WHERE    seat_no = '1'
AND      date_take = CURRENT DATE;
```

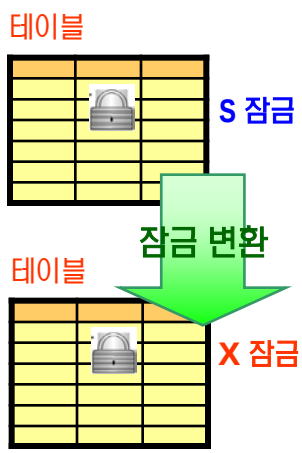


Figure 1108B ••• 행 잠금 유형의 변환

TOPIC 1109 잠금 상승 현상

Point

행잠금이 테이블 잠금으로 전환되는 현상을 잠금 상승 현상이라고 합니다. 잠금 상승 현상은 교착 상태를 유발시킬 수 있으며, 동시성을 저하시키므로, 가급적이면 발생하지 않도록 데이터베이스 구성 변수인 LOCKLIST와 MAXLOCKS을 조절합니다.

Tip

LOCKLIST 는 '온라인 구성 가능 변수' 이므로 동적으로 변경할 수 있습니다.

Tip

LOCKLIST를 AUTOMATIC으로 설정하면 자동으로 DBMS가 크기를 조정합니다.

1. update db cfg 명령어로 데이터베이스 구성 변수인 LOCKLIST 구성 변수를 이용하여 잠금 정보를 저장하는 메모리 영역의 크기를 조절합니다. <크기>는 4K 페이지 단위로 지정합니다.

```
$ db2 update db cfg for <데이터베이스명> using LOCKLIST <크기>
```

2. 잠금 정보는 데이터베이스별로 공유 메모리에 있는 LOCKLIST 영역에 저장됩니다. 잠금 정보의 양이 LOCKLIST의 크기를 초과하게 되면, 가장 많은 행 잠금을 가진 테이블을 식별하여 그 테이블에 대한 행 잠금을 모두 취소하고 테이블 잠금을 적용합니다.

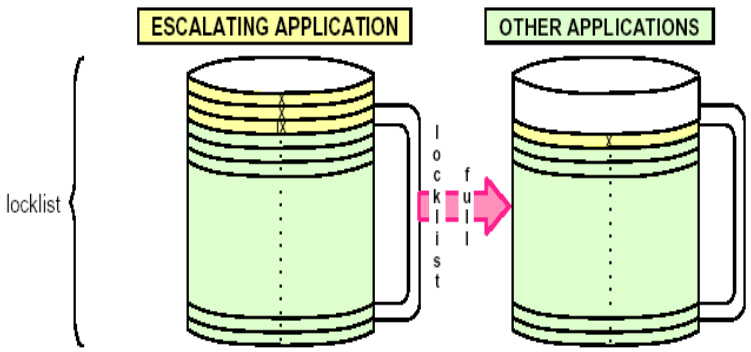


Figure 1109A ••• LOCKLIST 구성 변수와 잠금 상승 현상

Tip

MAXLOCKS 는 '온라인 구성 가능 변수' 이므로 동적으로 변경할 수 있습니다.

3. MAXLOCKS 구성 변수는 LOCKLIST 구성 변수가 지정한 크기에서 한 응용프로그램이 사용할 수 있는 최대 비율을 지정합니다. <비율>은 백분율로 표시합니다.

```
$ db2 update db cfg for <데이터베이스명> using MAXLOCKS <비율>
```

4. 한 응용프로그램에 대한 잠금 정보의 양이 MAXLOCKS의 비율을 초과하게 되면, 해당 응용프로그램에서 가장 많은 행 잠금을 가진 테이블을 식별하여 그 테이블에 대한 행 잠금을 모두 취소하고, 테이블 잠금을 적용합니다.

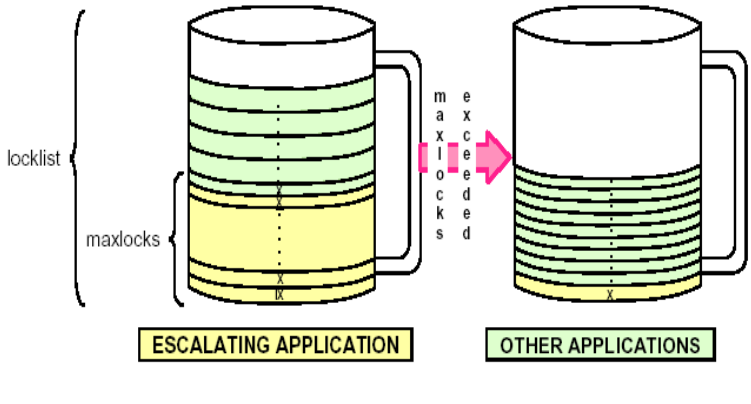


Figure 1109B ••• MAXLOCKS 구성 변수와 잠금 상승 현상

TOPIC 11-10 잠금 대기 현상

Point

한 응용프로그램이 잠금을 적용한 행을 다른 응용프로그램에서 액세스하려면, 그 잠금이 해제될 때까지 대기해야 합니다. 기본 잠금 대기 시간은 무한대이며, 데이터베이스 구성 변수인 LOCKTIMEOUT 을 이용하여 대기 시간을 조절할 수 있습니다.

Tip

LOCKTIMEOUT 구성 변수의 변경값을 반영하려면 데이터베이스의 재활성화가 필요합니다.

1. update db cfg 명령어로 LOCKTIMEOUT 데이터베이스 구성 변수를 설정합니다. <잠금 대기 시간>은 1초 단위로 표시합니다.

```
$ db2 update db cfg for <데이터베이스명> using LOCKTIMEOUT <잠금 대기 시간>
```

2. LOCKTIMEOUT 데이터베이스 구성 변수의 기본값은 -1로서 무한대로 대기하는 것을 의미합니다. 값을 0 으로 설정하면, SQL문 요청 시점에서 잠금을 획득하지 못하면 즉시 중단되게 합니다. 일반적인 OLTP 환경에서는 잠금 대기 시간을 30초 이내로 설정하도록 합니다. 잠금 대기 시간 이내에 필요한 잠금을 획득하면, 응용프로그램은 작업을 계속할 수 있습니다.

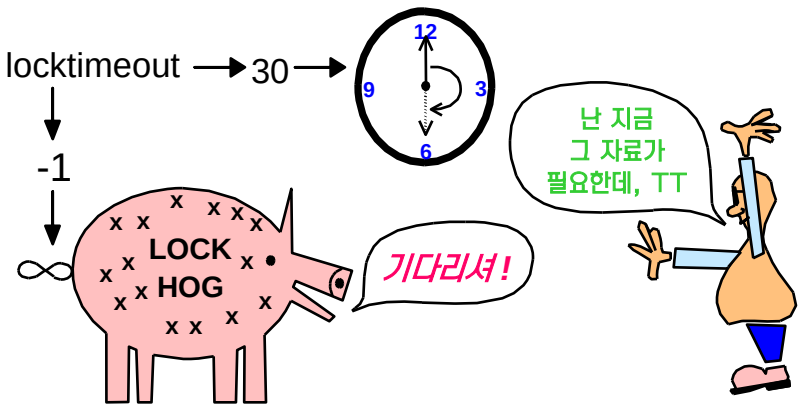


Figure 1110A ··· LOCKTIMEOUT 구성 변수

3. LOCKTIMEOUT 구성 변수의 값을 초과할 때까지 필요한 잠금을 획득하지 못하면, 응용프로그램은 중단되어 SQL0911N, SQLSTATE 40001과 이유 코드 68 이 반환됩니다.

```
<세션 A>
$ db2 connect to sample
$ db2 "create table t1 (c1 int not null primary key, c2 int)"
$ db2 "insert into t1 values (1,10),(2,20),(3,30)"
$ db2  +c "update t1 set c2 = c2 + 100 where c1= 2"
```

```
<세션 B>
$ db2 connect to sample
$ db2  "select * from t1 where c1 = 2"
SQL0911N  현재의 트랜잭션이 교착 상태 또는 시간종료로 인해 롤백되었습니다.
이유 코드 "68".  SQLSTATE=40001
```

Figure 1110B ··· 잠금 대기 시간 초과

TOPIC 1111 교착 상태

Point

⑩두 개의 응용프로그램이 서로 잠금 대기 상태가 되는 것을 교착 상태라고 합니다. 데이터베이스 구성 변수인 DLCHKTIME을 주기로 교착 상태에 있는 응용프로그램을 파악하고, 한 응용프로그램을 강제로 종료시키면, 다른 응용프로그램은 작업을 계속합니다.

Tip

DLCHKTIME 은 '온라인 구성 가능 변수' 이므로 attach 명령어를 이용하여 인스턴스에 접속한 후 변경하면 즉시 반영될 수 있습니다.

1. update db cfg 명령어로 DLCHKTIME 데이터베이스 구성 변수를 설정합니다. <점검 간격>은 밀리초(1000분의 1초) 단위로 표시합니다.

```
$ db2 update db cfg for <데이터베이스명> using DLCHKTIME <점검 간격>
```

2. DLCHKTIME 구성 변수의 기본값은 10000 ms로 10초 간격으로 교착 상태가 점검됩니다.

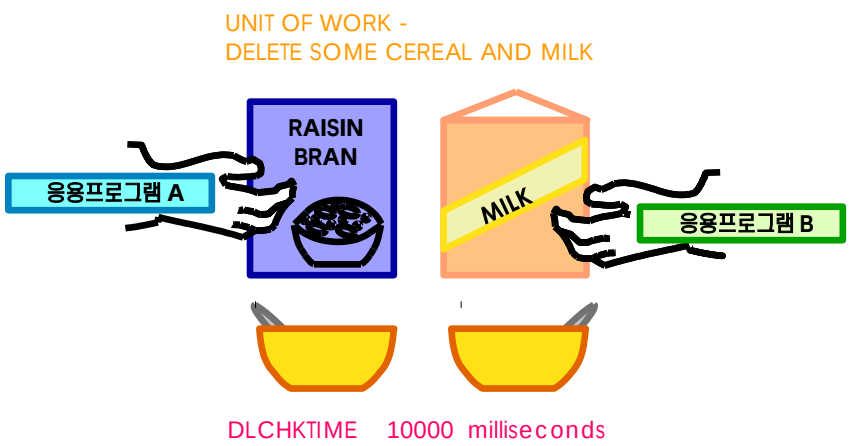


Figure 1111A ••• DLCHKTIME 구성 변수

Tip

교착 상태를 해결하기 위해 희생자 프로세스를 선택하는 기준은 엔진 내부의 알고리즘으로 사용자가 조절할 수 없습니다.

3. 교착 상태가 감지되면, 희생자(victim)로 선정된 한 쪽의 응용프로그램은 강제로 종료되고, SQL0911N, SQLSTATE 40001과 이유 코드 2가 반환됩니다.

```
<세션 A>
$ db2 connect to sample
$ db2 "create table t1 (c1 int not null primary key, c2 int)"
$ db2 "insert into t1 values (1,10),(2,20),(3,30)"
$ db2  +c "select * from t1 where c1 = 1 with rs"
$ db2  +c "update t1 set c2 = c2 + 100 where c1= 2"
```

```
<세션 B>
$ db2 connect to sample
$ db2  +c "select * from t1 where c1 = 2 with rs"
$ db2  +c "update t1 set c2 = c2 + 100 where c1= 1"
SQL0911N  현재의 트랜잭션이 교착 상태 또는 시간종료로 인해 롤백되었습니다.
이유 코드 "2".  SQLSTATE=40001
```

Figure 1111B ••• 교착 상태로 인한 트랜잭션의 롤백

UNIT 12

권한과 특권

DB2는 운영체제 또는 전문 보안 소프트웨어에서 제공하는 사용자 관리 기능을 이용하여 데이터베이스 접속 시에 사용자 인증을 실행합니다. 인스턴스 수준의 조작을 위한 여러 단계의 권한 체계를 제공하며, GRANT문과 REVOKE문으로 데이터베이스 오브젝트 별 특권을 제어하여 그룹과 사용자의 액세스 유형을 제한합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 사용자 인증
- SERVER 인증 유형
- CLIENT 인증 유형
- 권한
- 권한별 기능
- 인스턴스 권한 제어 방법
- 데이터베이스 권한 제어 방법
- 특권
- 특권 제어 방법
- 데이터베이스 특권
- 테이블스페이스 특권
- 스키마 특권
- 테이블 특권
- 뷰 특권
- 인덱스 특권
- 패키지 특권
- 루틴 특권
- 시퀀스 특권
- 간접 권한과 특권

TOPIC 12 01 사용자 인증

Point

DB2는 운영체제 또는 보안 소프트웨어의 사용자 관리 기능을 이용합니다. CONNECT문에서 제공된 사용자 ID와 암호를 이들에게 전달하여 유효한 사용자인지를 점검하게 합니다. 기본 인증 방식은 SERVER입니다.

Tip

잘못된 사용자ID 또는 암호를 제공하면, SQL30082N와 이유 "24 " 가 반환되면서 접속이 실패합니다.

Tip

원격 데이터베이스에 접속할 때는 반드시 사용자 ID와 암호를 명시적으로 제공해야 합니다.

Tip

catalog database 명령어에서 데이터베이스별로 인증 방식을 지정할 수 있습니다.

Tip

AUTHENTICATION의 값에 변경하면, 인스턴스를 재기동해야 합니다.

1. 모든 응용프로그램은 데이터베이스에 접속할 때, connect 문을 이용하여 사용자 ID와 암호명을 명시하게 됩니다. 명시한 사용자 ID와 암호명이 유효하지 않으면, 데이터베이스에 대한 접속은 허용되지 않습니다.

```
$ db2 connect to <데이터베이스명> user <사용자명> using <암호명>
```

2. connect 문에서 사용자 ID와 암호를 명시하지 않으면, 현재 세션의 로그인 사용자와 암호명이 사용됩니다.

```
$ db2 connect to <데이터베이스명>
```

3. connect 문에서 제공된 사용자 ID와 암호를 이용하여 OS의 사용자 관리 기능을 요청합니다. 사용자 인증의 방식은 인스턴스 구성 변수인 AUTHENTICATION에 의해 결정되며, 대표적인 인증 방식은 다음과 같습니다.

인 증 방 식	설 명
SERVER	목표 데이터베이스가 존재하는 서버에서 사용자 인증됩니다.
SERVER_ENCRYPT	목표 데이터베이스가 존재하는 서버에서 사용자 인증되며, 사용자 ID와 암호명이 암호화되어 전송됩니다.
CLIENT	연결 시도하는 클라이언트에서 사용자 인증됩니다.
KERBEROS	Kerberos 보안 메커니즘을 사용하여 인증됩니다.
DATA_ENCRYPT	목표 데이터베이스가 존재하는 서버에서 사용자 인증되며, 해당 연결은 데이터 암호화 됩니다.

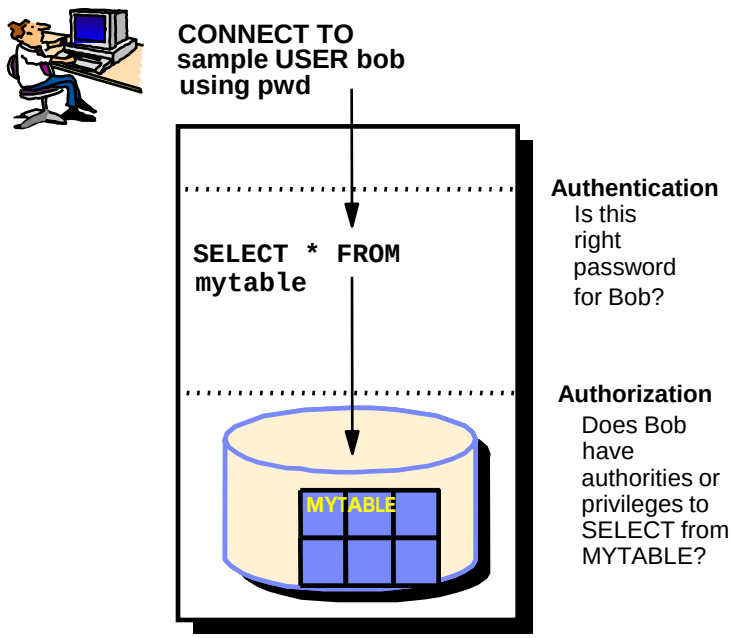


Figure 1201A • • • 사용자 인증과 테이블 액세스 권한

TOPIC 12 02 SERVER 인증 유형

Point

원격 클라이언트에서 데이터베이스 접속 시에 제공된 사용자 ID와 암호는 네트워을 통해서 데이터베이스 서버에 전달되고, 서버의 OS에 의해 사용자 ID와 암호가 점검됩니다. 인스턴스 구성 변수 AUTHENTICATION의 값은 SERVER 입니다.

Tip

SERVER_ENCRYPT 방식을 사용하면, 전달되는 사용자 ID와 암호를 내부적인 암호화 기법으로 변환하여 전송합니다.

Tip

클라이언트에는 사용자 ID가 존재할 필요가 없습니다.

Tip

클라이언트에 동일한 사용자가 존재하고, 암호가 다른 경우에도 점검은 서버에서 이루어지므로 상관이 없습니다.

1 서버의 인스턴스 구성 변수 AUTHENTICATION을 SERVER로 설정합니다.

```
$ db2 get dbm cfg | grep AUTHENTICATION
데이터베이스 관리 프로그램 인증       (AUTHENTICATION) = SERVER
```

2 클라이언트에서 catalog db 명령어를 이용하여 원격 데이터베이스를 등록합니다.

```
C:\> db2 catalog db <원격데이터베이스명> as <데이터베이스별명> at node
    <원격노드명>
C:\> list database directory | find “인증”
인증                                  = SERVER
```

3 클라이언트에서 connect 문을 이용하여 데이터베이스 서버에 접속을 요청하면, 제공된 사용자ID와 암호가 서버로 전송되어 서버의 OS에 의해 점검됩니다. 서버의 OS에 사용자 ID가 존재하고, 암호명이 일치하면 사용자 인증은 성공하여 데이터베이스에 재한 접속이 허용됩니다.

```
C:\> db2 connect to <데이터베이스별명> user <서버의 사용자 ID> using <서버
    의 암호명>
```

4 서버에 제공된 사용자 ID가 존재하지 않거나, 암호명이 일치하지 않으면 사용자 인증은 실패하고 데이터베이스에 대한 접속은 허용되지 않습니다.

```
C:>db2 connect to sample01 user kr001325 using xx
SQL30082N  보안 이유 "24"("USERNAME AND/OR PASSWORD
    INVALID")(으)로 인해 연결
시도에 실패했습니다.  SQLSTATE=08001
```

$ db2 connect to sample user db2user using db2pwd

④ "db2user" 와 "db2pwd" 가 서버에서 유효한 사용자인지 점검합니다.

③ 사용자 ID "db2user" 와 암호 "db2pwd" 가 서버로 전송됩니다.

② 데이터베이스 sample에 사용자 ID "db2user" 와 암호 "db2pwd" 를 이용하여 접속합니다.

① 클라이언트에 사용자 ID "peter" 와 암호 "peterpwd" 를 이용하여 로그온합니다.

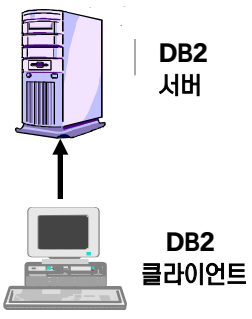


Figure 1202A SERVER 인증

TOPIC 12 03 CLIENT 인증 유형

Point

데이터베이스 접속 시에 제공된 사용자 ID와 암호는 클라이언트의 OS에 의해 점검되고, 인증이 성공하면 사용자 ID를 네트웍을 통해서 데이터베이스 서버에 전달합니다. 인스턴스 구성 변수 AUTHENTICATION의 값은 CLIENT 입니다.

Tip

서버에는 사용자 ID가 존재할 필요가 없습니다.

Tip

서버에 동일한 사용자가 존재하고, 암호가 다른 경우에도 점검은 클라이언트에서 이루어지므로 상관이 없습니다.

1. 서버의 인스턴스 구성 변수 AUTHENTICATION의 값을 CLIENT로 설정합니다.

```
$ db2 update dbm cfg using AUTHENTICATION CLIENT
$ db2stop force
$ db2start
```

2. 클라이언트에서 catalog db 명령어를 이용하여 원격 데이터베이스를 등록합니다.

```
C:\> db2 catalog db <원격데이터베이스명> as <데이터베이스별명> at node
    <원격노드명>
C:\> list database directory | find “인증”
인증                                = CLIENT
```

3. connect 문에서 제공된 사용자ID와 암호는 클라이언트의 OS에 의해 점검됩니다. 클라이언트의 OS에 사용자 ID가 존재하고, 암호명이 일치하면 사용자 인증은 성공하여 데이터베이스에 대한 접속이 허용됩니다. 사용자 ID만 서버로 전송됩니다.

```
C:\> db2 connect to <데이터베이스별명> user <클라이언트의 사용자 ID> using
    <클라이언트의 암호명>
```

4. 제공된 사용자 ID가 클라이언트에 존재하지 않거나, 암호명이 일치하지 않으면 사용자 인증은 실패하고 데이터베이스에 대한 접속은 허용되지 않습니다.

```
C:> db2 connect to <데이터베이스별명> user <클라이언트의 사용자 ID> using
    <잘못된 클라이언트의 암호명>
    SQL30082N  보안 이유 "24"("USERNAME AND/OR PASSWORD
    INVALID")(으)로 인해 연결시도에 실패했습니다.  SQLSTATE=08001
```

$ db2 connect to sample user db2user using db2pwd

⑤ 서버의 사용자 인증은 필요하지 않습니다.

④ 인증이 성공하면, 사용자 ID인 "db2user" 만 서버로 전송됩니다.

③ "db2user" 와 "db2pwd" 가 클라이언트에서 유효한 사용자인지 점검합니다.

② 데이터베이스 sample에 사용자 ID "db2user" 와 암호 "db2pwd" 를 이용하여 접속합니다.

① 클라이언트에 사용자 ID "peter" 와 암호 "peterpwd" 를 이용하여 로그온합니다.

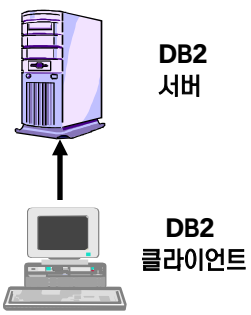


Figure 1203A ··· CLIENT 인증

TOPIC 1204 권한

Point

일련의 DB2 명령어를 실행하거나 데이터베이스를 액세스할 수 있는 능력을 권한이라고 합니다. 권한에는 인스턴스 권한과 데이터베이스 권한으로 분류됩니다.

Tip

상위 권한의 소유자는 하위 권한을 자동으로 소유합니다.

1 인스턴스 권한과 데이터베이스 권한의 체계는 다음과 같이 분류됩니다.

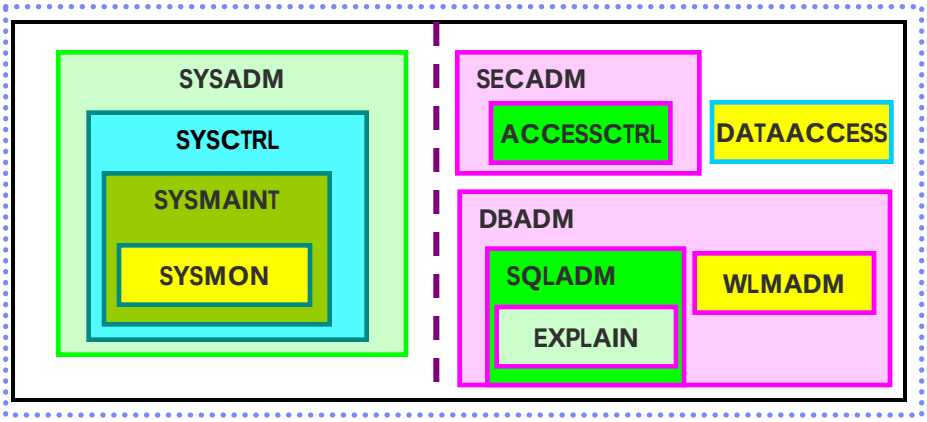


Figure 1204A • • • 인스턴스 권한과 데이터베이스 권한

Tip

인스턴스 권한은 그룹에만 부여되므로 권한 그룹이라고도 합니다.

2 인스턴스 권한은 4가지로 구분됩니다. 인스턴스 구성 변수를 이용하여 OS에 정의된 특정한 그룹 단위로 제어합니다.

권 한	설 명
SYSADM	최고의 인스턴스 권한 그룹으로 인스턴스와 데이터베이스에 대한 모든 유지 보수, 생성, 제거와 연관된 작업을 실행할 수 있습니다. 인스턴스에 존재하는 모든 데이터베이스의 데이터를 직접 액세스할 수 있습니다.
SYSCTRL	인스턴스와 데이터베이스의 유지 보수, 생성, 제거 등과 연관된 명령어를 실행할 수 있으며, 데이터에 대한 직접적인 액세스는 허용되지 않습니다.
SYSMAINT	인스턴스와 데이터베이스의 유지 보수와 관련된 백업 복구 명령어를 실행할 수 있으며, 데이터에 대한 직접적인 액세스는 허용되지 않습니다.
SYSMON	인스턴스 또는 데이터베이스의 시스템 모니터 스냅샷과 연관된 명령어를 실행할 수 있습니다. 데이터에 대한 직접적인 액세스는 허용되지 않습니다.

Tip

9.70이후에는 SYSADM사용자가 더 이상 DBADM권한을 기본으로 보유하지 않습니다. 단, SYSADM권한을 가진 사용자가 데이터베이스를 생성하는 경우에 기존과 동일한 권한을 부여 받을 수 있습니다. 만일 데이터베이스를 생성하지 않은 경우에는 동일한 권한을 가지기 위해서는 SECADM에 의해 DBADM권한을 부여받아야 합니다.

3 데이터베이스 권한은 다음과 같습니다. grant 문과 revoke문을 이용하여 사용자 또는 그룹 단위로 제어할 수 있습니다.

권 한	설 명
SECADM	데이터베이스에 대한 보안 관리자입니다.
DBADM	특정 데이터베이스의 오브젝트에 대한 모든 관리 작업을 할 수 있으며, 해당 데이터베이스의 데이터를 직접 액세스할 수 있습니다.
ACCESSCTRL	ACCESSCTRL, DATAACCESS, DBADM, SECADM 권한 및 감사 루틴에 대한 특권을 제외한 모든 특권과 데이터베이스 권한을 부여/취소 가능합니다.
DATAACCESS	데이터베이스 테이블에 저장된 데이터에 액세스할 수 있도록 허용합니다
SQLADM	SQL문을 모니터하고 조정할 수 있습니다.
WLMADM	워크로드 관리자 역할을 할 수 있습니다. 특히 WLMADM 권한 보유자는 오브젝트 작성 및 삭제, WLM 특권 부여 및 취소, WLM루틴 실행 등을 수행할 수 있습니다.
EXPLAIN	쿼리 계획에 참조된 테이블의 데이터에 액세스할 수 있는 특권 없이 해당 쿼리 계획을 확인할 수 있습니다.

Tip

9.70이후, SECADM은 DBADM 및 SECADM을 포함하여 모든 권한 및 특권을 부여 및 취소할 수 있습니다.

TOPIC 12 05 권한별 기능

Point

SYSADM, SYSCTRL, SYSMAINT, SYSMON 권한은 데이터베이스 시스템과 관련된 명령어를 실행합니다. SYSADM 또는 DBADM, LOAD 권한의 소유자는 데이터베이스와 관련된 명령어를 실행합니다.

Tip

SYSADM, DBADM 권한을 가진 사용자는 데이터베이스에 접속하여 SQL문으로 데이터를 액세스할 수 있습니다.

1 권한별로 실행 가능한 명령어는 다음과 같습니다.

Function	SYSADM	SYSCTRL	SYSMAINT	DBADM
MIGRATE DATABASE	YES			
UPDATE DBM CFG	YES			
GRANT/REVOKE DBADM	YES			
UPDATE db/node/dcs directories	YES	YES		
FORCE USERS OFF SYSTEM	YES	YES		
CREATE/DROP DATABASE	YES	YES		
CREATE/DROP/ALTER TABLE SPACE	YES	YES		
RESTORE TO NEW DATABASE	YES	YES		
UPDATE DB CFG	YES	YES	YES	
BACKUP DATABASE or TABLE SPACE	YES	YES	YES	
RESTORE TO EXISTING DATABASE	YES	YES	YES	
PERFORM ROLLFORWARD RECOVERY	YES	YES	YES	
START/STOP DATABASE INSTANCE	YES	YES	YES	
RESTORE TABLE SPACE	YES	YES	YES	
RUN TRACE	YES	YES	YES	
TAKE DBM or DB SNAPSHOTS	YES	YES	YES	
QUERY TABLE SPACE STATE	YES	YES	YES	YES
UPDATE LOG HISTORY FILES	YES	YES	YES	YES
QUIESCE TABLE SPACE	YES	YES	YES	YES
REORG TABLE	YES	YES	YES	YES
RUN RUNSTATS UTILITY	YES	YES	YES	YES
READ LOG FILES	YES			YES
CREATE/ACTIVATE/DROP EVENT MONITORS	YES			YES

Figure 1205A ··· 권한별로 사용 가능한 명령어

Tip

SYSADM,SYSCTRL,SYSMAINT 권한을 가진 사용자는 SYSMON 권한을 자동으로 소유합니다.

2 SYSMON 권한을 사용하여 사용자는 다음과 같은 명령어를 사용할 수 있습니다

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

Figure 1205B ··· SYSMON 권한으로 가능한 명령어

Tip

SYSADM,DBADM 권한을 가진 사용자는 LOAD 권한을 자동으로 소유합니다.

3 LOAD 권한을 사용하여 사용자는 다음과 같은 명령어를 사용할 수 있습니다

- LOAD
- LIST TABLESPACES
- RUNSTATS
- QUIESCE TABLESPACES FOR TABLE

Figure 1205C ··· LOAD 권한으로 가능한 명령어

TOPIC 12 05 권한별 기능

Point

SYSADM, SYSCTRL, SYSMAINT, SYSMON 권한은 데이터베이스 시스템과 관련된 명령어를 실행합니다. SYSADM 또는 DBADM, LOAD 권한의 소유자는 데이터베이스와 관련된 명령어를 실행합니다.

Tip

SYSADM, DBADM 권한을 가진 사용자는 데이터베이스에 접속하여 SQL문으로 데이터를 액세스할 수 있습니다.

1 권한별로 실행 가능한 명령어는 다음과 같습니다.

Function	SYSADM	SYSCTRL	SYSMAINT	DBADM
MIGRATE DATABASE	YES			
UPDATE DBM CFG	YES			
GRANT/REVOKE DBADM	YES			
UPDATE db/node/dcs directories	YES	YES		
FORCE USERS OFF SYSTEM	YES	YES		
CREATE/DROP DATABASE	YES	YES		
CREATE/DROP/ALTER TABLE SPACE	YES	YES		
RESTORE TO NEW DATABASE	YES	YES		
UPDATE DB CFG	YES	YES	YES	
BACKUP DATABASE or TABLE SPACE	YES	YES	YES	
RESTORE TO EXISTING DATABASE	YES	YES	YES	
PERFORM ROLLFORWARD RECOVERY	YES	YES	YES	
START/STOP DATABASE INSTANCE	YES	YES	YES	
RESTORE TABLE SPACE	YES	YES	YES	
RUN TRACE	YES	YES	YES	
TAKE DBM or DB SNAPSHOTS	YES	YES	YES	
QUERY TABLE SPACE STATE	YES	YES	YES	YES
UPDATE LOG HISTORY FILES	YES	YES	YES	YES
QUIESCE TABLE SPACE	YES	YES	YES	YES
REORG TABLE	YES	YES	YES	YES
RUN RUNSTATS UTILITY	YES	YES	YES	YES
READ LOG FILES	YES			YES
CREATE/ACTIVATE/DROP EVENT MONITORS	YES			YES

Figure 1205A ··· 권한별로 사용 가능한 명령어

Tip

SYSADM,SYSCTRL,SYSMAINT 권한을 가진 사용자는 SYSMON 권한을 자동으로 소유합니다.

2 SYSMON 권한을 사용하여 사용자는 다음과 같은 명령어를 사용할 수 있습니다

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

Figure 1205B ··· SYSMON 권한으로 가능한 명령어

Tip

SYSADM,DBADM 권한을 가진 사용자는 LOAD 권한을 자동으로 소유합니다.

3 LOAD 권한을 사용하여 사용자는 다음과 같은 명령어를 사용할 수 있습니다

- LOAD
- LIST TABLESPACES
- RUNSTATS
- QUIESCE TABLESPACES FOR TABLE

Figure 1205C ··· LOAD 권한으로 가능한 명령어

TOPIC 12 06 인스턴스 권한 제어 방법

Point

시스템 권한은 OS의 그룹에게 부여하므로, 해당 그룹에 속하는 사용자는 모두 동일한 인스턴스 권한을 가지게 됩니다. 인스턴스 구성 변수인 SYSADM_GROUP, SYSCTRL_GROUP, SYSMAINT_GROUP, SYSMON_GROUP으로 제어합니다.

Tip

SYSADM_GROUP 구성 변수에 값이 지정되지 않은 경우에는 인스턴스 사용자의 일차 그룹에 속하는 사용자들이 SYSADM 권한을 가지게 됩니다.

Tip

NULL 은 인스턴스 구성 변수와 데이터베이스 구성 변수의 값을 지정되지 않은 상태로 설정하는 특수한 키워드로 반드시 대문자로 표현합니다.

Tip

지정할 수 있는 그룹명의 최대 길이는 30 글자입니다. 그룹명의 최대 길이는 각 플랫폼에 따라 다를 수 있으며, AIX에서는 8글자까지 가능합니다.

1 SYSADM 권한을 가진 사용자로 로그인합니다.

```
$ login <인스턴스 사용자>
```

2 update dbm cfg 명령어를 이용하여 인스턴스 구성 변수인 SYSADM_GROUP, SYSCTRL_GROUP, SYSMAINT_GROUP, SYSMON_GROUP의 값을 OS에 정의된 <그룹명>으로 지정합니다. 지정한 그룹에 속한 사용자는 간접적으로 권한을 부여 받습니다.

```
$ db2 update dbm cfg using SYSCTRL_GROUP <그룹명>
```

3 update dbm cfg 명령어를 이용하여 인스턴스 구성 변수인 SYSADM_GROUP, SYSCTRL_GROUP, SYSMAINT_GROUP, SYSMON_GROUP의 값을 NULL로 지정하면 기본값으로 복귀하므로, 해당 그룹은 더 이상 인스턴스 권한을 가질 수 없습니다.

```
$ db2 update dbm cfg using SYSCTRL_GROUP NULL
```

4 각 구성 변수의 변경은 인스턴스를 재시작해야 반영됩니다.

```
$ db2stop force
$ db2start
```

5 get dbm cfg 명령어를 이용하여 인스턴스 권한과 관련된 구성 변수의 값을 확인합니다.

```
$ db2 get dbm cfg | grep "_GROUP"
 SYSADM 그룹 이름                (SYSADM_GROUP) = ADMGRP
 SYSCTRL 그룹 이름               (SYSCTRL_GROUP) = CTRLGRP
 SYSMAINT 그룹 이름              (SYSMAINT_GROUP) = MAINTGRP
 SYSMON 그룹 이름                (SYSMON_GROUP) = MOMGRP
```

TOPIC 1207 데이터베이스 권한 제어 방법

Point

데이터베이스 권한은 OS에 정의된 그룹명 또는 사용자 계정에게 부여됩니다. 그룹에게 권한이 부여되면, 해당 그룹에 속하는 사용자 계정은 모두 동일한 데이터베이스 권한을 가지게 됩니다.

1 SYSADM 권한을 가진 사용자로 데이터베이스에 접속합니다.

```
$ login <인스턴스 사용자>
$ db2 connect to <데이터베이스명>
```

2 grant문으로 DBADM과 LOAD 권한을 사용자에게 부여합니다.

```
$ db2 GRANT DBADM ON DATABASE TO USER <사용자명>
$ db2 GRANT LOAD ON DATABASE TO USER <사용자명>
```

3 grant문으로 DBADM과 LOAD 권한을 그룹에 부여하면, 그룹에 속한 사용자는 간접적으로 권한을 부여 받습니다.

```
$ db2 GRANT DBADM ON DATABASE TO GROUP <그룹명>
$ db2 GRANT LOAD ON DATABASE TO GROUP <그룹명>
```

4 revoke 문으로 DBADM과 LOAD 권한을 제거할 수 있습니다.

```
$ db2 REVOKE DBADM ON DATABASE FROM USER <사용자명>
$ db2 REVOKE LOAD ON DATABASE FROM USER <사용자명>
$ db2 REVOKE DBADM ON DATABASE FROM GROUP <그룹명>
$ db2 REVOKE LOAD ON DATABASE FROM GROUP <그룹명>
```

5 DBADM과 LOAD 권한은 SYSCAT.DBAUTH 뷰를 통해 확인합니다.

```
$ db2 "select grantee, DBADM, LOAD from syscat.dbauth"
```

Tip

DBADM 권한을 부여하면 데이터베이스에 대한 특권인 BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, IMPLICIT_SCHEMA 등이 자동으로 부여됩니다.

Tip

LOAD 권한을 부여 받은 사용자는 실제로 LOAD 유틸리티를 실행할 때, 해당 테이블에 대한 INSERT 권한도 있어야 합니다.

Tip

DBADM 권한을 제거해도 간접적으로 부여받은 데이터베이스에 대한 특권은 자동으로 제거되지 않으므로, 필요시 revoke문으로 명시적으로 제거합니다.

TOPIC 12 08 특권

Point

데이터베이스, 테이블스페이스, 스키마, 테이블, 뷰, 인덱스, 패키지, SP, UDF 등의 데이터베이스 오브젝트에 대한 구체적인 SQL 문을 실행할 수 있는 능력을 특권이라고 합니다. grant문과 revoke 문을 이용하여 그룹 또는 사용자별로 특권을 제어합니다.

Tip

오브젝트를 생성한 사용자를 '오브젝트의 소유자' 라고 합니다. 소유자는 DROP문으로 해당 오브젝트를 제거하거나, GRANT문과 REVOKE 문으로 해당 특권을 제어할 수 있는 할 수 있는 CONTROL 특권을 부여받습니다.

1 데이터베이스 오브젝트별 특권의 체계는 다음과 같이 분류됩니다.

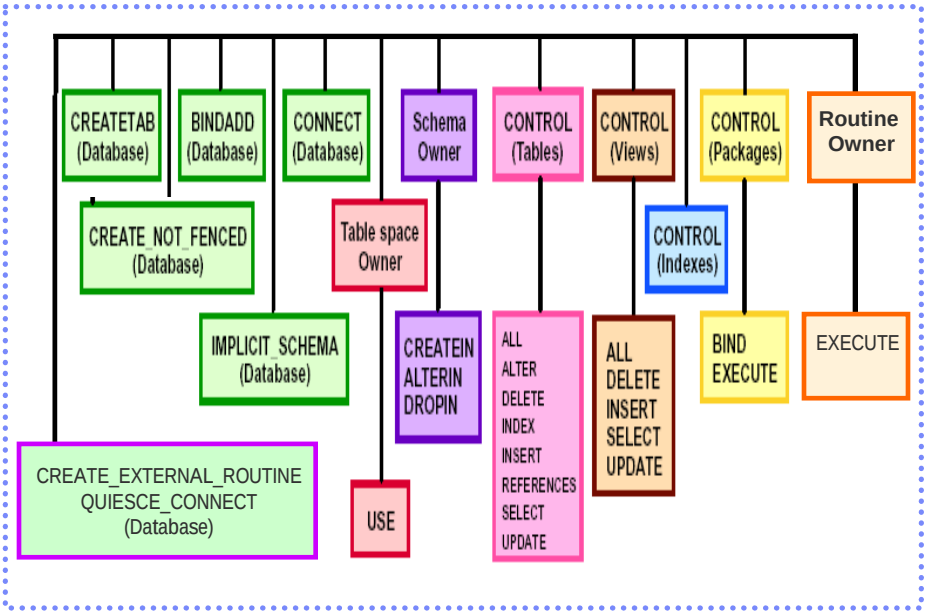


Figure 1208A ••• 데이터베이스 오브젝트별 특권

Tip

그룹에 부여된 특권은 정적 SQL 문을 실행할 때는 적용되지 않습니다. 응용프로그램에 포함된 정적 SQL문을 실행하는 사용자는 해당 SQL문에 대한 명시적인 특권을 가지고 있어야 합니다.

2 오브젝트별로 구체적인 특권명은 다음과 같으며, grant 문과 revoke문을 이용하여 사용자 또는 그룹 단위로 제어할 수 있습니다.

오브젝트	설명
데이터베이스	CONNECT, CREATETAB, IMPLICIT_SCHEMA, BINDADD, CREATE_NOT_FENCED, QUIESCE_CONNECT, CREATE_EXTERNAL_ROUTINE 특권이 있습니다.
테이블스페이스	USE 특권이 있습니다.
스키마	CREATEIN, ALTERIN, DROPIN 특권이 있습니다.
테이블	ALTER, INDEX, SELECT, INSERT, UPDATE, DELETE, REFERENCES 특권이 있습니다.
뷰	SELECT, INSERT, UPDATE, DELETE 특권이 있습니다.
인덱스	CONTROL 등이 있습니다.
패키지	BIND, EXECUTE 특권이 있습니다.
SP, UDF	EXECUTE 특권이 있습니다.
시퀀스	USAGE, ALTER 특권이 있습니다.

TOPIC 12 09 특권 제어 방법

Point

특권은 GRSANT 문과 REVOKE 문을 이용하여 사용자 또는 그룹별로 제어합니다. 특정 그룹에 특권을 부여하면, 해당 그룹에 속하는 사용자 계정은 동일한 특권을 간접적으로 부여받게 됩니다. PUBLIC은 모든 사용자를 의미하는 특별한 키워드입니다.

Tip

DBADM과 LOAD는 특권이 아닌 권한이지만, grant문과 revoke 문으로 제어합니다.

1 특권은 grant 문과 revoke 문으로 제어합니다.

* *GRANT/ REVOKE*	Database privileges	ON DATABASE	*TO/ FROM*	*USER/ GROUP*	*userid groupid PUBLIC*
	Package privileges	ON PACKAGE package_name			
	Table/view privileges	ON TABLE table/view_name			
	CONTROL	ON INDEX index_name			
	Schema privileges	ON SCHEMA schema_name			
	USE	OF TABLESPACE tablespacename			

** must be SYSADM, DBADM, or have CONTROL on object*

Figure 1209A GRANT 문과 REVOKE 문

Tip

GROUP과 USER 키워드는 생략할 수 있지만, 동일한 그룹 ID와 사용자 계정이 존재하는 경우에는 SQLSTATE 56092 가 반환되므로, 명시하는 것이 좋습니다.

2 grant 문으로 OS에 정의된 그룹 또는 사용자에게 특권을 부여합니다.

```
$ db2 grant <특권명> on <오브젝트명> to group <그룹명>
$ db2 grant <특권명> on <오브젝트명> to user <사용자명>
```

Tip

OS 에 존재하지 않는 사용자명을 이용하여 특권을 부여할 수 있지만, 데이터베이스에 접속할 때는 OS에 정의된 사용자명을 이용하여 사용자 인증이 이루어지므로, 실제 운영시에는 의미가 없습니다.

3 한 가지 이상의 특권을 한 개의 grant 문으로 부여할 수 있습니다.

```
$ db2 grant <특권명1>, <특권명2> on <오브젝트명> to user <사용자명>
```

4 모든 사용자에게 권한을 부여하려면 PUBLIC이라는 키워드를 이용합니다.

```
$ db2 grant <특권명> on <오브젝트명> to public
```

Tip

WITH GRANT OPTION 옵션은 package, routine, schema, table, view, table space에만 적용됩니다.

5 WITH GRANT OPTION을 사용하면, 권한을 부여받은 사용자는 다른 사용자에게 부여받은 권한을 전달할 수 있습니다.

```
$ db2 grant <특권명> on <오브젝트명> to user <사용자명> with grant
    option
```

6 revoke 문으로 OS 그룹 또는 사용자에게 특권을 제거합니다.

```
$ db2 revoke <특권명> on <오브젝트명> from user <사용자명>
```

TOPIC 12.10 데이터베이스 특권

Point

CONNECT, CREATETAB, IMPLICIT_SCHEMA, BINDADD, CREATE_NOT_FENCED, QUIESCE_CONNECT, CREATE_EXTERNAL_ROUTINE 등의 특권을 SYSADM, DBADM 사용자가 GRANT 문과 REVOKE 문으로 제어합니다.

Tip

데이블스페이스에 대한 USE 특권과 데이터베이스에 대한 CREATETAB 특권이 모두 있어야 해당 테이블스페이스에 테이블을 생성할 수 있습니다.

Tip

IMPLICIT_SCHEMA 특권의 소유자가 자동으로 생성된 스키마의 소유자는 SYSIBM 이 되고, 스키마에 대한 사용 권한은 모든 사용자에게 부여됩니다.

Tip

DBADM과 BINDADD는 데이터베이스에 대한 특권으로 분류하지 않고, 권한이라고 합니다. 특권과 동일하게 GRANT, REVOKE 문으로 제어합니다.

Tip

WITH GRANT OPTION 은 지원되지 않습니다.

1 데이터베이스에 관한 특권은 다음과 같습니다.

특 권	설 명
CONNECT	접속할 수 있습니다.
BINDADD	새로운 패키지를 생성할 수 있습니다.
CREATETAB	새로운 테이블을 생성할 수 있습니다.
CREATE_NOT_FENCED	'NOT FENCED" 모드의 UDF 또는 SP를 생성할 수 있습니다.
IMPLICIT_SCHEMA	존재하지 않는 스키마명을 이용하여 데이터베이스 오브젝트를 생성하면, 스키마가 생성됩니다.
QUIESCE_CONNECT	데이터베이스가 QUIESCE 모드에 있는 경우에도 접속할 수 있습니다.
CREATE_EXTERNAL_ROUTINE	SQL 프로시저 이외의 외부 소스 프로시저를 생성할 수 있습니다.

2 <특권명>을 지정하여 그룹이나 사용자별로 제어합니다.

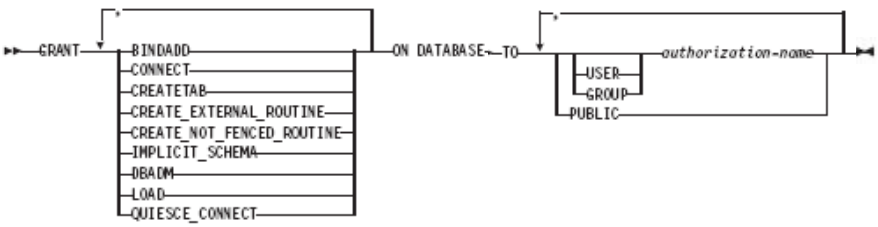


Figure 1210A ••• 데이터베이스 특권을 제어하는 GRANT 문

3 grant 문과 revoke 문으로 그룹 또는 사용자에게 부여합니다.

```
$ db2 grant CONNECT on database to GROUP <그룹명>
$ db2 grant CONNECT on database to USER <사용자명>
$ db2 grant CONNECT on database to PUBLIC
```

4 데이터베이스에 관한 특권은 카탈로그 뷰인 SYSCAT.DBAUTH 를 이용하여 확인합니다.

```
$ db2 "select * from SYSCAT.DBAUTH"
```

TOPIC 12.11 테이블스페이스 특권

Point

사용자는 USE 특권이 있어야 특정 테이블스페이스에 테이블을 생성할 수 있습니다. SYSADM, SYSCTRL 또는 DBADM 권한이 있는 사용자가 GRANT 문과 REVOKE 문으로 제어합니다.

Tip

SYSCATSPACE와 시스템 임시 테이블스페이스는 USE 권한을 부여할 수 없습니다.

Tip

데이터베이스를 생성하면 USERSPACE1에 대한 USE 권한은 모든 사용자에게 부여됩니다.

Tip

테이블스페이스에 대한 USE 특권과 데이터베이스에 대한 CREATETAB 특권이 모두 있어야 해당 테이블스페이스에 테이블을 생성할 수 있습니다.

1 테이블스페이스에 관한 특권은 다음과 같습니다.

특 권	설 명
USE	테이블스페이스에 새로운 테이블을 생성할 수 있습니다.

2 <특권명>과 <테이블스페이스명>을 지정하여 그룹이나 사용자별로 제어합니다.

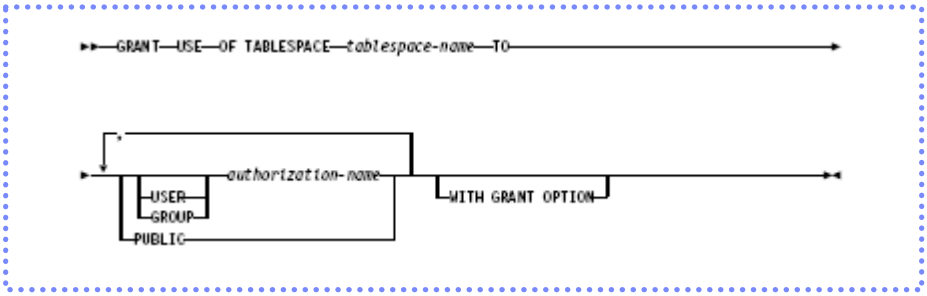


Figure 1211A • • • 테이블스페이스 특권을 제어하는 GRANT 문

3 grant 문과 revoke 문으로 그룹 또는 사용자에게 부여합니다.

```
$ db2 grant USE OF TABLESPACE <테이블스페이스명> to GROUP <그룹명>
$ db2 grant USE OF TABLESPACE <테이블스페이스명> to USER <사용자명>
$ db2 grant USE OF TABLESPACE <테이블스페이스명> to PUBLIC
$ db2 grant USE OF TABLESPACE <테이블스페이스명> to USER <사용자명>
    WITH GRANT OPTION
```

4 테이블스페이스에 관한 특권은 SYSCAT.TBSPACEAUTH 뷰를 이용하여 확인합니다.

```
$ db2 “select * from SYSCAT.TBSPACEAUTH ”
```

TOPIC 12.12 스키마 특권

Point

CREATEIN, ALTERIN, DROPIN 특권이 있습니다. SYSADM 또는 DBADM 권한이 있는 사용자가 데이터베이스에 접속한 후 GRANT 문과 REVOKE 문으로 제어합니다.

Tip

DROPIN 특권은 스키마를 제거할 수 있는 특권이 아닙니다. 스키마는 SYSADM,DBADM 또는 스키마를 생성한 소유자만 제거할 수 있습니다.

1 스키마에 관한 특권은 다음과 같습니다.

특 권	설 명
CREATEIN	특정한 스키마명을 이용하여 새로운 오브젝트를 생성합니다.
ALTERIN	특정한 스키마명을 가진 오브젝트를 변경할 수 있습니다.
DROPIN	특정한 스키마명을 가진 오브젝트를 제거합니다.

2 <특권명>과 <스키마명>을 지정하여 그룹이나 사용자별로 제어합니다.

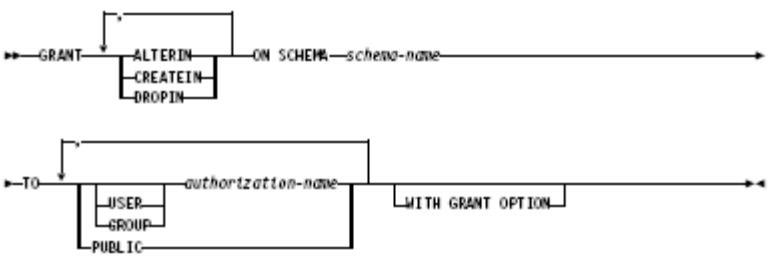


Figure 12.12A ••• 스키마 특권을 제어하는 GRANT 문

3 grant 문과 revoke 문으로 그룹 또는 사용자에게 부여합니다.

```
$ db2 grant ALTERIN on schema <스키마명> to GROUP <그룹명>
$ db2 grant DROPIN on schema <스키마명> to USER <사용자명>
$ db2 grant CREATEIN on schema <스키마명> to PUBLIC
$ db2 grant DROPIN on schema <스키마명> to USER <사용자명> WITH
    GRANT OPTION
```

4 스키마에 관한 특권은 SYSCAT.SCHEMAAUTH 뷰를 이용하여 확인합니다.

```
$ db2 “select * from SYSCAT.SCHEMAAUTH”
```

TOPIC 12 13 테이블 특권

Point

ALL, CONTROL, ALTER, INDEX, SELECT, INSERT, UPDATE, DELETE, REFERENCES 등이 있습니다. SYSADM 또는 DBADM 권한이 있는 사용자가 데이터베이스에 접속한 후 GRANT 문과 REVOKE 문으로 제어합니다.

Tip

REFERENCES 특권과 UPDATE 특권은 컬럼별 지정이 가능합니다.

1 테이블에 관한 특권은 다음과 같습니다.

특 권	설 명
CONTROL	테이블의 소유자는 테이블에 대한 모든 특권을 가집니다. CONTROL 특권을 제외한 모든 특권을 다른 사용자에게 부여할 수 있습니다. 테이블을 제거할 수 있습니다.
ALL	CONTROL 특권을 제외한 모든 특권을 가집니다.
ALTER	기존 테이블에 컬럼 추가,기본키와 고유키 추가, 컬럼 점검 조건을 추가 또는 제거할 수 있고, 트리거를 작성할 수 있습니다.
DELETE	테이블에서 행을 삭제합니다.
INDEX	테이블에 인덱스를 추가합니다.
INSERT	테이블에 행을 추가합니다.
REFERENCES	RI 관계를 정의할 때 parent table 로 사용될 수 있습니다.
SELECT	테이블에서 데이터를 조회할 수 있습니다. export 유틸리티에 사용되는 SELECT문을 요청할 수 있습니다.
UPDATE	테이블의 컬럼값을 변경할 수 있습니다. 특정 컬럼에 대한 UPDATE 특권을 부여할 수도 있습니다.

2 <특권명>과 <테이블명>을 지정하여 그룹이나 사용자별로 제어합니다.

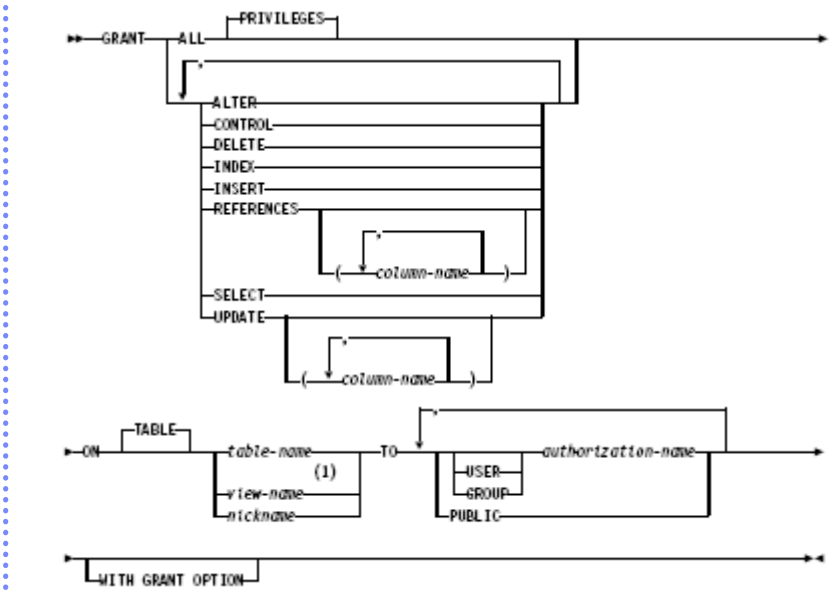


Figure 1213A ••• 테이블과 뷰 특권을 제어하는 GRANT 문

3 테이블에 관한 특권은 SYSCAT.TABAUTH 뷰를 이용하여 확인합니다.

```
$ db2 “select * from SYSCAT.TABAUTH”
```

TOPIC 12.14 뷰 특권

Point

뷰에 대한 특권으로는 ALL, CONTROL, SELECT, INSERT, UPDATE, DELETE 등이 있습니다. SYSADM 또는 DBADM 권한이 있는 사용자가 데이터베이스에 접속한 후 GRANT 문과 REVOKE 문으로 제어합니다.

Tip

뷰의 정의에 사용된 테이블에 대한 특권이 있으면, 뷰에 대한 특권도 자동으로 부여됩니다.

1 뷰에 관한 특권은 다음과 같습니다.

특 권	설 명
CONTROL	뷰의 소유자에게 자동으로 부여되는 특권으로 뷰에 대한 모든 특권을 가집니다. CONTROL 특권을 제외한 모든 특권을 다른 사용자에게 부여할 수 있습니다. 뷰를 제거할 수 있습니다.
ALL	CONTROL 특권을 제외한 모든 특권을 가집니다.
DELETE	뷰에서 행을 삭제합니다.
INSERT	뷰에 행을 추가합니다.
SELECT	뷰에서 데이터를 조회할 수 있습니다. export 유틸리티에 사용되는 SELECT문을 요청할 수 있습니다.
UPDATE	뷰의 컬럼값을 변경할 수 있습니다. 특정 컬럼에 대한 UPDATE 특권을 부여할 수도 있습니다.

2 테이블과 뷰는 동일한 방법으로 grant 문과 revoke 문으로 <특권명>과 <뷰명>을 지정하여 그룹이나 사용자별로 제어합니다.

```
$ db2 grant UPDATE on table <테이블명> to GROUP <그룹명>
$ db2 grant UPDATE (<컬럼명>) on table <테이블명> to USER <사용자명>
$ db2 grant UPDATE on table <테이블명> to USER <사용자명>
$ db2 grant UPDATE on table <테이블명> to USER <사용자명> WITH
    GRANT OPTION
$ db2 grant UPDATE on table <뷰명> to PUBLIC
```

3 뷰에 관한 특권은 SYSCAT.TABAUTH 뷰를 이용하여 확인합니다.

```
$ db2 “select * from SYSCAT.TABAUTH”
```

TOPIC 12.15 인덱스 특권

Point

CONTROL 특권이 있는 사용자는 DROP INDEX 문을 이용하여 해당 인덱스를 제거할 수 있습니다. SYSADM 또는 DBADM 권한이 있는 사용자가 데이터베이스에 접속한 후 GRANT 문과 REVOKE 문으로 제어합니다.

Tip

인덱스에 대한 사용 특권은 필요하지 않습니다. 테이블에 대한 SELECT특권이 있는 경우에는 인덱스도 사용될 수 있습니다.

1 인덱스에 관한 특권은 다음과 같습니다.

특 권	설 명
CONTROL	인덱스의 소유자에게 자동으로 부여되는 특권으로 인덱스를 제거할 수 있습니다.

2 <특권명>과 <인덱스명>을 지정하여 그룹이나 사용자별로 제어합니다.

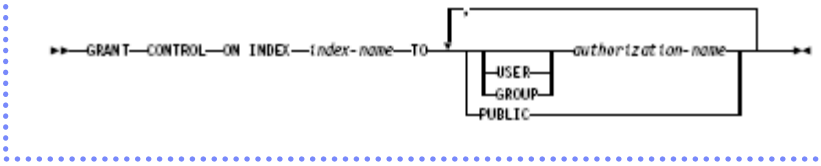


Figure 1215A 인덱스 특권을 제어하는 GRANT 문

Tip

WITH GRANT OPTION 은 지원되지 않습니다.

3 grant 문과 revoke 문으로 그룹이나 사용자별로 제어합니다.

```
$ db2 grant CONTROL on index <인덱스명> to GROUP <그룹명>
$ db2 grant CONTROL on index <인덱스명> to USER <사용자명>
$ db2 grant CONTROL on index <인덱스명> to PUBLIC
```

4 인덱스에 관한 특권은 SYSCAT.INDEXAUTH 뷰를 이용하여 확인합니다.

```
$ db2 “select * from SYSCAT.INDEXAUTH”
```

TOPIC 12.16 패키지 특권

Point

CONTROL, BIND, EXECUTE 특권이 있습니다. SYSADM 또는 DBADM 권한이 있는 사용자가 데이터베이스에 접속한 후 GRANT 문과 REVOKE 문으로 제어합니다.

Tip

패키지는 SQL문에 대한 액세스 플랜을 가지고 있는 데이터베이스 오브젝트로, 시스템 카탈로그에 저장됩니다.

Tip

패키지에 대한 EXECUTE 특권을 부여받으면, 패키지를 실행하는 동안에는 패키지에 포함된 정적 SQL문에 대한 특권을 임시로 가지게 되므로, 해당 SQL문에 대한 명시적인 특권이 없는 사용자도 패키지를 실행할 수 있습니다.

1 패키지에 관한 특권은 다음과 같습니다.

특 권	설 명
CONTROL	패키지의 소유자에게 자동으로 부여되는 특권으로 패키지에 대한 모든 특권을 가집니다. 패키지에 대한 drop, rebind, execute 특권을 가지며, CONTROL 특권을 제외한 모든 특권을 다른 사용자에게 부여할 수 있습니다.
BIND	기존의 패키지를 리바인드할 수 있습니다.
EXECUTE	패키지를 실행할 수 있습니다.

2 <특권명>과 <패키지명>을 지정하여 그룹이나 사용자별로 제어합니다.

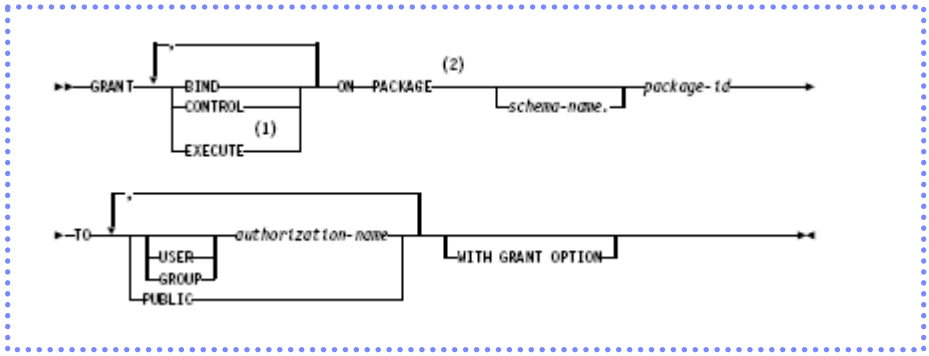


Figure 12.16A ••• 패키지 특권을 제어하는 GRANT 문

2 grant 문과 revoke 문으로 그룹이나 사용자별로 제어합니다.

```
$ db2 grant EXECUTE on package <패키지명> to GROUP <그룹명>
$ db2 grant EXECUTE on package <패키지명> to USER <사용자명>
$ db2 grant EXECUTE on package <패키지명> to PUBLIC
$ db2 grant EXECUTE on package <패키지명> to USER <사용자명> WITH
    GRANT OPTION
```

3 패키지에 관한 특권은 SYSCAT.PACKAGEAUTH 뷰를 이용하여 확인합니다.

```
$ db2 “select * from SYSCAT.PACKAGEAUTH”
```

TOPIC 12.17 루틴 특권

Point

SP(저장 프로시저) 또는 UDF(사용자 정의 함수) 등의 루틴에 대한 특권으로는 EXECUTE 가 있습니다. SYSADM 또는 DBADM 권한이 있는 사용자가 데이터베이스에 접속한 후 GRANT 문과 REVOKE 문으로 제어합니다.

1 UDF 또는 SP 에 관한 특권은 다음과 같습니다.

특 권	설 명
EXECUTE	CALL 문을 이용하여 저장프로시저를 호출하거나, SQL문에서 UDF를 참조할 수 있습니다.

Tip

오버로딩된 UDF 또는 SP인 경우에는 SYSCAT.ROUTINES 뷰에서 고유한 이름인 SPECIFIC 명을 확인하여 사용합니다.

2 <UDF명> 또는 <SP명> 을 지정하여 그룹이나 사용자별로 제어합니다.

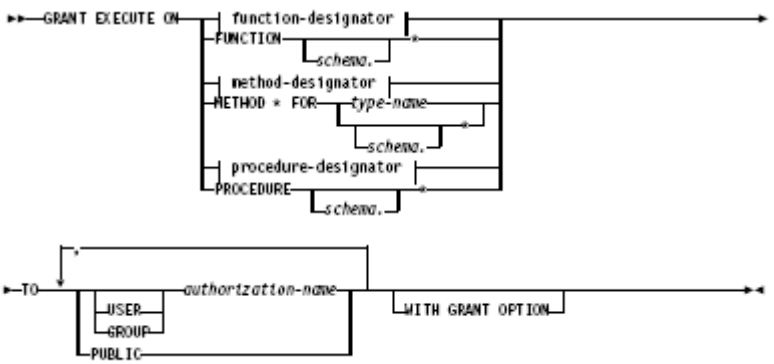


Figure 1217A ••• 루틴 특권을 제어하는 GRANT 문

Tip

<스키마명>.* (asterisk)는 지정한 <스키마명>을 가지는 모든 UDF 또는 SP를 지정하는 특수한 옵션입니다.

3 grant 문과 revoke 문으로 그룹이나 사용자별로 제어합니다.

```
$ db2 grant EXECUTE on FUNCTION <UDF명> to GROUP <그룹명>
$ db2 grant EXECUTE on FUNCTION <UDF명> to USER <사용자명>
$ db2 grant EXECUTE on FUNCTION <UDF명> to PUBLIC
$ db2 grant EXECUTE on FUNCTION <스키마명>.*  to USER <사용자명>
$ db2 grant EXECUTE on FUNCTION <UDF명> to USER <사용자명> WITH
    GRANT OPTION
$ db2 grant EXECUTE on SPECIFIC FUNCTION <UDF에 대한 SPECIFIC명>
    to USER <사용자명>
$ db2 grant EXECUTE on PROCEDURE <SP명> to GROUP <그룹명>
$ db2 grant EXECUTE on PROCEDURE <SP명> to USER <사용자명>
$ db2 grant EXECUTE on PROCEDURE <SP명> to PUBLIC
$ db2 grant EXECUTE on PROCEDURE <스키마명>.* to USER <사용자명>
$ db2 grant EXECUTE on PROCEDURE <SP명> to USER <사용자명> WITH
    GRANT OPTION
$ db2 grant EXECUTE on SPECIFIC PROCEDURE <SP에 대한 SPECIFIC명>
    to USER <사용자명>
```

4 루틴에 관한 특권은 SYSCAT.ROUTINEAUTH 뷰를 이용하여 확인합니다.

```
$ db2 “select * from SYSCAT.ROUTINEAUTH”
```

TOPIC 12-18 시퀀스 특권

Point

ALTER, USAGE 등이 있습니다. SYSADM 또는 DBADM 권한이 있는 사용자 또는 소유자가 데이터베이스에 접속한 후 GRANT 문과 REVOKE 문으로 제어합니다.

1. 시퀀스에 관한 특권은 다음과 같습니다.

특 권	설 명
USAGE	시퀀스에 대한 NEXTVAL과 PREVVAL 표현식을 사용할 수 있습니다.
ALTER	ALTER SEQUENCE 문을 이용하여 시퀀스의 정의를 변경할 수 있습니다.

2. <특권명>과 <시퀀스명> 을 지정하여 그룹이나 사용자별로 제어합니다.

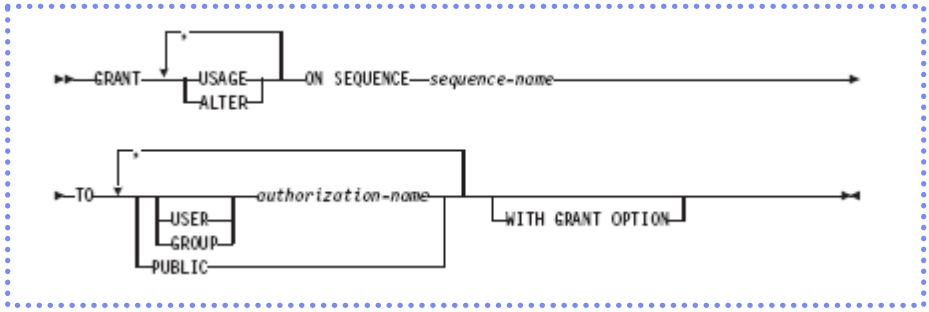


Figure 1218A ··· 시퀀스 특권을 제어하는 GRANT 문

2. grant 문과 revoke 문으로 그룹이나 사용자별로 제어합니다.

```
$ db2 grant USAGE on sequence <시퀀스명> to GROUP <그룹명>
$ db2 grant USAGE on sequence <시퀀스명> to USER <사용자명>
$ db2 grant USAGE on sequence <시퀀스명> to PUBLIC
$ db2 grant USAGE on sequence <시퀀스명> to USER <사용자명> WITH
    GRANT OPTION
```

3. 시퀀스에 관한 특권은 SYSCAT.SEQUENCEAUTH 뷰를 이용하여 확인합니다.

```
$ db2 “select * from SYSCAT.SEQUENCEAUTH”
```

TOPIC 12-19 간접 권한과 특권

Point

사용자 또는 그룹이 GRANT 문을 통하지 않고 부여 받은 권한과 특권입니다. 정적 SQL문을 제외한 모든 환경에서 직접 부여 받은 경우와 동일하게 사용할 수 있습니다.

Tip

간접적으로 부여받은 권한과 특권은 revoke 명령어를 이용하여 명시적으로 제거할 때까지 유지됩니다.

Tip

실제 운영 환경에서는 revoke 문을 이용하여 기본적으로 부여된 모든 간접 특권을 제거하고, grant 문을 이용하여 필요한 특권을 다시 제어하도록 합니다.

Tip

명시적인 특권이 없으므로 동일한 SQL문을 패키지의 외부에서 직접 실행할 수는 없습니다.

1 데이터베이스를 생성한 사용자는 해당 데이터베이스에 대한 DBADM 권한을 가지게 됩니다.

```
$ db2 create db <데이터베이스명>
$ db2 “select * from syscat.abauth”
```

2 데이터베이스를 생성하면 간접적으로 다음의 특권이 PUBLIC에게 부여됩니다.

대 상	특 권
데이터베이스	CONNECT, CREATETAB, BINDADD, IMPLICIT_SCHEMA 특권
USERSPACE1 테이블스페이스	USE 특권
시스템 카탈로그 테이블	SELECT 특권
기본 패키지	BIND, EXECUTE 특권
SYSFUN 스키마를 가진 함수	EXECUTE WITH GRANT 특권

3 DBADM 권한을 부여받은 그룹 또는 사용자는 간접적으로 데이터베이스에 대한 모든 특권을 부여받습니다.

```
$ db2 connect to <데이터베이스명>
$ db2 grant DBADM on database to user <사용자명>
$ db2 “select * from syscat.abauth"| grep –i <사용자명>
```

4 테이블, 인덱스, 패키지 등의 데이터베이스 오브젝트를 생성한 사용자는 오브젝트의 소유자로서 CONTROL 특권을 가지게 됩니다.

```
$ db2 connect to <데이터베이스명>
$ db2 "create table <테이블명> (<컬럼명> <데이터유형명>)"
$ db2 “select * from syscat.tabauth” | grep –i <사용자명>
```

5 패키지에 대한 EXECUTE 특권을 부여받으면, 패키지를 실행하는 동안에는 패키지에 포함된 정적 SQL문에 대한 특권을 임시로 가지게 되므로, 해당 SQL문에 대한 명시적인 특권이 없는 사용자도 패키지를 실행할 때는 해당 SQL문을 실행할 수 있습니다.

```
$ db2 connect to <데이터베이스명>
$ db2 grant execute on package <패키지명> to user <사용자명>
$ db2 “select * from syscat.packageauth” | grep –i <사용자명>
```

UNIT 13

백업과 복구

백업 방법으로 데이터베이스 백업과 온라인, 테이블스페이스, 인크리멘탈, 델타 백업 등을 소개합니다. 복구방법으로 크래쉬, 버전, 롤포워드 복구 방법이 소개되며, 백업 이미지 파일과 아카이브 로그 파일을 보관한다면, 손상 직전 시점까지 데이터베이스를 복구할 수 있습니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 데이터베이스 로깅
- 데이터베이스 로그를 위한 구성 변수
- 순환 로깅
- 아카이브 로깅
- USER EXIT
- 복구 기록 파일
- LIST HISTORY 명령어
- PRUNE HISTORY 명령어
- 백업의 종류
- BACKUP DB 명령어
- 백업 이미지 파일
- FULL 백업
- INCREMENTAL 백업
- DELTA 백업
- 테이블스페이스 백업
- 복구의 종류
- RESTART DB 명령어
- RESTORE DB 명령어
- ROLLFORWARD DB 명령어
- 크래쉬 복구
- 버전 복구
- 경로 재지정 복구
- 롤포워드 복구
- INCREMENTAL 복구
- DELTA 복구
- 테이블스페이스의 상태
- 테이블스페이스 복구

TOPIC 1301 데이터베이스 로깅

Point

INSERT, UPDATE, DELETE문이 실행되면, 버퍼풀의 데이터가 변경됩니다. 변경 이전의 값과 변경 이후의 값은 로그 버퍼를 통해서 로그 파일에 기록되어 트랜잭션의 롤백 작업에 사용됩니다. 보관된 로그 파일은 데이터베이스 복구에 사용됩니다.

Tip

데이터베이스 단위로 로그 파일을 운영합니다. 한 데이터베이스에 대해 실행되는 모든 트랜잭션은 공통으로 사용되는 로그 파일에 로깅을 합니다.

Tip

새로운 SQL문이 요청되는 시점부터 COMMIT 또는 ROLLBACK문이 요청되는 시점까지를 한 개의 트랜잭션 또는 UOW(Unit Of Work)라고 합니다.

Tip

변경 이전의 값을 '사전(Before 또는 Old) 이미지' 라고 하며, 변경 이후의 값을 '사후(After 또는 New) 이미지' 라고 합니다.

Tip

버퍼풀에서 이미 변경되었으나, 디스크에 반영되지 않은 데이터가 포함된 페이지를 '더티(Dirty) 페이지'라고 합니다.

Tip

변경된 데이터가 디스크에 반영될 때는 반드시 로그 파일에 먼저 반영되는데, 이것을 '로그 먼저 쓰기(Log Ahead Write)' 라고 합니다.

1 데이터베이스가 활성화되면 고정된 개수의 로그 파일들이 로그 디렉토리에 새롭게 생성됩니다. SQL문의 처리에 필요한 데이터는 테이블스페이스 컨테이너에서 버퍼풀로 로드됩니다. 버퍼풀의 데이터를 변경하면 변경 내용은 로그 버퍼에 기록되어 로그 버퍼가 가득 차거나, 트랜잭션이 COMMIT 되면 로그 파일로 반영됩니다. 버퍼풀의 변경된 페이지가 지정한 비율을 초과하면 변경 내용은 로그 파일에 먼저 반영되고, 디스크의 컨테이너에 반영됩니다. 기본적으로 로그 파일 한 개 분량의 로깅이 완료되면 버퍼풀의 변경 내용이 컨테이너로 반영됩니다.

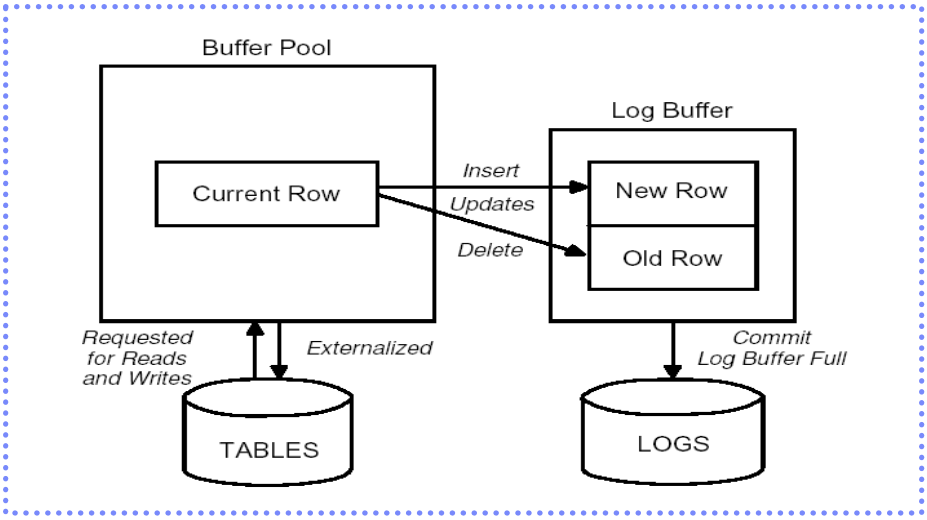


Figure 1301A ••• 버퍼풀과 데이터베이스 로깅

2 SQL문의 종류에 따라 로그 파일에 기록되는 내용이 다릅니다.

SQL문	설 명
SELECT 문	기록되지 않습니다.
INSERT 문	추가된 데이터에 대한 사후 이미지가 기록됩니다.
UPDATE 문	변경된 데이터에 대한 사전, 사후 이미지가 기록됩니다.
DELETE 문	삭제된 데이터에 대한 사전 이미지가 기록됩니다.

3 로그 파일은 용도와 상태에 따라 4가지로 구분됩니다.

로 그	설 명
활성	종료되지 않은 트랜잭션 정보를 포함하고 있는 로그
비활성 로그	종료된 트랜잭션 정보를 포함하고 있는 로그
온라인 아카이브	재사용되지 않고 현재의 활성 로그 디렉토리에 보관된 비활성 로그
오프라인 아카이브	재사용되지 않고 다른 저장 위치에 보관된 비활성 로그

4 로그 파일을 운영하는 방법에 따라 2가지의 로깅 방법으로 구분됩니다.

로 깅 방 법	설 명
순환 로깅	비활성 로그를 재사용하는 방식입니다.
아카이브 로깅	비활성 로그를 아카이브 로그로 보관하는 방식입니다.

TOPIC 13-02 데이터베이스 로그를 위한 구성 변수

Point

로깅과 관련된 환경은 LOGPRIMARY, LOGSECOND, LOGFILSIZ, LOGBUFSZ, MINCOMMIT, NEWLOGPATH, LOGRETAIN, USEREXIT, SOFTMAX 등의 데이터베이스 구성 변수를 이용하여 제어합니다.

Tip
기본 로그 디렉토리는 데이터베이스가 생성된 디렉토리의 SQLOGDIR 서브 디렉토리입니다.

Tip
NEWLOGPATH 에서 지정하는 경로명은 미리 생성되어야 합니다.변경 후에 데이터베이스가 재활성화되면 NEWLOGPATH에 지정했던 값이 <로그 파일에 대한 경로>에 표시되고, NEWLOGPATH 에는 표시되지 않습니다.

Tip
LOGRETAIN과 USEREXIT, LOGARCHMETH1 구성 변수가 모두 OFF인 경우는 순환 로깅이 사용됩니다.

Tip
LOGRETAIN 변수를 ON으로 설정하면 LOGARCHMETH1 변수가 LOGRETAIN 으로 자동적으로 설정됩니다.

Tip
USEREXIT 변수를 ON으로 설정하면 LOGARCHMETH1 변수가 USEREXIT 으로 자동적으로 설정됩니다.

Tip
SOFTMAX는 로그 파일의 크기를 이용한 백분율로 표시됩니다.

Tip
MINCOMMIT 변수의 기본값은 1 이므로 COMMIT 이 요청될 때마다 로그 버퍼의 내용이 로그 파일로 반영됩니다.

Tip
logsecond 변수의 값이 -1 이면, logprimary <= 256 입니다.
logsecond 의 값이 -1이 아니면, (logprimary + logsecond) <= 256입니다

1 데이터베이스 로그와 관련된 주요 데이터베이스 구성 변수는 다음과 같습니다.

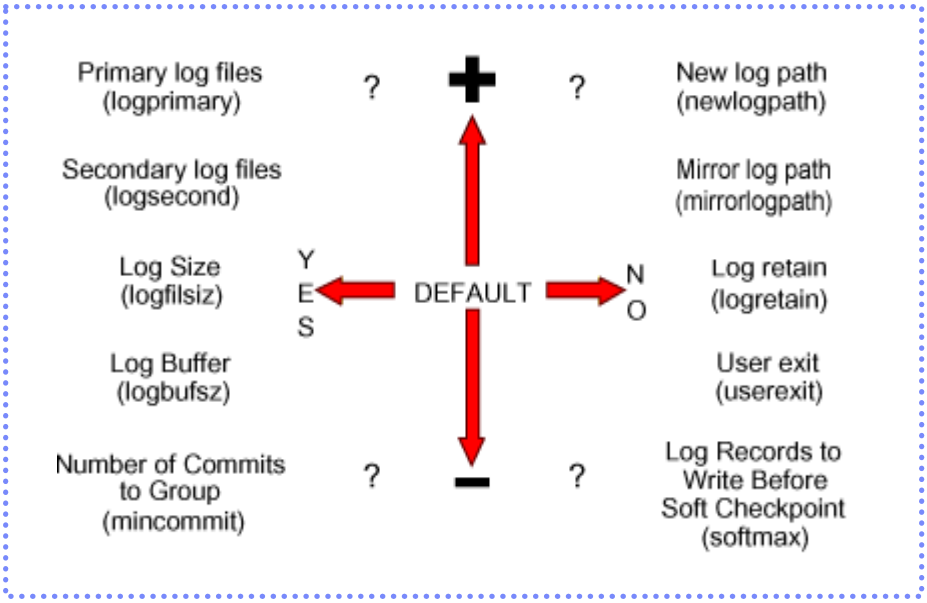


Figure 1302A • • • 로깅과 연관된 데이터베이스 구성 변수

2 UPDATA DB CFG 명령어로 변경하며, 데이터베이스가 재활성화가 필요할 수 있습니다.

구성변수	설명
LOGPRIMARY	1차 로그(Primary Log)의 개수입니다. 데이터베이스의 변경 내용을 기록하며, 데이터베이스가 활성화될 때 미리 할당됩니다.
LOGSECOND	2차 로그(Secondary Log)의 개수입니다. 1차 로그가 모두 사용되면 필요에 따라 한 개씩 할당됩니다.
LOGFILSIZ	로그 파일 한 개의 크기로 4K 페이지 단위로 설정합니다.
LOGBUFSZ	로그 파일을 위한 버퍼의 크기로 4K 페이지 단위로 설정합니다.
NEWLOGPATH	활성 로그 파일을 저장할 경로명을 지정합니다.
LOGRETAIN	비활성 로그를 아카이브 로그로 보관합니다. ON으로 설정하면, 아카이브 로깅 방식이 사용됩니다.
USEREXIT	사용자가 제공한 프로그램을 이용하여 아카이브 로그를 이동시킵니다. ON으로 설정하면, 아카이브 로깅 방식이 사용됩니다.
LOGARCHMETH1, LOGARCHMETH2	비활성 로그를 현재의 로그 디렉토리가 아닌 위치에 아카이브하도록 합니다. 기본값은 OFF이며, LOGRETAIN, USEREXIT, DISK, TSM, VENDER 중에서 설정합니다. DISK를 이용한 아카이브 경로명은 <DISK:절대경로명> 의 형태로 설정합니다.
MINCOMMIT	지정한 개수의 COMMIT 요청을 수행할 때까지 로그 파일에 기록하는 것을 지연합니다. 최대 지연 시간은 1초입니다.
SOFTMAX	응급 복구시에 사용되는 활성 로그의 양을 지정합니다.
CHNGPGS_THRESH	버퍼풀에서 디스크로 반영되지 않은 변경된 페이지의 최대 비율을 설정합니다.

TOPIC 13 03 순환 로깅

Point

고정된 개수의 로그 파일을 순환하며 재사용하는 방식입니다. 기본적으로 적용되는 로깅 방식으로 운영하는 로그 파일의 최대 개수는 LOGPRIMARY 와 LOGSECOND 데이터베이스 구성 변수에 의해 결정됩니다.

Tip

로그 파일명은 S0000000.LOG 부터 시작합니다.

Tip

활성 로그는 응급 복구시에 사용됩니다. 비활성 로그는 재사용되므로 데이터베이스 복구에 이용될 수 없습니다.

1 데이터베이스가 활성화되면 LOGPRIMARY 데이터베이스 구성 변수에 설정된 개수의 로그 파일들이 로그 디렉토리에 새롭게 생성됩니다. 생성된 로그 파일의 번호는 0 번부터 시작합니다.

```
$ db2 get db cfg for <DB명> | grep LOGPRIMARY
$ db2 get db cfg for <DB명> | grep LOGSECOND
```

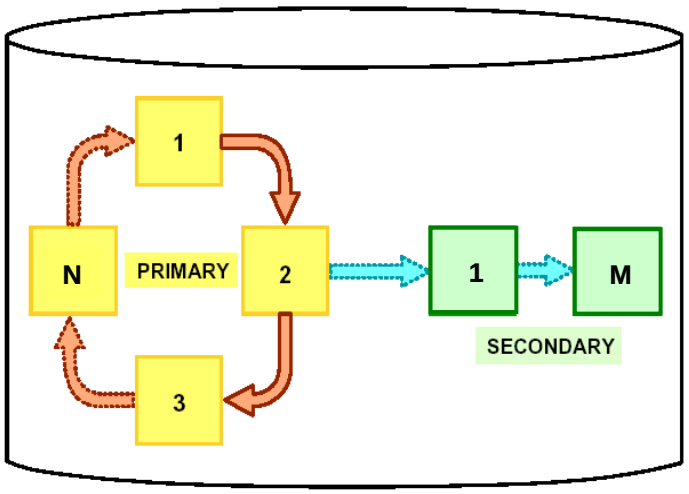


Figure 1303A 비활성 로그를 재사용하는 순환 로깅

Tip

LOGRETAIN과 USEREXIT 변수를 모두 OFF로 설정해도 순환 로깅 모드로 전환됩니다. 이전 버전과의 호환을 위해 유지하는 구성 변수이므로 사용하지 않을 것을 권장합니다.

2 update db cfg 명령어로 LOGARCHMETH1 구성 변수의 값을 OFF로 설정합니다.

```
$ db2 update db cfg for <DB명> using LOGARCHMETH1 OFF
```

Tip

LOGSECOND의 값을 -1로 설정하면 2차 로그가 무제한으로 할당됩니다. 트랜잭션의 개수나 크기에는 제한이 없어지지만, 응급 복구시에 많은 시간이 소모될 수 있으므로 주의가 필요합니다.

3 UOW가 시작되면 0번 로그는 활성 로그가 되어 기록을 시작하고, 0번을 모두 채우면 1번이 활성 로그가 되어 0번 로그에 이어서 기록을 시작합니다.

4 0번 로그에 포함된 트랜잭션들이 모두 종료되었다면, 0번 로그는 비활성 로그가 되어서 다음 번에 재사용됩니다. 0번 로그에 아직 종료되지 않은 트랜잭션이 포함되어 있다면, 0번 로그와 1번 로그는 모두 활성 로그가 됩니다.

Tip

임시로 사용되었던 2차 로그는 비활성 로그가 되면 엔진에 의해 비동기적으로 제거됩니다.

5 유사한 방법으로 현재 N번 로그가 활성 로그이고, 더 이상 기록할 공간이 없으면 다음 번호의 로그 파일을 요청합니다. 이 시점에서 0번 로그가 비활성 로그였다면, N번 로그에 뒤를 이어 재사용됩니다. 0번 로그가 활성 로그였다면, LOGSECOND에서 지정한 로그가 사용되어 전체 활성 로그의 수는 N+1 이 됩니다.

Tip

활성 로그가 LOGPRIMARY 와 LOGSECOND 의 합을 초과하면 'LOG FULL' 상태라고 합니다.

6 유사한 방법으로 LOGSECOND 가 지정한 M개의 로그가 모두 활성 로그가 된 상태에서 더 이상 기록할 공간이 없으면, 로그 공간 부족 현상이 발생하면서, 진행 중이던 트랜잭션은 모두 롤백됩니다. 즉, 활성 로그의 최대 개수는 LOGPRIMARY 구성 변수와 LOGSECOMD 구성 변수의 합을 초과할 수 없습니다.

7 LOGSECOND로 생성된 M개의 로그들이 비활성 로그가 되면, 비동기식으로 엔진에 의해 제거되고, 다시 LOGPRIMARY 의 값인 N개의 로그가 남게 됩니다.

TOPIC 1304 아카이브 로깅

Point

한 개의 로그 파일이 비활성화되면, 새로운 로그 파일을 생성하여 항상 고정된 개수의 로그 파일을 운영하는 방식입니다. 데이터베이스 구성 변수인 LOGARCHMETH1 의 값을 LOGRETAIN 으로 설정하면 됩니다.

Tip

보관된 로그 파일은 롤포워드 복구시에 사용됩니다. 보관된 로그 파일 중에서 특정 시점 이전의 불필요한 로그 파일은 제거해도 됩니다.

Tip

아카이브 로깅에서 순환 로깅으로 전환하면 로그 파일의 번호는 0 부터 다시 시작됩니다.

Tip

순환 로깅에서 아카이브 로깅으로 전환되면 복구시에 사용될 최초의 데이터베이스 백업이 필요하게 되므로, 데이터베이스는 백업 보류 'Backup Pending' 상태가 됩니다. backup db 명령어로 데이터베이스 백업을 실행하면 됩니다.

Tip

LOGRETAIN 또는 USEREXIT 변수를 ON으로 설정해도 아카이브 로깅 모드로 전환됩니다. 이전 버전과의 호환을 위해 유지하는 구성 변수이므로 사용하지 않을 것을 권장합니다.

Tip

비활성 로그는 기본적으로 활성 로그와 동일한 로그 디렉토리에 보관되는데, 이러한 로그를 '온라인 보관 로그' 라고 합니다.

Tip

사용자의 수작업 또는 user exit 프로그램에 의해 다른 저장 공간으로 이동된 비활성 로그 '오프라인 보관 로그' 라고 합니다.

Tip

활성 로그가 LOGPRIMARY 와 LOGSECOND 의 합을 초과하면 'LOG FULL' 상태라고 합니다.

1. 데이터베이스가 활성화되면 LOGPRIMARY 데이터베이스 구성 변수에 설정된 개수의 로그 파일들이 로그 디렉토리에 새롭게 생성됩니다. 생성된 로그 파일의 번호는 최근에 사용했던 로그 파일의 다음 번호입니다.

```
$ db2 get db cfg for <DB명> | grep "처음에 사용되는 로그 파일"
```

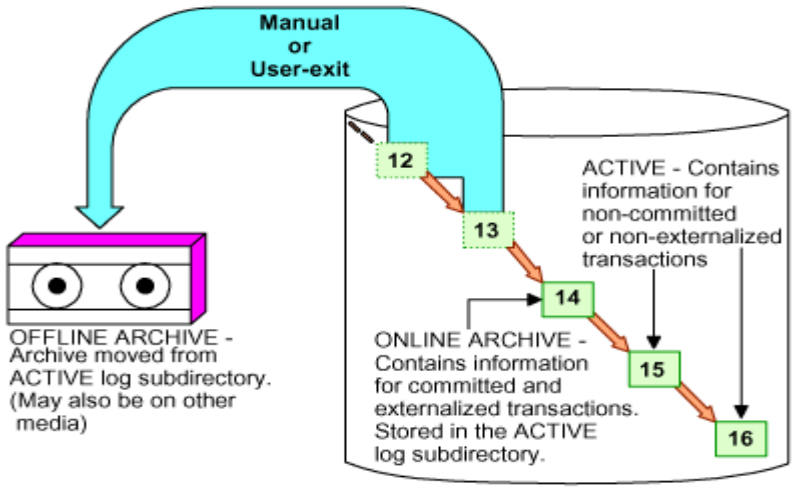


Figure 1304A ··· 온라인 아카이브 로그와 오프라인 아카이브 로그

2. update db cfg 명령어로 LOGARCHMETH1 구성 변수의 값을 LOGRETAIN 으로 설정합니다.

```
$ db2 update db cfg for <DB명> using LOGARCHMETH1 LOGRETAIN
```

3. UOW가 시작되면 0번 로그는 활성 로그가 되어 기록을 시작하고, 0번을 모두 채우면 1번이 활성 로그가 되어 0번 로그에 이어서 기록을 시작합니다.

4. 0번 로그에 포함된 트랜잭션들이 모두 종료되었다면, 0번 로그는 비활성 로그가 되어 더 이상 사용되지 않고 보관됩니다. 새로운 N+1 번의 로그 파일을 생성하여 전체 로그 파일의 개수는 N개를 유지하도록 합니다. 0번 로그에 아직 종료되지 않은 트랜잭션이 포함되어 있다면, 0번 로그와 1번 로그는 모두 활성 로그가 됩고, N+1 번 로그는 생성되지 않습니다.

5. 현재 N개의 로그가 활성 로그이고, 더 이상 기록할 공간이 없으면 LOGSECOND에서 지정한 로그가 사용되어 전체 활성 로그의 수는 N+1 이 됩니다.

6. 유사한 방법으로 LOGSECOND 가 지정한 M개의 로그가 모두 활성 로그가 된 상태에서 더 이상 기록할 공간이 없으면, 로그 공간 부족 현상이 발생하면서, 진행 중이던 트랜잭션은 모두 롤백됩니다. 즉, 활성 로그의 최대 개수는 LOGPRIMARY 구성 변수와 LOGSECOMD 구성 변수의 합을 초과할 수 없습니다.

7. LOGSECOND로 생성된 M개의 로그들이 비활성 로그가 되면, 제거되지 않고 그대로 보관됩니다. 다시 LOGPRIMARY 의 값인 N개의 로그가 남게 됩니다.

TOPIC 13 05 USER EXIT

Point

아카이브 로깅 방식을 운영할 때, 비활성 로그는 기본적으로 활성 로그와 동일한 디렉토리에 보관됩니다. LOGARCHMETH1 구성 변수를 설정하여 비활성 로그를 지정한 저장 영역으로 이동시킬 수 있습니다.

Tip

아카이브 디렉토리는 미리 생성되어 있지 않으면, SQL5099N 오류가 반환됩니다.

Tip

LOGARCHMETH1 변수는 TSM 서버와 VENDER API를 이용한 USEREXIT도 지원합니다.

Tip

deactivate db 명령어를 실행하기 전에 데이터베이스에 접속된 모든 응용프로그램은 종료되어야 합니다.

1. update db cfg 명령어로 LOGARCHMETH1 구성 변수의 값을 DISK:<경로명> 으로 설정합니다. <경로명>은 절대 경로명으로 표현합니다.

```
$ db2 update db cfg for <DB명> using LOGARCHMETH1 DISK:<경로명>
```

2. get db cfg 명령어의 show detail 옵션으로 변경값이 반영되었는지 확인합니다.

```
$ db2 connect to <DB명>
$ db2 get db cfg for <DB명> SHOW DETAIL | grep  LOGARCHMETH1
  첫 번째 로그 아카이브 메소드            (LOGARCHMETH1) = OFF
```

3. update db cfg 명령어를 실행한 후에 SQL1363W 경고를 받았다면, 변경값이 동적으로 반영되지 않았으므로 데이터베이스를 재활성화합니다.

```
$ db2 terminate
$ db2 deactivate db <DB명>
```

4. activate db 명령어로 데이터베이스를 재활성화하면 SQL1116N 오류가 반환됩니다.

```
$ db2 get db cfg for <DB명>  | grep  LOGARCHMETH1
  첫 번째 로그 아카이브 메소드            (LOGARCHMETH1) = DISK:<경로명>
$ db2 activate db <DB명>
  SQL1116N  BACKUP PENDING 때문에 데이터베이스 "<DB명>"에 연결하거나 활성화할 수 없습니다.  SQLSTATE=57019
```

5. LOGARCHMETH1 의 값이 OFF 가 아닌 경우는 아카이브로깅 모드로 설정되므로 최초의 오프라인 모드의 데이터베이스 FULL 백업이 필요합니다. backup db 명령어를 실행하여 데이터베이스의 '백업 보류' 상태를 해제하고, 다시 데이터베이스를 재활성화합니다.

```
$ db2 backup db <DB명>
$ db2 activate db <DB명>
```

6. 다양한 트랜잭션을 실행시키면 로그 파일에 변경 내용이 기록되고, 비활성 로그가 되면 자동적으로 LOGARCHMETH1 변수가 지정한 <경로명>로 이동됩니다.

```
$ ls –R <경로명>
S0000000.LOG  S0000001.LOG  S0000002.LOG
```

7. 활성 로그 파일의 시작 번호를 확인하면 마지막 아카이브 로그의 번호를 알 수 있습니다.

```
$ db2 get db cfg for <DB명> | grep "처음에 사용되는 로그 파일"
  처음에 사용되는 로그 파일                    = S0000002.LOG
```

TOPIC 1306 복구 기록 파일

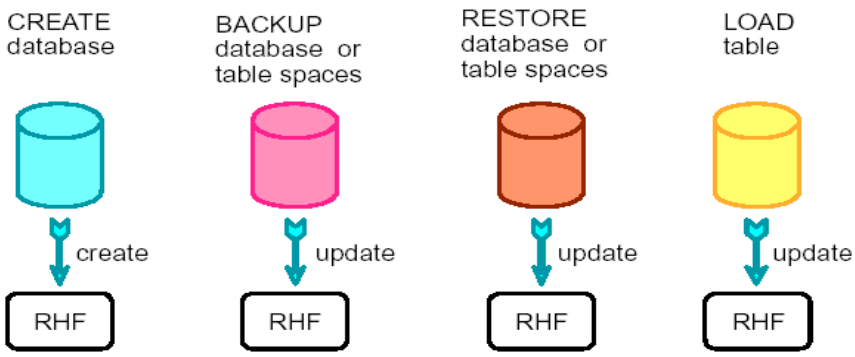
Point

데이터베이스를 생성하면 복구 실행 기록 파일이 생성되어, 데이터베이스에 대한 BACKUP, RESTORE, ROLLFORWARD, LOAD 등의 작업에 대한 정보가 관리됩니다.

Tip

복구 기록 파일이 손상되면, restore db 명령어로 복구할 수 있습니다.

1 create db 명령어로 데이터베이스를 생성하면 복구 기록 파일이 생성됩니다.



- The part of the database that was copied and how
- The time the copy was made
- The location of the copy
- The last time a restore was done

→ `db2 LIST HISTORY ALL FOR MUSICDB`

Figure 1306A ••• 복구 기록 파일의 생성과 갱신

2 복구 기록 파일은 다음과 같은 명령어가 실행될 때 갱신됩니다.

대 상	명 령 어
데이터베이스	BACKUP DATABASE RESTORE DATABASE ROLLFORWARD DATABASE
테이블스페이스	BACKUP DATABASE RESTORE DATABASE ROLLFORWARD DATABASE CREATE TABLESPACE ALTER TABLESPACE QUIESCE TABLESPACE RENAME TABLESPACE DROP TABLESPACE
테이블	LOAD REORG TABLE REORG INDEXES DROP TABLE
로그	ARCHIVE LOG 로그 파일이 아카이브 로그로 저장될 때

TOPIC 1307 LIST HISTORY 명령어

Point

LIST HISTORY 명령어로 복구 실행 기록 파일에서 작업 유형별로 기록을 확인할 수 있습니다. SINCE 옵션을 이용하면 특정 시점 이후의 기록을 확인할 수 있으며, CONTAINING 옵션을 이용하면 지정한 오브젝트와 관련된 기록만 추출할 수 있습니다.

Tip

restore db 명령어를 실행한 기록을 BACKUP 옵션으로 backup db 명령어를 실행한 기록과 함께 확인합니다.

Tip

drop tablespace 문을 실행한 기록을 CREATE TABLESPACE 옵션으로 create tablespace문을 실행한 기록과 함께 확인합니다.

Tip

조작 종류 (Op)는 다음과 같습니다.

```
A - Create table space
B - Backup
C - Load copy
D - Dropped table
F - Roll forward
G - Reorganize table
L - Load
N - Rename table space
O - Drop table space
Q - Quiesce
R - Restore
T - Alter table space
U - Unload
X - Archive log
```

Tip

조작 대상 (Obj)은 다음과 같습니다.

```
D - Database
P - Tablespace
T - Table
I - Index
```

Tip

조작 유형은 다음과 같습니다.

```
Backup types:
  F - Offline
  N - Online
  I - Incremental offline
  O - Incremental online
  D - Delta offline
  E - Delta online

Rollforward types:
  E - End of logs
  P - Point in time

Load types:
  I - Insert
  R - Replace

Alter table space types:
  C - Add containers
  R - Rebalance

Quiesce types:
  S - Quiesce share
  U - Quiesce update
  X - Quiesce exclusive
  Z - Quiesce reset
```

1 list history 명령어를 이용하여 복구 기록 파일의 내용을 조회할 수 있습니다.

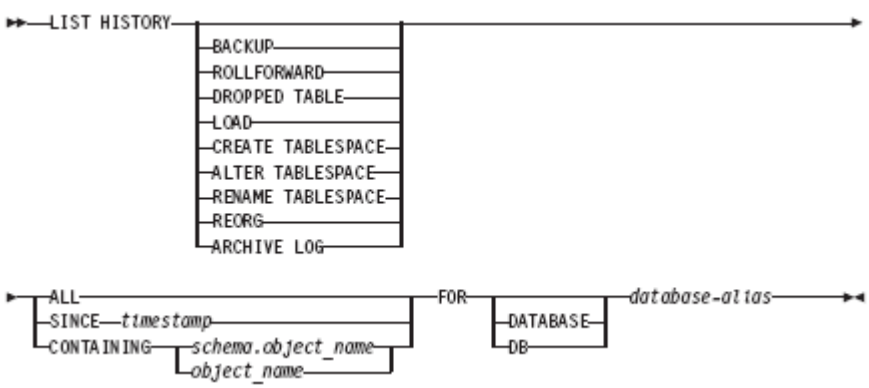


Figure 1307A ··· LIST HISTORY 명령어

2 list history 명령어를 이용하여 복구 기록 파일의 내용을 조회할 수 있습니다.

```
$ db2 list history ALL for <DB명>
$ db2 list history BACKUP all for <DB명>
$ db2 list history ROLLFORWARD all for <DB명>
$ db2 list history backup SINCE <시간소인>  for <DB명>
$ db2 list history rollforward CONTAINING <스키마명>.<테이블명> for <DB명>
$ db2 list history REORG all for sample
```

3 복구 기록 파일이 저장하는 정보는 다음과 같습니다.

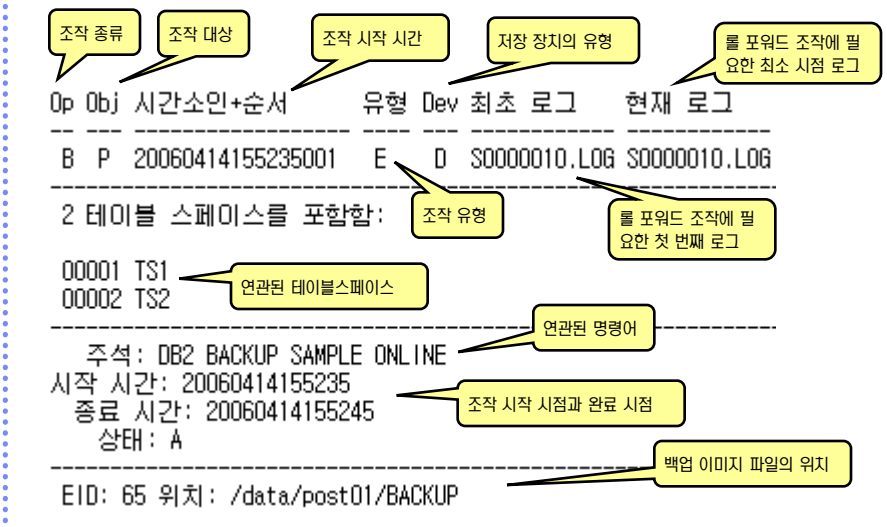


Figure 1307B ··· backup db 명령어의 실행에 관한 기록 정보

TOPIC 1308 PRUNE HISTORY 명령어

Point

PRUNE HISTORY 명령어로 복구 실행 기록 파일에서 특정 <시간 소인> 이전의 기록을 제거할 수 있습니다. 복구 실행 기록 파일의 기본 보관 주기는 366일이며, REC_HIS_RETENTN 데이터베이스 구성 변수로 조절할 수 있습니다.

1 prune history 명령을 이용하여 복구 기록 파일의 내용 중에서 지정한 시점 이전의 기록을 삭제할 수 있습니다.

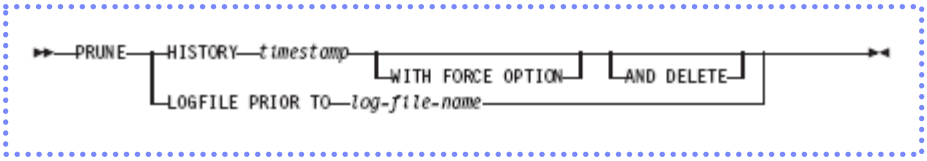


Figure 1308A • • • PRUNE HISTORY 명령어

2 prune history 명령을 이용하여 복구 기록 파일의 내용 중에서 지정한 시점 이전의 기록을 삭제할 수 있습니다. <시간소인>은 <yyyymmddhhmmss> 형식으로 표시합니다. <yyyy> 이상을 명시할 수 있습니다.

```
$ db2 prune history <yyyy>
$ db2 prune history <yyyymmdd>
$ db2 prune history <yyyymmddhhmmss>
```

3 WITH FORCE 옵션을 이용하면 복구 기록 파일의 모든 항목을 완전히 제거합니다. 옵션을 지정하지 않으면, 최근의 FULL 백업 정보와 연관되는 복구 작업의 기록은 유지됩니다.

```
$ db2 prune history <yyyymmddhhmmss> WITH FORCE OPTION
```

4 AND DELETE 옵션을 이용하면 복구 기록 파일에서 항목이 삭제되고, 해당 항목과 관련된 아카이브 로그 파일도 물리적으로 제거됩니다. USEREXIT 프로그램을 사용하여 아카이브 로그 파일을 관리하는 경우에는 지원되지 않습니다.

```
$ db2 prune history <yyyymmddhhmmss> AND DELETE
```

5 prune logfile prior to 명령어로 지정한 번호보다 작은 번호의 로그 파일을 모드 제거합니다. LOGRETAIN 변수가 ON으로 설정되어야 합니다. 로그 파일명은 S0000100.LOG 와 같은 형식으로 표시합니다.

```
$ get db cfg for <DB명> | grep LOGARCHMETH1
첫 번째 로그 아카이브 메소드          (LOGARCHMETH1) = LOGRETAIN
$ db2 prune logfile prior to <로그파일명>
```

6 update dbm cfg 명령어를 이용하여 REC_HIS_RETENTN 데이터베이스 구성 변수를 변경합니다. <기간>은 1일 단위로 지정합니다. 기본값은 366일입니다.

```
$ db2 get db cfg for <DB명> | grep REC_HIS_RETENTN
  복구 실행기록 보유(일)            (REC_HIS_RETENTN) = 366
$ db2 update db cfg for <DB명> using REC_HIS_RETENTN <기간>
```

Tip

update db cfg 명령어를 실행한 후에 SQL1363W 경고를 받았다면, 변경값이 동적으로 반영되지 않았으므로 데이터베이스를 재활성화합니다.

TOPIC 1309 백업의 종류

Point

BACKUP DB 명령어를 이용하여 특정한 데이터베이스의 모든 데이터와 제어 정보를 저장한 이미지 파일을 생성할 수 있습니다. ONLINE/OFFLINE 백업, FULL/TABLESPACE 백업, FULL/INCREMENTAL/DELTA 백업으로 분류됩니다.

Tip

백업 이미지는 한 개 이상의 파일로 생성되어 디스크, 테이프, TSM 등으로 저장됩니다.

Tip

백업 이미지에는 해당 데이터베이스에 대한 모든 정보와 데이터가 저장되므로, 새로운 데이터베이스를 생성하는데 이용되기도 합니다.

1 SYSADM, SYSCTRL, SYSMAINT 권한의 사용자가 원격 또는 지역 데이터베이스에 대해 백업을 실행하면 백업 이미지 파일이 생성됩니다.

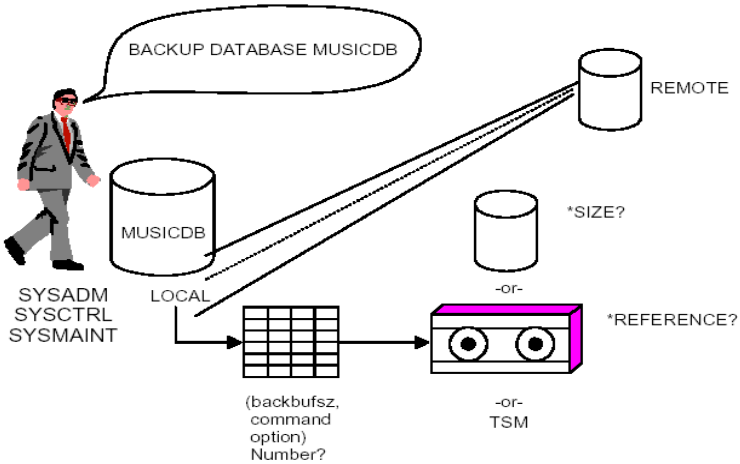


Figure 1309A ••• 지역 또는 원격 데이터베이스의 백업

2 백업의 모드는 2가지로 분류됩니다.

모 드	설 명
OFFLINE	데이터베이스에 접속된 사용자가 없는 상태에서 백업하는 모드입니다. 백업이 진행되는 동안 데이터의 변경은 발생하지 않으므로, 백업 이미지 파일만 보관하면 복구에 사용할 수 있습니다.
ONLINE	사용자가 데이터베이스에 접속하고 있는 상태에서 백업하는 모드입니다. 백업이 진행되는 동안 데이터가 변경 중일 수도 있으므로, 백업 이미지 파일과 백업이 진행되는 동안 변경된 데이터에 대한 로그 파일을 함께 보관해야 복구에 사용할 수 있습니다. 아카이브 로깅에서만 지원됩니다.

3 백업의 수준은 2가지로 분류됩니다.

대 상	설 명
DATABASE	데이터베이스의 모든 테이블스페이스를 백업합니다.
TABLESPACE	지정한 테이블스페이스만 백업합니다. 아카이브 로깅에서만 지원됩니다.

4 백업의 범위는 3가지로 분류됩니다.

범 위	설 명
FULL	데이터베이스의 모든 데이터와 제어 정보를 백업합니다.
INCREMENTAL	최근의 FULL 백업 이후에 변경된 부분만 백업합니다. 아카이브 로깅에서만 지원됩니다.
DELTA	최근의 FULL 또는 INCREMENTAL 백업 이후에 변경된 부분만 백업합니다. 아카이브 로깅에서만 지원됩니다.

TOPIC 13-10 BACKUP DB 명령어

Point

다양한 방식의 데이터베이스 백업은 모두 BACKUP DB 명령어를 이용해서 실행합니다. SYSADM, SYSCTRL, SYSMAINT 권한을 가진 사용자가 실행하며, 백업 이미지 파일을 생성합니다.

1 BACKUP DB 명령어의 형식은 다음과 같습니다.

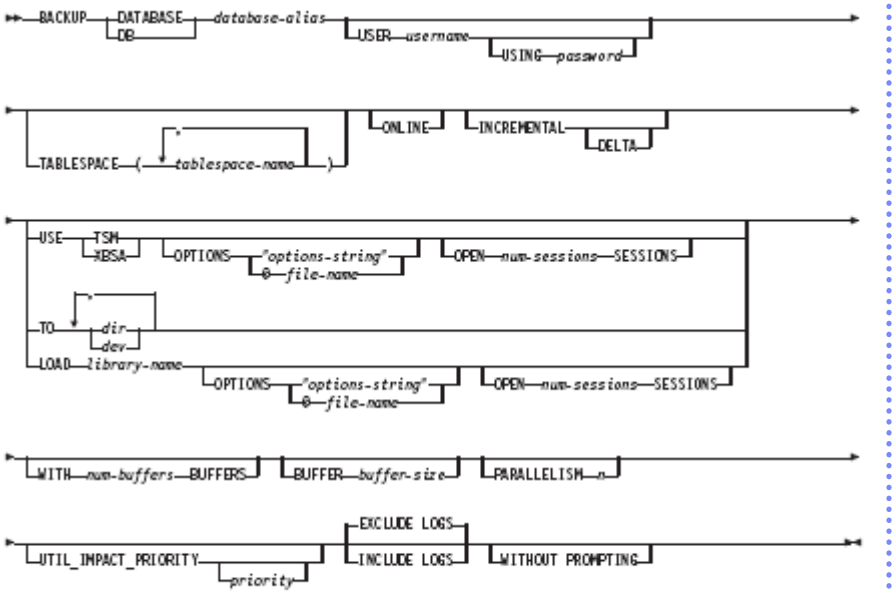


Figure 1310A • • • BACKUP DB 명령어

2 자주 사용되는 옵션은 다음과 같습니다.

옵션	설명
DB명	지역 DB명 또는 catalog db 명령어로 등록된 원격 DB명을 지정합니다.
USER, USING	백업을 실행하는 사용자명과 암호명을 지정합니다. 원격 데이터베이스의 백업에는 반드시 USER와 USING 옵션으로 사용자명과 암호명을 지정해야 합니다.
TABLESPACE	괄호안에 백업을 원하는 테이블스페이스의 이름들만 나열합니다.
ONLINE	온라인 모드의 백업을 실행합니다. 아카이브 로깅에서만 지원됩니다.
INCREMENTAL	INCREMENTAL 백업을 실행합니다. 아카이브 로깅에서만 지원됩니다.
DELTA	DELTA 백업을 실행합니다. INCREMENTAL 옵션과 함께 사용합니다. 아카이브 로깅에서만 지원됩니다.
TO	백업 이미지 파일이 생성되는 디렉토리를 지정할 수 있습니다. 한 개 이상의 경로명을 지정하면, 백업 이미지 파일은 여러 개의 파일로 분할되어 생성됩니다. 옵션을 지정하지 않으면, 현재 디렉토리에 한 개의 파일로 생성됩니다.
WITH ~ BUFFERES	백업 작업을 위해 데이터베이스 공유 메모리에서 할당하는 버퍼의 개수를 지정합니다.
BUFFER	백업 작업을 위해 할당하는 버퍼의 크기를 지정합니다.
INCLUDE LOGS	복구에 필요한 최소한의 로그 파일을 백업 이미지에 함께 저장하는 옵션으로 오프라인 백업에는 적용되지 않습니다.

Tip

오프라인 모드의 데이터베이스 백업을 받으려면, 접속된 모든 응용프로그램을 종료해야 합니다.

Tip

INCREMENTAL 백업, DELTA 백업은 아카이브 로깅 모드에서 TRACKMOD 구성 변수가 YES로 설정되어야 가능합니다.

Tip

원격 데이터베이스의 백업 이미지는 서버 머신에 생성되므로 TO 옵션에는 원격 서버의 경로명을 지정합니다.

Tip

백업이 완료되면, 사용되던 마지막 활성 로그는 truncate 되고, 새로운 로그 파일이 사용됩니다.,

TOPIC 13.11 백업 이미지 파일

Point

백업 이미지 파일은 기본적으로 한 개입니다. UNIX 에서 백업 이미지 파일의 이름에는 DB명, 백업의 유형, 인스턴스명, 데이터베이스 파티션 번호, 백업 완료 시간 소인, 파일 일련 번호 등의 정보가 반영됩니다.

Tip

데이프 장치에 생성된 백업 이미지는 db2ckbkp 유틸리티를 이용하여 헤더 파일에 저장된 백업 정보를 확인할 수 있습니다.

Tip

UNIX 에서는 복구시에 임의의 디렉토리에 이미지 파일을 복사하면 됩니다. Windows 인 경우에는 하위 경로명이 백업 이미지를 생성했을 때와 동일해야 합니다.

Tip

백업 이미지 파일의 이름에 온라인, 오프라인 유형 정보는 반영되지 않습니다. 상세한 list history 명령어로 확인할 수 있습니다.

Tip

백업 이미지 파일의 크기가 OS에서 지정한 한계를 초과하거나 backup db 명령어의 TO 옵션으로 강제로 분할해서 생성한 경우에는 두 개 이상의 파일로 분할됩니다.

1. UNIX에서 백업 이미지 파일명이 백업에 관한 정보를 반영하고 있습니다..

Figure 1311A ··· UNIX 에서 생성된 백업 이미지 파일명

2. Windows 에서는 서브경로명과 백업 이미지 파일명이 함께 백업 정보를 반영합니다.

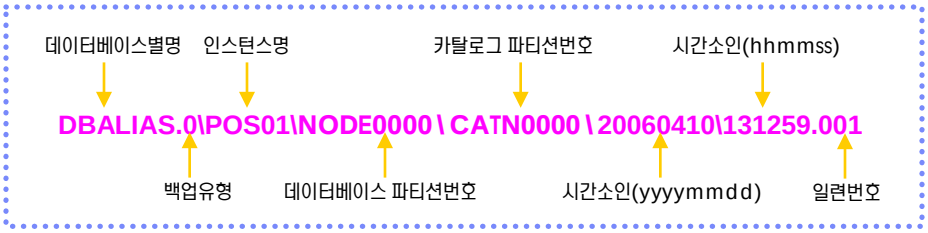


Figure 1311B ··· Windows 에서 생성된 백업 이미지 파일명

3. 백업 이미지 파일명의 각 항목은 다음과 같습니다.

옵션	설명
DB명	지역 DB명 또는 catalog db 명령어로 등록된 원격 DB명을 표시합니다.
백업 유형	0 은 데이터베이스 백업 이미지, 3은 테이블스페이스 백업 이미지, 4 는 LOAD 유틸리티의 COPY YES 옵션으로 생성된 백업 이미지를 나타냅니다.
인스턴스명	데이터베이스가 존재하는 서버의 인스턴스 이름입니다. 원격 데이터베이스인 경우에는 원격 서버 머신에 백업 이미지 파일이 생성되고, 원격 인스턴스명이 표시됩니다.
데이터베이스 파티션 번호	백업된 데이터베이스 파티션의 번호입니다. DPF 환경에서는 파티션별로 백업 이미지가 생성되며, 단일 파티션인 경우에는 항상 NODE0000 이 됩니다.
카탈로그 파티션 번호	카탈로그 테이블이 생성된 데이터베이스 파티션의 번호입니다. 단일 파티션인 경우에는 항상 CATN0000 이 됩니다.
시간 소인	백업 작업이 완료된 시간 소인입니다. UNIX 머신에서는 <yyyymmddhhmmss>의 형태로 표시되고, Windows 에서는 <yyyymmdd>과 <hhmmss>로 분리됩니다.
일련 번호	백업 이미지 파일이 두 개 이상의 파일로 분할된 경우에는 동일한 이름을 가지며, 일련 번호가 다른 여러 개의 백업 이미지 파일이 생성됩니다.

TOPIC 13-12 FULL 백업

Point

BACKUP DB 명령어로 데이터베이스 전체에 대한 백업 이미지 파일을 생성합니다. 순환 로깅에서는 오프라인 백업만 지원되며, 아카이브 로깅 모드에서는 오프라인 백업과 온라인 백업을 모두 지원합니다. 원격 데이터베이스도 백업할 수 있습니다.

Tip

SYSADM,SYSCTRL,SYSMAINT 권한이 있는 사용자가 실행합니다.

Tip

아카이브 로깅 모드를 사용하면 백업을 실행하는 시간동안 연관되었던 로그 파일들도 함께 보관해야 복구를 완료할 수 있습니다. 순환 로깅에서 생성된 백업 이미지는 로그 파일과 연관되지 않습니다.

Tip

backup db 명령어를 실행하는 세션은 내부적으로 데이터베이스에 접속하여 백업 이미지 파일을 생성하고, 작업이 완료되면 데이터베이스에 대한 접속을 종료합니다. 온라인 백업을 받은 세션도 마찬가지로 종료됩니다.

Tip

원격 데이터베이스의 백업 이미지 파일은 원격 데이터베이스 서버 머신에 생성됩니다.

Tip

원격 데이터베이스의 백업 경로명은 원격 데이터베이스 서버 머신의 유효한 경로명으로 지정합니다.

Tip

FULL 백업은 <Obj> 항목에 'D'라고 표시되며, <유형> 항목에 오프라인 백업이면 'F', 온라인 백업이면 'N'이라고 표시됩니다.

1 주기적으로 FULL 백업을 실행합니다. 마지막으로 생성한 FULL 백업의 이미지는 복구시에 사용되므로 안전하게 보관합니다. 온라인 백업이라면 백업 작업 동안 사용된 로그 파일을 함께 보관해야 합니다.

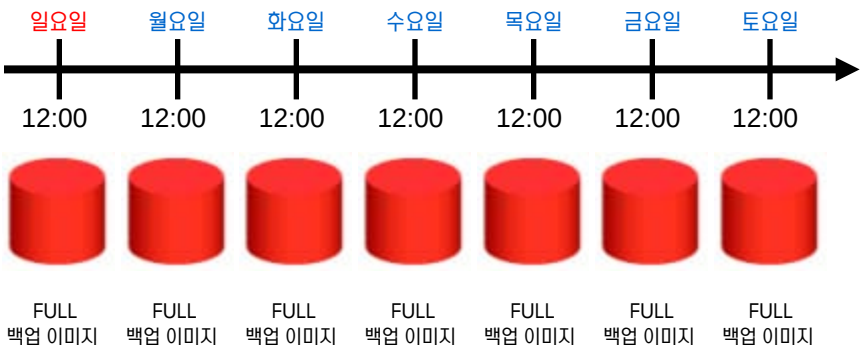


Figure 1312A FULL 백업

2 순환 로깅 모드인 경우에는 오프라인 모드의 FULL 백업만 가능합니다.

```
$ get db cfg for <DB명> | grep LOGARCHMETH1
첫 번째 로그 아카이브 메소드          (LOGARCHMETH1) = OFF
$ db2 backup db <DB명>
```

3 아카이브 로깅 모드인 경우에만 오프라인 백업과 온라인 백업이 모두 가능합니다.

```
$ get db cfg for <DB명> | grep LOGARCHMETH1
첫 번째 로그 아카이브 메소드          (LOGARCHMETH1) = LOGRETAIN
$ db2 backup db <DB명>
$ db2 backup db <DB명> ONLINE
```

4 원격 데이터베이스의 백업을 실행할 때는 반드시 <사용자명>과 <암호명>를 입력해야 합니다.

```
$ db2 backup db <DB명> user <사용자명> using <암호명>
```

5 TO 옵션을 이용하여 백업 이미지를 독립적인 디렉토리에 따로 생성하는 것이 좋습니다.

```
$ db2 backup db <DB명> TO <경로명>
$ db2 backup db <DB명> TO <경로명1>, <경로명2>
```

6 list history 명령어로 해당 INCREMENTAL 백업과 연관된 로그 파일의 번호를 확인합니다.

```
$ db2 list history backup for <DB명>
```

TOPIC 13.13 INCREMENTAL 백업

Point

BACKUP DB 명령어의 INCREMENTAL 옵션을 이용하여 최근의 FULL 백업 이후에 변경된 부분만 백업 이미지 파일에 저장합니다. 아카이브 로깅 모드이고, TRACKMOD 구성 변수가 ON으로 설정되어 있어야 합니다.

Tip

SYSADM,SYSCTRL,SYSMAINT 권한이 있는 사용자가 실행합니다.

Tip

증분식(cumulative) 백업이라고도 합니다.

1. 주기적으로 FULL 백업을 실행하고, 다음 FULL 백업이 되기 전까지는 INCREMENTAL 백업을 여러 번 실행합니다. 기준이 되는 FULL 백업 이미지와 마지막으로 생성한 INCREMENTAL 백업의 이미지, 백업 작업 동안 사용된 로그 파일은 복구를 대비하여 안전하게 보관해야 합니다.

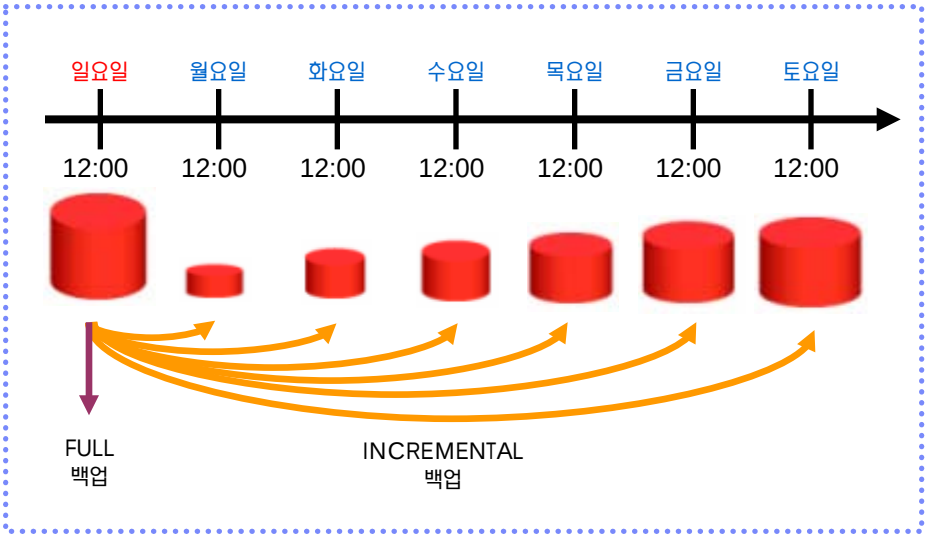


Figure 1313A FULL, INCREMENTAL 백업

Tip

아카이브 로깅 모드를 사용하므로 백업을 실행하는 시간동안 연관되었던 로그 파일들도 함께 보관해야 복구를 완료할 수 있습니다.

2. 아카이브 로깅 모드에서만 지원되므로 LOGARCHMETH1 데이터베이스 구성 변수의 값이 OFF 가 아닌 것을 확인합니다.

```
$ get db cfg for <DB명> | grep LOGARCHMETH1
첫 번째 로그 아카이브 메소드          (LOGARCHMETH1) = LOGRETAIN
```

3. TRACKMOD 데이터베이스 구성 변수의 값이 ON 인 것을 확인합니다.

```
$ db2 get db cfg for <DB명> | grep TRACKMOD
 트랙 수정 페이지                    (TRACKMOD) = ON
```

4. backup database 명령에서 INCREMENTAL 옵션을 이용하여 최근의 FULL 백업 이후에 변경된 부분에 대해서만 백업 이미지를 생성합니다.

```
$ db2 backup db <DB명> INCREMENTAL
$ db2 backup db <DB명> ONLINE incremental
$ db2 backup db <DB명> online incremental  TO <디렉토리명>
$ db2 backup db <DB명> online incremental  to <디렉토리명>  INCLUDE
    LOGS
```

Tip

INCREMENTAL 백업은 <유형> 항목에 오프라인 백업이면 'I', 온라인 백업이면 'O' 라고 표시됩니다.

5. list history 명령어로 해당 INCREMENTAL 백업과 연관된 로그 파일의 번호를 확인합니다.

```
$ db2 list history backup for <DB명>
```

TOPIC 13 14 DELTA 백업

Point

BACKUP DB 명령어의 INCREMENTAL DELTA 옵션을 이용하여 최근의 FULL 또는 INCREMENTAL 백업 이후에 변경된 부분만 백업 이미지 파일에 저장합니다. 아카이브 로깅 모드이고, TRACKMOD 구성 변수가 ON으로 설정되어 있어야 합니다.

Tip

SYSADM,SYSCTRL,SYSMAINT 권한이 있는 사용자가 실행합니다.

1 주기적으로 FULL 백업을 실행하고, 다음 FULL 백업이 되기 전까지는 INCREMENTAL 백업 또는 DELTA 백업을 여러 번 실행합니다. 기준이 되는 FULL 백업 이미지와 마지막으로 생성한 INCREMENTAL 백업 이미지, 최근까지의 DELTA 백업 이미지, 백업 작업 동안 사용된 로그 파일은 복구시에 사용되므로 안전하게 보관해야 합니다.

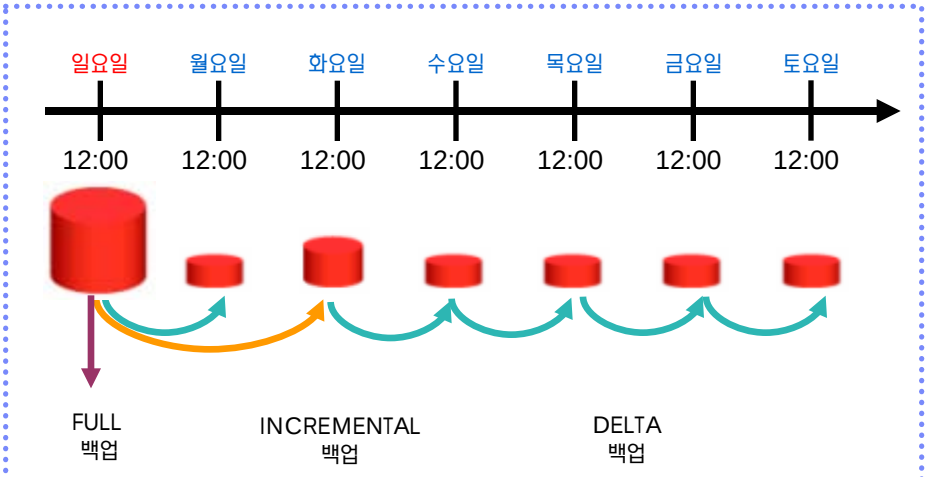


Figure 1314A • • • FULL, INCREMENTAL, DELTA 백업

Tip

아카이브 로깅 모드를 사용하므로 백업을 실행하는 시간동안 연관되었던 로그 파일들도 함께 보관해야 복구를 완료할 수 있습니다.

2 아카이브 로깅 모드에서만 지원되므로 LOGARCHMETH1 데이터베이스 구성 변수의 값이 OFF 가 아닌 것을 확인합니다.

```
$ get db cfg for <DB명> | grep LOGARCHMETH1
첫 번째 로그 아카이브 메소드          (LOGARCHMETH1) = LOGRETAIN
```

3 TRACKMOD 데이터베이스 구성 변수의 값이 ON 인 것을 확인합니다.

```
$ db2 get db cfg for <DB명> | grep TRACKMOD
 트랙 수정 페이지                    (TRACKMOD) = ON
```

4 backup database 명령에서 INCREMENTAL DELTA 옵션을 이용하여 최근의 FULL 또는 INCREMENTAL 백업 이후에 변경된 부분에 대해서만 백업 이미지를 생성합니다.

```
$ db2 backup db <DB명> INCREMENTAL DELTA
$ db2 backup db <DB명> ONLINE incremental delta
$ db2 backup db <DB명> online incremental  delta TO <디렉토리명>
$ db2 backup db <DB명> online incremental  delta to <디렉토리명>
    INCLUDE LOGS
```

Tip

DELTA 백업은 <유형> 항목에 오프라인 백업이면 'D', 온라인 백업이면 'E' 라고 표시됩니다.

5 list history 명령어로 해당 INCREMENTAL 백업과 연관된 로그 파일의 번호를 확인합니다.

```
$ db2 list history backup for <DB명>
```

TOPIC 13.15 테이블스페이스 백업

Point

데이터베이스 전체를 백업하는 것이 아니라, 중요한 테이블스페이스만 백업합니다. BACKUP DB 명령어에서 TABLESPACE 옵션을 이용합니다. 한 개 이상의 테이블스페이스를 함께 백업할 수 있으며, 아카이브 로깅 모드에서 지원됩니다.

Tip

SYSADM,SYSCTRL,SYSMAINT 권한이 있는 사용자가 실행합니다.

1. 카탈로그 테이블스페이스 또는 중요한 테이블스페이스만 별도로 백업합니다. 테이블스페이스에 대한 백업 이미지 파일과 백업 작업 동안 사용된 로그 파일을 안전하게 보관해야 합니다.

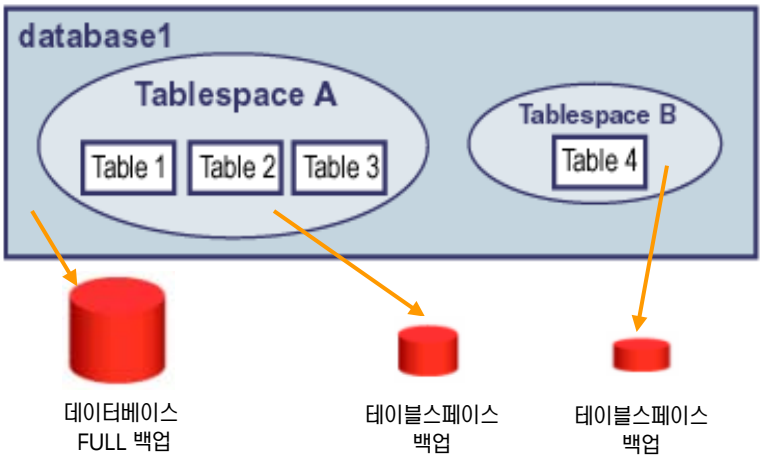


Figure 1315A 테이블스페이스 백업

2. 아카이브 로깅 모드에서만 지원되므로 LOGARCHMETH1 데이터베이스 구성 변수의 값이 OFF 가 아닌 것을 확인합니다.

```
$ get db cfg for <DB명> | grep LOGARCHMETH1
첫 번째 로그 아카이브 메소드            (LOGARCHMETH1) = LOGRETAIN
```

3. backup database 명령에서 TABLESPACE 옵션을 이용하여 한 개 이상의 테이블스페이스에 대한 백업 이미지를 생성합니다. 연관된 테이블스페이스는 함께 백업합니다.

```
$ db2 "backup db <DB명> TABLESPACE (<TS명>)"
$ db2 "backup db <DB명> TABLESPACE (<TS명 1>, <TS명 2>)"
```

4. ONLINE, TO, INCREMENTAL, DELTA 등의 옵션과 함께 사용될 수 있습니다.

```
$ db2 "backup db <DB명> tablespace (<TS명>) ONLINE "
$ db2 "backup db <DB명> tablespace (<TS명>) online TO <디렉토리명>"
$ db2 "backup db <DB명> tablespace (<TS명>) online INCREMENTAL"
$ db2 "backup db <DB명> tablespace (<TS명>) online INCREMENTAL
    DELTA"
```

Tip

TABLESPACE 백업은 <Obj> 항목에 'P'라고 표시됩니다.

5. list history 명령어로 해당 INCREMENTAL 백업과 연관된 로그 파일의 번호를 확인합니다.

```
$ db2 list history backup for <DB명>
```

TOPIC 13-16 복구의 종류

Point

RESTORE DB, ROLLFORWARD DB 명령어에서 백업 이미지와 아카이브 로그로 원하는 시점으로 복구합니다. CRASH/VERSION/ROLLFORWARD, DATABASE/TABLESPACE, FULL/INCREMENTAL/DELTA 복구로 분류됩니다.

Tip

백업 이미지는 한 개 이상의 파일로 생성되어 디스크, 테이프, TSM 등으로 저장됩니다.

Tip

롤포워드 복구, 테이블스페이스 복구, INCREMENTAL 복구, DELTA 복구는 아카이브 로깅에서만 지원되며, 백업 이미지와 연관된 최소한의 아카이브 로그가 반드시 존재해야 합니다.

1 데이터베이스를 사용할 수 없는 상태가 되면 SYSADM, SYSCTRL, SYSMAINT 권한의 사용자가 원격 또는 지역 데이터베이스의 백업 이미지 파일, 아카이브 로그 파일, 활성 로그 파일 등을 이용하여 데이터베이스를 복구합니다.

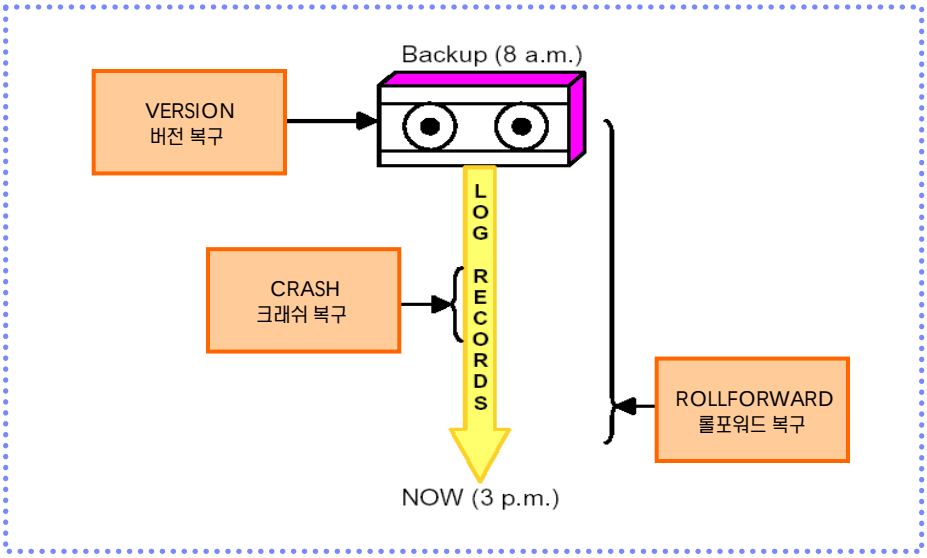


Figure 1316A 다양한 복구 방법

2 복구의 유형은 세 가지로 분류됩니다.

방 법	설 명
크래쉬 복구	정전 등의 비상시에 활성 로그 파일을 이용하여 자동 실행됩니다.
버전 복구	과거 시점에 생성했던 오프라인 모드의 데이터베이스 백업 이미지를 이용하여 과거의 시점과 동일한 상태로 데이터베이스를 복구합니다.
롤포워드 복구	버전 복구를 완료한 후에 아카이브 로그 파일을 이용하여 원하는 시점까지의 로그 파일을 적용하는 복구입니다.

3 복구의 수준은 2가지로 분류됩니다.

대 상	설 명
DATABASE	데이터베이스의 모든 테이블스페이스를 복원합니다.
TABLESPACE	지정한 테이블스페이스만 복원합니다.

4 복구의 범위는 3가지로 분류됩니다.

범 위	설 명
FULL	FULL 백업 이미지를 이용하여 복구합니다.
INCREMENTAL	최근의 INCREMENTAL 백업 이미지와 기준이 되는 FULL 백업 이미지를 이용하여 복구합니다.
DELTA	최근의 DELTA 백업 이미지들과 기준이 되는 INCREMENTAL 백업 이미지, 기준이 되는 FULL 백업 이미지를 이용하여 복구합니다.

TOPIC 13.17 RESTART DB 명령어

Point

시스템 정전 등의 비상시에 실행되는 크래쉬 복구를 실행하는 명령어입니다. 인스턴스가 비정상적으로 종료되는 경우에도 크래쉬 복구가 필요합니다. 크래쉬 복구의 자동 실행 여부는 AUTORESTART 데이터베이스 구성 변수로 조절합니다.

Tip

restart db 명령어를 실행하는 세션은 내부적으로 데이터베이스에 접속합니다. 작업을 완료한 후에 데이터베이스에 대한 접속은 유지됩니다.

1 RESTART DB 명령어의 형식은 다음과 같습니다.

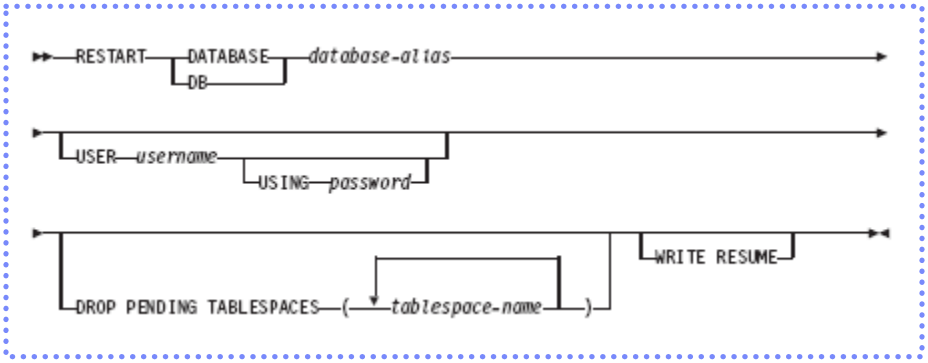


Figure 1317A RESTART DB 명령어

2 사용되는 옵션은 다음과 같습니다.

옵션	설명
DB명	지역 DB명 또는 catalog db 명령어로 등록된 원격 DB명을 지정합니다.
USER, USING	크래쉬 복구를 실행하는 사용자명과 암호명을 지정합니다. 원격 데이터베이스의 복구에는 반드시 USER와 USING 옵션으로 사용자명과 암호명을 지정해야 합니다.
DROP PENDING TABLESPACES	테이블스페이스의 컨테이너가 손상되면 해당 테이블스페이스는 사용할 수 없게 됩니다. 이러한 테이블스페이스가 발견되면, 크래쉬 복구를 완료할 수 없습니다. 괄호안에 DROP을 원하는 테이블스페이스의 이름들을 나열하면 해당 테이블스페이스는 'DROP 보류' 상태가 됩니다. 크래쉬 복구가 완료된 후에 데이터베이스에 접속하여 테이블스페이스를 재생성해야 합니다.

TOPIC 13-18 RESTORE DB 명령어

Point

BACKUP DB 명령어로 생성된 백업 이미지 파일을 이용하여 데이터베이스를 과거의 시점과 동일한 버전으로 복구합니다. SYSADM, SYSCTRL, SYSMAINT 권한을 가진 사용자가 실행하며, 백업 이미지 파일이 필요합니다.

1 RESTORE DB 명령어의 형식은 다음과 같습니다.

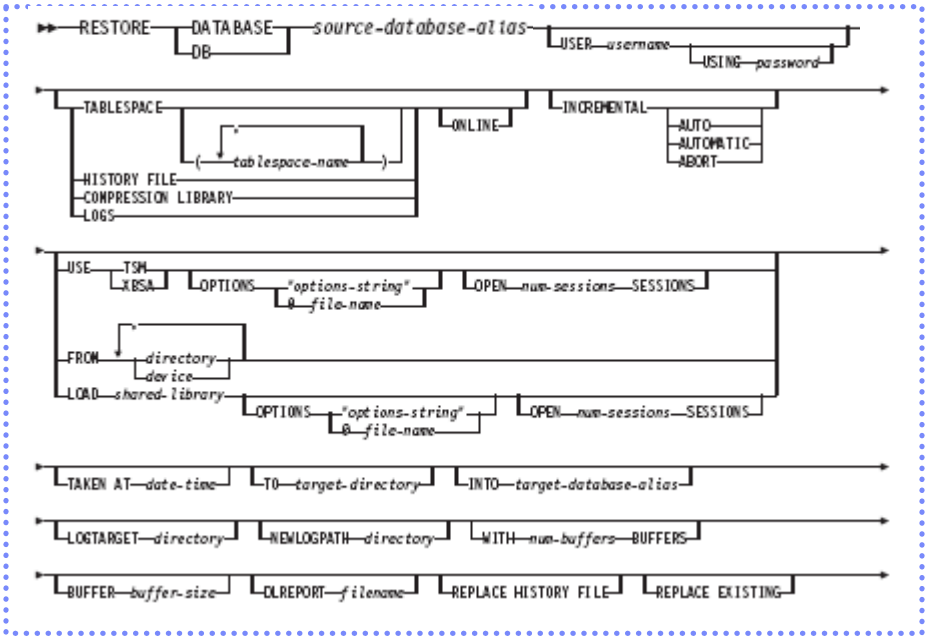


Figure 1318A ··· RESTORE DB 명령어

2 자주 사용되는 옵션은 다음과 같습니다.

옵션	설명
DB명	지역 DB명 또는 등록된 원격 DB의 별명을 지정합니다.
USER, USING	복구를 실행하는 사용자명과 암호명을 지정합니다. 원격 데이터베이스에는 반드시 사용자명과 암호명을 지정해야 합니다.
TABLESPACE	괄호안에 복구를 원하는 테이블스페이스의 이름들만 나열합니다.
HISTORY FILE	복구 실행 기록 파일만 복구합니다.
ONLINE	온라인 모드로 복구합니다. TABLESPACE 복구에 지원됩니다.
INCREMENTAL	INCREMENTAL 또는 DELTA 백업 이미지의 복구를 실행합니다.
FROM	백업 이미지 파일이 존재하는 디렉토리를 지정합니다. 백업 이미지 파일이 여러 개의 파일로 분할되어 생성되었으면, 한 개 이상의 경로명을 지정해야 합니다. 옵션을 지정하지 않으면, 현재 디렉토리에 한 개의 백업 이미지 파일을 이용합니다.
TAKEN AT	지정한 경로명에 동일한 DB에 대한 백업 이미지가 여러 개 있는 경우에는 백업 이미지의 생성 시간 소인을 명시하여 구별합니다.
TO	새로운 데이터베이스가 생성될 경로명을 지정합니다.
INTO	새로 생성될 데이터베이스명을 지정합니다.
WITH ~ BUFFERES	복구 작업을 위해 데이터베이스 공유 메모리에서 할당하는 버퍼의 개수를 지정합니다.
BUFFER	복구 작업을 위해 할당하는 버퍼의 크기를 지정합니다.

Tip

지정한 디렉토리에 동일한 데이터베이스에 대한 백업이 한 개만 있으면 TAKEN AT 옵션이 필요없습니다.

Tip

INCREMENTAL 복구, DELTA 복구는 아카이브 로깅 모드에서 TRACKMOD 구성 변수가 YES로 설정되어야 가능합니다.

Tip

FULL 데이터베이스 복구시에는 데이터베이스가 비활성화되어야 하므로 접속된 모든 응용프로그램을 종료해야 합니다.

Tip

원격 데이터베이스의 백업 이미지는 서버 머신에 존재하므로 FROM 옵션에는 원격 서버의 경로명을 지정합니다.

TOPIC 13-19 ROLLFORWARD DB 명령어

Point

RESTORE DB 명령어로 데이터베이스가 과거의 시점으로 버전 복구가 완료된 후에 아카이브 로그를 차례로 읽어들여 로그에 기록된 변경 내용을 데이터베이스에 반영합니다. 버전 복구에 사용된 백업 이미지가 생성된 시점 이후의 아카이브 로그가 필요합니다.

1 ROLLFORWARD DB 명령어의 형식은 다음과 같습니다.

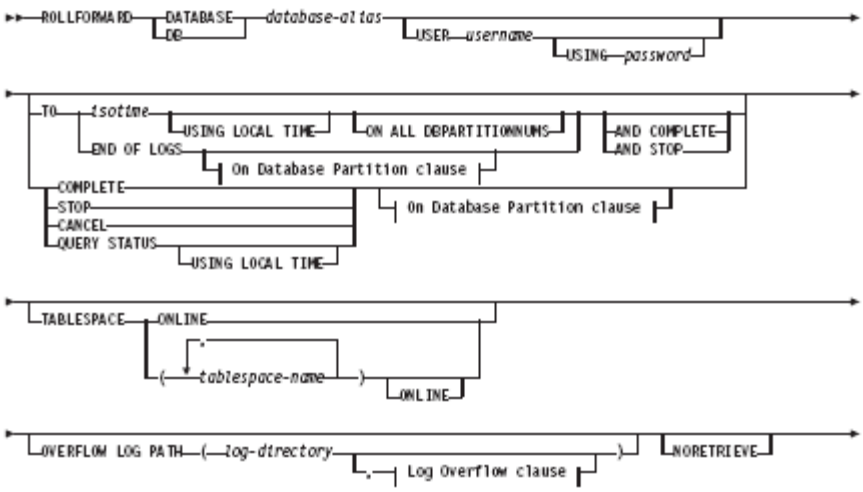


Figure 1319A ··· ROLLFORWARD DB 명령어

2 자주 사용되는 옵션은 다음과 같습니다.

옵션	설명
DB명	롤포워드의 대상이 되는 데이터베이스의 별명을 지정합니다.
USER, USING	롤포워드를 실행하는 사용자명과 암호명을 지정합니다. 원격 데이터베이스는 반드시 사용자명과 암호명을 지정해야 합니다.
TO isotime	<표준시간> 으로 표시한 시점까지의 로그 파일을 적용합니다. OS의 TIMEZONE 값을 이용한 환산이 필요합니다. <표준시간>은 <yyyy-mm-dd-hh.mm.ss.nnnnnn>의 형태로 표현합니다.
USING LOCAL TIME	TO isotime 옵션과 함께 사용합니다. <표준시간> 대신에 현 시스템의 시간대에서 사용하는 <지역 시간>을 지정합니다. <지역 시간>도 <yyyy-mm-dd-hh.mm.ss.nnnnnn>의 형태로 표현합니다.
TO END OF LOGS	로그 디렉토리에 존재하는 온라인 아카이브 로그 파일의 최대 번호까지 롤포워드를 실행합니다.
STOP	롤포워드를 종료하고, '롤포워드 보류' 상태를 해제합니다.
CANCEL	롤포워드를 중단하고, 데이터베이스를 다시 '리스토어 보류' 상태가 되게 합니다. RESTORE DB 명령어로 다시 복구 작업을 실행해야 합니다.
QUERY STATUS	롤포워드 작업의 진행 정보를 알려줍니다.
TABLESPACE	테이블스페이스 수준의 롤포워드 작업시에 사용합니다.
ONLINE	테이블스페이스 수준의 롤포워드를 온라인 모드로 실행합니다. 다른 테이블스페이스는 액세스할 수 있습니다.

Tip

로그 파일의 내용은 표준 시간으로 기록됩니다. 한국에서 사용하는 데이터베이스 서버는 대부분 TIMEZONE의 값이 +9 이므로, 원하는 시점에서 -9를 하면 표준 시간이 산출됩니다. USING LOCAL TIME 옵션을 사용할 것을 권장합니다.

Tip

END OF LOGS 는 최근의 로그까지 적용한다는 의미가 아니라, 로그 디렉토리에 존재하는 모든 아카이브 로그를 적용한다는 의미이므로 구별해야 합니다.

Tip

ROLLFORWARD 명령은 여러 번 반복적으로 실행할 수 있습니다. 적용할 아카이브 로그 파일이 너무 많아서 로그 디렉토리로 한꺼번에 복사할 수 없는 경우에는 일정 개수씩 분할하여 여러 번 반복해서 END OF LOGS 옵션으로 적용합니다. 마지막 적용이 끝나면 STOP 옵션으로 롤포워드를 종료합니다.

TOPIC 13-20 크래쉬 복구

Point

데이터베이스가 비정상적으로 종료되면, 데이터의 일관성을 보장하기 위해 restart db 명령어로 활성 로그 파일을 읽어서 실행 중이던 트랜잭션의 COMMIT과 ROLLBACK을 완료합니다. AUTORESTART 변수가 ON 이므로 자동적으로 실행됩니다.

Tip

특별한 권한이 필요하지 않습니다.

Tip

크래쉬 복구는 로깅 모드와 상관이 없습니다. 크래쉬 복구에 필요한 것은 손상 직전의 활성 로그입니다.

1. 응용프로그램이 연결되어 있을 때 데이터베이스가 비정상 종료하는 경우에는 크래쉬 복구가 필요합니다. 데이터베이스의 비정상 종료는 전원 공급 중단이나 시스템 소프트웨어 장애로 인해 발생할 수 있습니다. 이러한 종료는 실패시 데이터베이스 버퍼 풀에 있으나 디스크에 기록되지 않은 커미트 트랜잭션에 적용됩니다. 또한 디스크에 기록된 미확약 트랜잭션을 롤백시킵니다.

2. get dbm cfg 명령어로 데이터베이스 구성 변수인 AUTORESTART의 기본값이 ON 인 것을 확인합니다. ON 은 필요시에 크래쉬 복구가 자동적으로 실행되는 것을 의미합니다.

```
$ login <인스턴스 사용자>
$ db2 get db cfg for <DB명> | grep AUTORESTART
자동 재시작 사용                    (AUTORESTART) = ON
```

3. AUTORESTART 값을 OFF로 설정하면, 필요시에 크래쉬 복구가 자동적으로 실행되지 않습니다. activate db 명령어로 데이터베이스를 활성화시키거나, connect 명령어로 최초의 데이터베이스 접속을 요청하면, SQL1015N 오류가 반환됩니다. restart db 명령어를 사용하여 크래쉬 복구를 실행합니다. 크래쉬 복구가 완료되면 데이터베이스를 사용할 수 있습니다.

```
$ db2 RESTART DATABASE <DB명>
$ db2 ACTIVATE DATABASE <DB명>
$ db2 CONNECT TO <DB명>
```

4. 크래쉬 복구에 대한 정보는 오류 진단 파일인 db2diag.log 파일에서 확인할 수 있습니다.

```
$ vi  $HOME/sqllib/db2diag.log
```

```
2006-03-08-16.32.39.217725+540 I990834A425        LEVEL: Warning
PID     : 4190438              TID  : 1           PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1               NODE : 000         DB   : SAMPLE
APPHDL  : 0-7                  APPID: 11210612.C3E6.06ADB8073211
FUNCTION: DB2 UDB, recovery manager, sqlpresr, probe:410
MESSAGE : Crash recovery started. LowtranLSN 00000297C9853A02 MinbuffLSN
          00000297C8275DB1

2006-03-08-16.32.39.271097+540 I991260A409        LEVEL: Warning
PID     : 4190438              TID  : 1           PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1               NODE : 000         DB   : SAMPLE
APPHDL  : 0-7                  APPID: 11210612.C3E6.06ADB8073211
FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:2000
MESSAGE : Using parallel recovery with 13 agents 52 QSets 624 queues and 256 chunks

2006-03-08-16.32.44.217305+540 I991670A306        LEVEL: Warning
PID     : 1863868              TID  : 1           PROC : db2lfr (SAMPLE) 0
INSTANCE: db2inst1               NODE : 000
FUNCTION: DB2 UDB, recovery manager, sqlplfrVerifyLogPages, probe:1750
MESSAGE : Saving partial page, pageLsn is 0297C9854854

2006-03-08-16.32.46.835129+540 I991977A435        LEVEL: Warning
PID     : 4190438              TID  : 1           PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1               NODE : 000         DB   : SAMPLE
APPHDL  : 0-7                  APPID: 11210612.C3E6.06ADB8073211
FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:4000
MESSAGE : DIA2051W Forward phase of crash recovery has completed.  Next LSN is
          "00000297C9854855".

2006-03-08-16.32.46.857746+540 I992413A390        LEVEL: Warning
PID     : 4190438              TID  : 1           PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1               NODE : 000         DB   : SAMPLE
APPHDL  : 0-7                  APPID: 11210612.C3E6.06ADB8073211
FUNCTION: DB2 UDB, recovery manager, sqlpresr, probe:3170
MESSAGE : Crash recovery completed. Next LSN is 00000297C9854855
```

Figure 1320A • • • 크래쉬 복구에 관한 메시지

TOPIC 13.21 버전 복구

Point

RESTORE DB 명령어로 원격 또는 지역 데이터베이스를 이전 버전으로 복구하는 유틸리티입니다. 오프라인 모드의 FULL 백업 이미지 파일이 필요합니다. RESTBUFSZ 옵션으로 복구 작업을 위한 버퍼를 할당할 수 있습니다.

Tip

SYSADM,SYSCTRL,SYSMAINT 권한이 있는 사용자가 실행합니다.

Tip

백업 이미지는 한 개 이상의 파일로서 디스크, 테이프, TSM 등에 미리 저장되어 있어야 됩니다.

1 데이터베이스가 손상된 경우에는 최근에 생성했던 FULL 백업 이미지 파일만 있으면, restore db 명령으로 데이터베이스를 이전 버전으로 복구할 수 있습니다. 복구에 사용된 백업 이미지가 생성되었던 시점 이후부터 현재까지의 데이터베이스에 대한 모든 변경 내용은 유실됩니다.

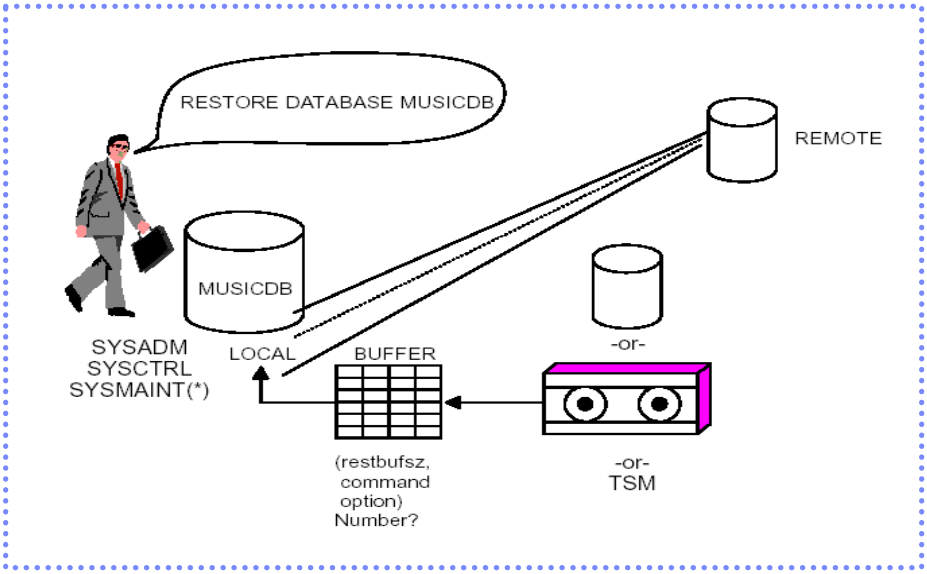


Figure 1321A RESTORE DB 명령어를 이용한 버전 복구

2 restore database 명령을 이용하여 데이터베이스의 백업 이미지로부터 기존 데이터베이스로 복구합니다. 접속된 모든 응용프로그램은 종료되어야 합니다.

```
$ db2 force applications all
$ db2 restore db <DB명>
```

Tip

신규 데이터베이스로 복구하는 경우에는 SYSADM 또는 SYSCTRL 권한이 필요합니다.

3 INTO 옵션을 사용하면 신규 데이터베이스로 복구할 수 있습니다.

```
$ db2 restore db <DB명> INTO <신규 DB명>
```

4 백업 이미지가 다른 디렉토리에 있는 경우에는 restore 명령어의 FROM 옵션을 이용합니다.

```
$ db2 restore db <DB명> FROM <디렉토리명>
```

Tip

<백업시간소인> 은 백업 이미지 파일명의 <yyyymmddhhmmss> 형태로 지정합니다.

5 백업 이미지가 여러 개 있으면 restore 명령어의 TAKEN AT 옵션을 이용합니다.

```
$ db2 restore db <DB명> taken at <백업시간소인>
```

Tip

restore db 명령어는 <Op> 항목에 'R' 이라고 표시됩니다.

6 list history 명령어로 해당 버전 복구와 연관된 기록을 확인합니다.

```
$ db2 list history backup for <DB명>
```

TOPIC 13-22 경로 재지정 복구

Point

백업 이미지 파일에 저장된 테이블스페이스의 컨테이너 구성을 사용자가 변경하여 복구하는 방식입니다. 테이블 스페이스 컨테이너가 손상된 경우의 복구에 유용합니다. RESTORE DB 명령어의 REDIRECT 옵션을 이용합니다.

Tip

SYSADM,SYSCTRL,SYSMAINT 권한이 있는 사용자가 실행합니다.

Tip

restore db 명령어와 set tablespace containers 명령어는 동일한 세션에서 실행합니다.

Tip

원본 데이터베이스의 테이블스페이스의 ID와 컨테이너에 대한 정보를 미리 알고 있어야 합니다. 원본 데이터베이스에서 list tablespace containers 명령어 또는 db2look 명령어를 이용합니다.

1. 백업 이미지 파일에는 테이블스페이스의 컨테이너에 대한 정보가 함께 저장되어 있으므로 복구가 시작되면, 백업 이미지 파일에 설정된 정보에 의해 테이블스페이스가 생성됩니다. 지정한 테이블스페이스 컨테이너를 액세스할 수 없는 상태가 되면 복구 작업은 진행될 수 없습니다. RESTORE DB 명령어의 REDIRECT 옵션과 SET TABLESPACE CONTAINERS 문을 이용하여 액세스 불가능한 컨테이너에 대한 경로를 재지정하면 새로운 컨테이너로 복구됩니다.

2. 백업 이미지는 기본적으로 기존의 데이터베이스를 복구할 때 사용됩니다. 백업 이미지에는 데이터베이스에 대한 모든 정보가 저장되어 있으므로 새로운 데이터베이스로 복구하는 것도 가능합니다. 동일한 데이터베이스 서버에서 새로운 데이터베이스로 복구하게 되면, 테이블스페이스의 컨테이너가 중복되므로 경로 재지정 복구 방법이 사용됩니다.

3. RESTORE DB 명령어에서 REDIRECT 옵션을 사용하면 경로 재지정 복구를 지정됩니다.

```
$ RESTORE DB <백업이미지의 DB명> INTO <새로운 DB명> REDIRECT
```

4. SET TABLESPACE CONTAINERS 명령어에서 컨테이너에 대한 경로를 재지정합니다. FOR 옵션에서 사용되는 번호는 테이블스페이스 ID입니다. 기존의 테이블스페이스의 컨테이너를 제거 또는 변경할 수 있으며, 새로운 컨테이너를 추가할 수도 있습니다.

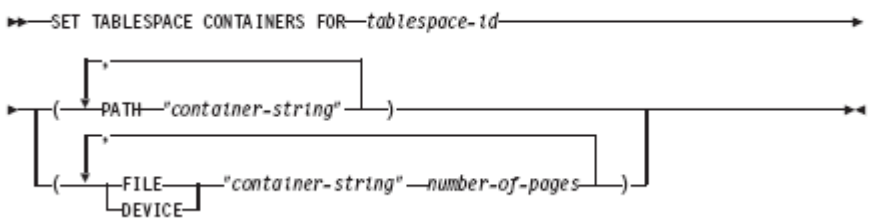


Figure 1322A ••• SET TABLESPACE CONTAINERS 명령어

```
$ SET TABLESPACE CONTAINERS FOR 0 USING (FILE  "<파일명>"  <크기>)
$ SET TABLESPACE CONTAINERS FOR 1 USING (PATH "<경로명>")
```

5. RESTORE DB 명령어에서 ABORT 옵션을 사용하면 경로 재지정 복구가 취소됩니다.

```
$ RESTORE DB <백업이미지의 DB명> INTO <새로운 DB명> ABORT
```

6. RESTORE DB 명령어에서 CONTINUE 옵션을 사용하면 경로 재지정 복구가 시작됩니다.

```
$ RESTORE DB <백업이미지의 DB명> INTO <새로운 DB명> CONTINUE
```

7. ROLLFORWARD DB 명령어로 아카이브 로그를 적용합니다.

```
$ ROLLFORWARD DB <새로운 DB명> TO END OF LOGS AND STOP
```

TOPIC 13.23 롤포워드 복구

Point

RESTORE DB 명령어로 버전 복구를 완료하고, ROLLFORWARD DB 명령어로 로그 디렉토리에 제공된 아카이브 로그 파일들을 차례로 이용하여 과거의 UOW에 의한 변경 내역을 복구합니다. 아카이브 로깅 방식에서만 지원됩니다.

Tip

SYSADM,SYSCTRL,SYSMAINT 권한이 있는 사용자가 실행합니다.

Tip

롤포워드 복구는 restore db 명령어를 이용한 데이터베이스 버전 복구와 rollforward db 명령어를 이용한 로그 재적용 과정을 합친 복구 방법입니다. rollforward db 명령어는 아카이브 로그 파일을 데이터베이스에 재적용하는 조작이므로 롤포워드 복구와 구별해야 합니다.

1 데이터베이스가 손상된 경우에는 최근에 생성했던 FULL 백업 이미지 파일과 아카이브 로그가 있으면, restore db 명령어와 rollforward db 명령어로 데이터베이스를 원하는 시점으로 복구할 수 있습니다. 복구에 사용된 마지막 아카이브 로그에 저장된 트랜잭션 이후부터 현재까지의 데이터베이스에 대한 변경 내용은 유실됩니다.

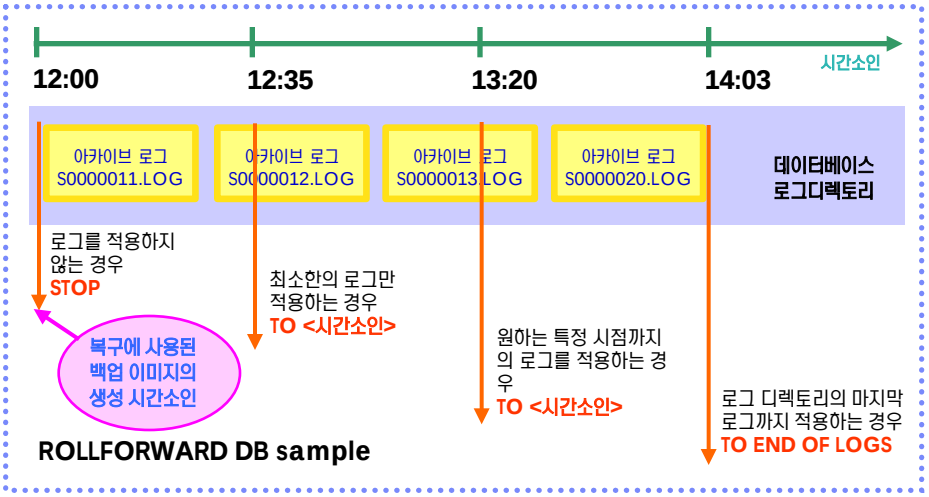


Figure 1323A ··· ROLLFORWARD DB 명령어를 이용한 로그 재적용의 범위

Tip

ROLLFORWARD 명령은 여러 번 반복적으로 실행할 수 있습니다.

Tip

END OF LOGS 옵션은 로그 디렉토리에 존재하는 로그 파일을 모드 적용시키는 옵션입니다. 최근의 로그 파일을 의미하는 것이 아닙니다.

2 최초의 rollforward db 명령어는 버전 복구를 완료된 이후에 시작할 수 있습니다.

```
$ db2 restore db <DB명>
$ db2 rollforward db <DB명> to end of logs
```

3 적용할 로그 파일의 크기가 크고, 개수가 많은 경우에는 로그 디렉토리에 로그 파일의 일부만 옮겨 두고, 여러 번에 걸쳐 rollforward db 명령어를 실행합니다.

```
$ db2 rollforward db <DB명> to end of logs
```

Tip

USING LOCAL TIME 옵션이 없으면, GMT 시간을 명시해야 합니다.

4 마지막 rollforward db 명령을 입력할 때, TO 옵션을 이용하여 원하는 시점까지의 복구가 가능합니다. <시간소인>은 USING LOCAL TIME 옵션으로 지정합니다.

```
$ db2 rollforward db <DB명> to <시간소인> using local time
```

Tip

<시간 소인>은 yyyy-mm-dd-hh:mm:ss 형태로 지정합니다.

5 STOP 옵션을 이용하면, 로그 재적용 작업은 중지되고, 마지막으로 적용된 로그 파일의 다음 번호부터 데이터베이스 로깅이 다시 시작됩니다.

```
$ db2 rollforward db <DB명> stop
```

Tip

rollforward db 명령어는 <Op> 항목에 'F' 라고 표시됩니다. END OF LOGS 옵션은 'E', <시간소인> 옵션은 'P' 로 표시됩니다.

6 list history 명령어로 해당 롤포워드 복구와 연관된 기록을 확인합니다.

```
$ db2 list history rollforward for <DB명>
```

TOPIC 13-24 INCREMENTAL 복구

Point

최근의 FULL 백업 이미지 파일과 FULL 백업 이후에 생성된 최근의 INCREMENTAL 백업 이미지 파일, INCREMENTAL 백업 시점 이후의 아카이브 로그 파일이 보관되어 있으면, INCREMENTAL 복구가 가능합니다.

Tip

SYSADM,SYSCTRL,SYSMAINT 권한이 있는 사용자가 실행합니다.

1 restore db 명령어로 최근의 FULL 백업 이미지를 이용하여 버전 복구를 합니다. restore db 명령어로 FULL 백업 시점 이후의 INCREMENTAL 백업 이미지 중에서 최근 것으로 FULL 백업과 INCREMENTAL 백업 시점 구간에 있는 변경 내용을 복구합니다. 마지막으로 INCREMENTAL 백업 이후 시점부터 현재까지의 아카이브 로그를 차례로 적용하면 데이터베이스는 손상 직전의 상태로 복구됩니다.

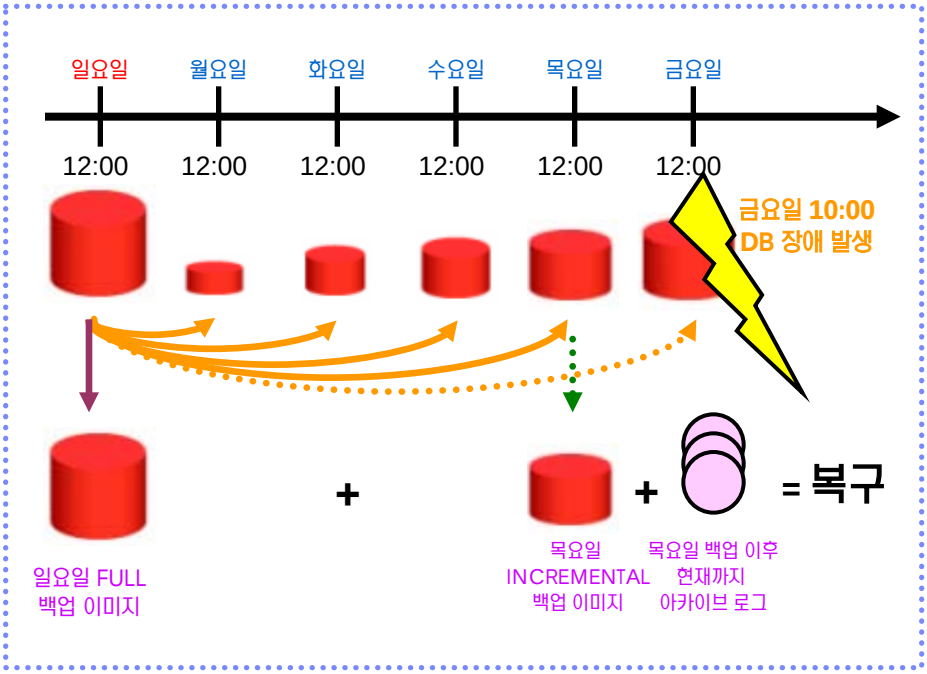


Figure 1324A FULL, INCREMENTAL 이미지, 아카이브 로그를 이용한 복구

Tip

INCREMENTAL AUTOMATIC 옵션을 이용하면, 복구 기록 파일을 이용하여 필요한 INCREMENTAL 백업 이미지를 자동적으로 찾아서 복구를 실행합니다.

2 최근의 FULL 백업 이미지 파일을 이용하여 restore database 명령으로 데이터베이스를 이전 시점으로 버전 복구합니다.

```
$ db2 restore db <DB명>
```

3 restore database 명령의 INCREMENTAL 옵션으로 최근의 INCREMENTAL 백업 이미지 파일을 이용하여 FULL 백업 이후의 변경분을 복구합니다.

```
$ db2 restore db <DB명> INCREMENTAL
```

Tip

손상 직전의 시점으로 복구하려면, 마지막 INCREMENTAL 백업 이후부터 손상 직전 시점까지의 아카이브 로그 파일을 로그 디렉토리에 모두 복사하고, TO END OF LOGS AND STOP 옵션을 사용합니다.

4 rollforward db 명령으로 아카이브 로그 파일을 이용하여 최근의 INCREMENTAL 백업 이미지 생성 시점 이후부터 원하는 시점까지의 변경분을 복구합니다.

```
$ db2 rollforward db <DB명> to <시간소인> using local time and stop
```

5 list history 명령어로 해당 롤포워드 복구와 연관된 기록을 확인합니다.

```
$ db2 list history rollforward for <DB명>
```

TOPIC 13.25 DELTA 복구

Point

최근의 FULL 백업 이미지 파일과 INCREMENTAL 백업 또는 DELTA 백업 이미지 파일, 데이터베이스 로그 파일이 보관되어 있으면, INCREMENTAL 복구가 가능합니다.

Tip

SYSADM,SYSCTRL,SYSMAINT 권한이 있는 사용자가 실행합니다.

1. 최근의 FULL 백업 이미지 파일, 최근의 INCREMENTAL 백업 이미지, 현 시점까지의 DELTA 백업 이미지 파일, 아카이브 로그 파일을 준비하고, restore db 명령어와 rollforward db 명령어를 실행합니다.

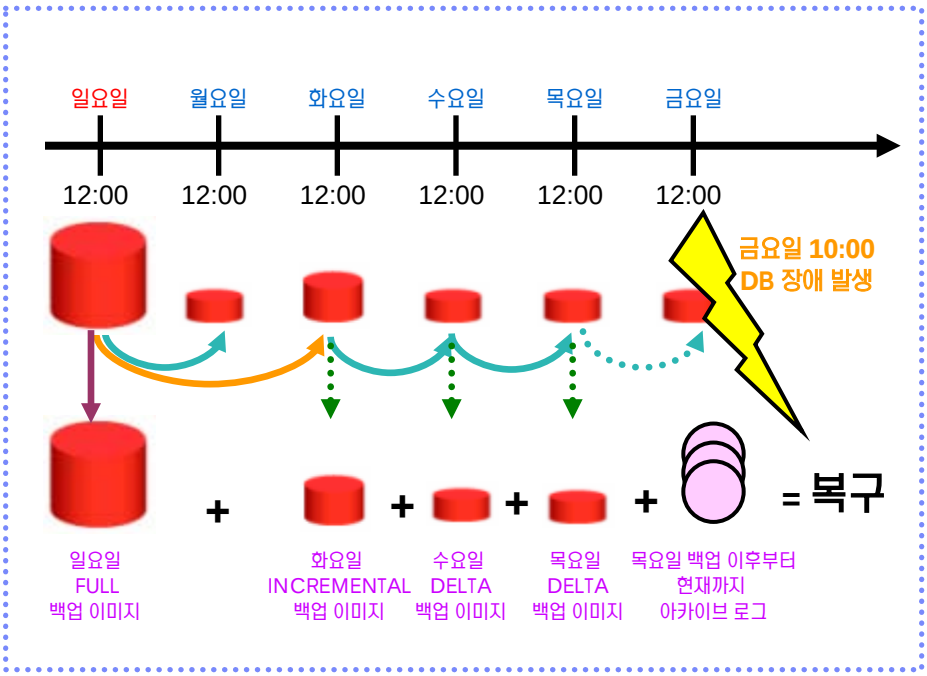


Figure 1325A FULL, INCREMENTAL, DETA 이미지, 아카이브 로그를 이용한 복구

Tip

restore db 명령어에서 사용하는 옵션은 INCREMENTAL 복구와 동일합니다.

2. 최근의 FULL 백업 이미지 파일을 이용하여 Restore database 명령으로 데이터베이스를 과거의 시점으로 복구합니다.

```
$ db2 restore db <DB명>
```

Tip

INCREMENTAL AUTOMATIC 옵션을 이용하면, 복구 기록 파일을 이용하여 필요한 INCREMENTAL 백업 이미지를 자동적으로 찾아서 복구를 실행합니다.

3. 최근의 INCREMENTAL 백업 이미지 파일이 있다면 restore db 명령으로 FULL 백업 이후의 변경분을 복구합니다. restore db 명령을 반복적으로 사용하여 최근까지의 DELTA 백업 이미지 파일들로부터 변경분을 복구합니다.

```
$ db2 restore db <DB명> INCREMENTAL
```

Tip

손상 직전의 시점으로 복구하려면, 마지막 INCREMENTAL 백업 이후부터 손상 직전 시점까지의 아카이브 로그 파일을 로그 디렉토리에 모두 복사하고, TO END OF LOGS AND STOP 옵션을 사용합니다.

4. 데이터베이스 로그 파일을 이용하여 rollforward database 명령으로 최근의 DELTA 백업 이미지 생성 시점 이후의 변경분을 복구합니다.

```
$ db2 rollforward db <DB명> to end of logs
$ db2 rollforward db <DB명> to <시간소인> using local time and stop
```

5. list history 명령어로 해당 롤포워드 복구와 연관된 기록을 확인합니다.

```
$ db2 list history rollforward for <DB명>
```

TOPIC 13-26 테이블스페이스의 상태

Point

테이블 스페이스의 컨테이너 액세스에 문제가 발생하면, 테이블 스페이스의 상태가 OFFLINE이 되어 해당 테이블 스페이스를 액세스할 수 없습니다. 문제가 발생한 테이블스페이스를 제외한 다른 테이블스페이스는 사용 가능합니다.

1 컨테이너에 문제가 발생한 테이블스페이스는 OFFLINE 상태가 되고, 사용할 수 없습니다.

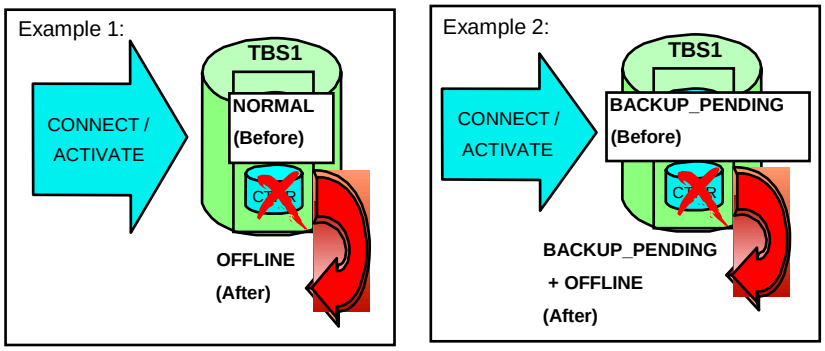


Figure 1326A ••• 테이블스페이스 오프라인

2 테이블 스페이스의 현재 상태를 확인합니다.

```
$ db2 "SELECT * FROM <테이블명>"
  SQL0290N  테이블 스페이스 액세스가 허용되지 않습니다.  SQLSTATE=55039
$ db2 list tablespaces | grep –p –i <테이블스페이스명>
  테이블 스페이스 ID                    = 3
  이름                                  = <테이블스페이스명>
  유형                                  = 데이터베이스 관리 스페이스
  내용                                  = 임의의 데이터
  상태                                  = 0x4000
  세부사항 설명:
    오프라인
$ db2tbst 0x4000
State = Offline and not accessible
```

3 테이블스페이스 컨테이너의 문제점을 해결하고, alter tablespace 문을 이용하여 테이블스페이스의 상태를 ON 으로 변환하고, 데이터를 정상적으로 액세스할 수 있습니다. 컨테이너의 문제점이 해결되지 않으면, SQL0293N 오류 코드와 함께 명령은 실패합니다.

```
$ db2 ALTER TABLESPACE <테이블스페이스명> SWITCH ONLINE
```

4 특정한 테이블스페이스 컨테이너에 문제점이 발생하여 데이터베이스가 활성화될 수 없다면, restart db 명령어를 이용하여 해당 테이블스페이스만 DROP PENDING 상태로 전환하고, 데이터베이스를 활성화할 수 있습니다.

```
$ db2 "RESTART DATABASE <DB명> DROP PENDING (<TS명>)"
$ db2 drop tablespace <TS명>
```

Tip

테이블스페이스의 주요 상태값은 다음과 같습니다.

0x0	Normal
0x1	Quiesced: SHARE
0x2	Quiesced: UPDATE
0x4	Quiesced: EXCLUSIVE
0x8	Load pending
0x10	Delete pending
0x20	Backup pending
0x40	Rollforward in progress
0x80	Rollforward pending
0x100	Restore pending
0x200	Disable pending
0x400	Reorg in progress
0x800	Backup in progress
0x1000	Storage must be defined
0x2000	Restore in progress
0x4000	Offline
0x8000	Drop pending

Tip

DROP PENDING 상태로 전환된 테이블스페이스는 접속 후에 DROP 하고 재생성해야 합니다.

TOPIC 13.27 테이블스페이스 복구

Point

RESTORE DB 명령어로 테이블스페이스 백업 이미지 파일을 이용하여 특정 테이블스페이스만 과거 시점으로 복구하고, ROLLFORWARD DB 명령어로 아카이브 로그를 적용하여 다른 테이블스페이스와의 일관성을 복구합니다.

Tip

SYSADM,SYSCTRL,SYSMAINT 권한이 있는 사용자가 실행합니다.

1 원하는 시점의 TABLESPACE 백업 이미지 파일과 아카이브 로그 파일을 준비하고, restore db 명령어와 rollforward db 명령어를 실행하면, 테이블스페이스를 복구할 수 있습니다.

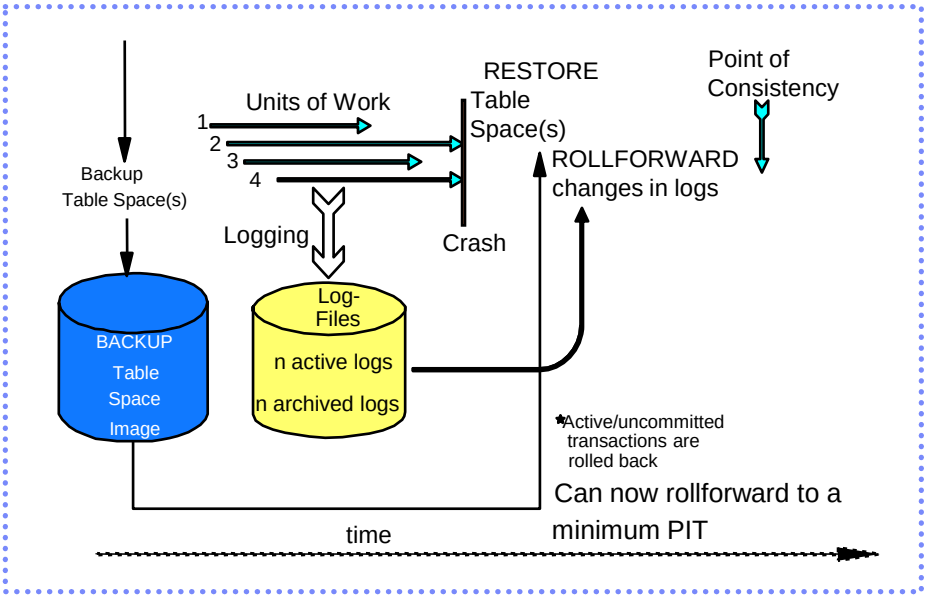


Figure 1327A ··· 테이블스페이스 백업 이미지를 이용한 복구

Tip

일관성을 보장하기 위해 적용되어야 할 최소한의 로그 파일 번호는 LIST HISTORY 명령어를 이용하여 복구 기록 파일에서 확인할 수 있습니다.

2 원하는 TABLESPACE 백업 이미지 파일을 이용하여 restore db 명령으로 데이터베이스의 특정 테이블스페이스를 과거 시점으로 복구합니다. restore db 명령어가 완료되면, 해당 테이블스페이스가 '롤포워드 보류' 상태에 놓이게 됩니다.

```
$ db2 "restore db <DB명> TABLESPACE (<TS명>) ONLINE"
$ db2 connect to <데이터베이스명>
$ db2 "SELECT * FROM <테이블명>"
SQL0290N  테이블 스페이스 액세스가 허용되지 않습니다.  SQLSTATE=55039
$ db2 list tablespaces | grep -p –i <테이블스페이스명> | grep "상태"
 상태                                = 0x0080
```

Tip

RESTORE DB 명령어 또는 ROLLFORWARD DB 명령어를 실행한 세션의 데이터베이스에 대한 접속은 해제됩니다.

3 rollforward db 명령어의 TO 옵션으로 해당 테이블스페이스와 다른 테이블스페이스의 일관성을 보장할 수 있는 시점 이상의 로그 파일을 복구합니다.

```
$ db2 ROLLFORWARD DB <DB명> TO END OF LOGS AND STOP
  TABLESPACE (<TS명>) ONLINE"
SQL4907W  데이터베이스 "<데이터베이스명>"이(가) 복구되었으나, 롤 포워드 조작을
  위해 포함된 테이블 스페이스 목록에서 하나 이상의 테이블이 점검 보류 상태에 있습니다.
```

Tip

복구된 테이블스페이스에 존재하는 테이블이 다른 테이블과 RI 관계에 있는 부모 테이블이라면, 자손 테이블은 '점검 보류' 상태에 놓이게 됩니다. SET INTEGRITY 문을 이용하여 해결합니다.

4 복구된 테이블스페이스에 포함된 테이블의 데이터를 정상적으로 액세스할 수 있습니다.

```
$ db2 "SELECT * FROM <테이블명>"
```

UNIT 14

모니터링

데이터베이스와 응용프로그램에 대한 모니터링은 데이터베이스의 효율적인 운영과 장애 예방을 대비하는 중요한 대책입니다. DB2는 특정 시점의 모니터링 정보를 위한 스냅샷 모니터와 스냅샷 함수를 제공하며, 일정 기간 동안의 모니터링 정보 수집을 위한 이벤트 모니터를 제공합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 오류 진단 파일
- 시스템 모니터 스위치
- 세션별 모니터 스위치
- 스냅샷 모니터
- 스냅샷 테이블 함수
- 응용프로그램 목록
- 응용프로그램이 사용한 CPU 시간
- 응용프로그램이 처리한 행의 수
- 응용프로그램별 잠금
- 파티션별 잠금
- 테이블별 잠금
- 잠금 대기 에이전트
- 잠금 대기 에이전트의 정적 SQL문
- 잠금 대기 에이전트의 동적 SQL문
- 잠금 보유 에이전트의 정적 SQL문
- 잠금 보유 에이전트의 동적 SQL문
- 응용프로그램별 로그 사용량
- 데이터베이스별 로그 사용량
- 테이블 스페이스 사용량
- 테이블 스페이스 컨테이너 사용량
- 테이블 스페이스 적중률
- 이벤트 모니터
- CREATE EVENT MONITOR 문
- 파일 이벤트 모니터
- 테이블 이벤트 모니터
- 시간소요 모니터
- db2pd 모니터링
- db2top
- db2top - Application
- db2top - Memory
- db2top - Lock
- db2top - Table
- db2top - Partitioning
- db2top - Dynamic SQL
- db2top - Utility
- db2top - Tablespace

TOPIC 14-01 오류 진단 파일

Point

데이터베이스 시스템 오류가 발생하면 db2diag.log 라는 파일에 기록됩니다. DIAGLEVEL과 DIAGPATH 인스턴스 구성 변수를 이용하여 기록할 오류의 수준과 진단 파일을 저장할 디렉토리를 결정합니다.

Tip

db2diag.log가 생성되는 기본 디렉토리는 <인스턴스의 홈디렉토리>/sqllib/db2dump입니다.

Tip

DIAGLEVEL의 기본값은 3으로 심각한 오류, 일반 오류, 경고 메시지를 기록합니다.

Tip

db2diag.log 를 비롯한 진단 파일은 삭제해도 무방합니다. 필요시에 자동적으로 새로운db2diag.log 파일이 다시 생성됩니다.

1 인스턴스 사용자로 로그인하여 db2diag.log 파일을 확인합니다.

```
$ login <인스턴스 사용자명>
$ more  $HOME/sqllib/db2dump/db2diag.log
```

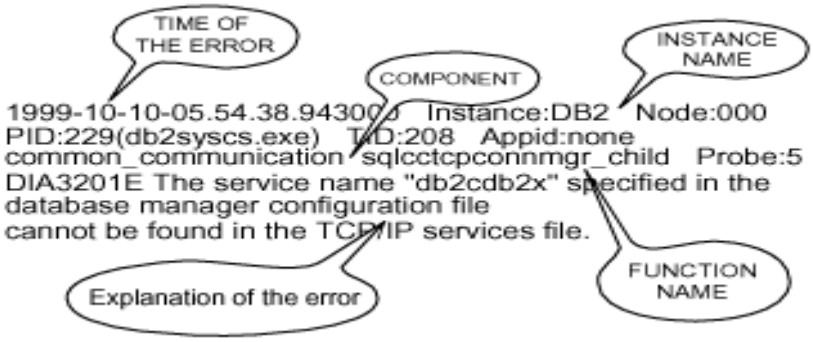


Figure 1401A ••• db2diag.log 파일

Tip

DIAGPATH 구성 변수에서 지정한 디렉토리는 미리 생성되어야 합니다.

2 update dbm cfg 명령어로 인스턴스 구성 변수인 DIAGLEVEL과 DIAGPATH를 변경하면, db2diag.log 에 기록되는 진단 정보의 수준과 db2diag.log 를 비롯한 시스템 오류 덤프 파일이 저장되는 디렉토리가 변경됩니다.

```
$ db2 attach to <인스턴스명>
$ db2 update dbm cfg using DIAGLEVEL <진단 수준>
$ db2 update dbm cfg using DIAGPATH <진단 디렉토리명>
$ db2 get dbm cfg | grep DIAG
```

Tip

심각한 오류가 발생한 경우에는 trap(tnnnnn.000), dump(nnnnn.dmp), core 파일 등이 추가적으로 생성됩니다.

Tip

IBM에 문제 해결을 요청할 때는 반드시 db2diag.log를 비롯한 trap, dump, core 파일 등을 보관하고 있어야 합니다.

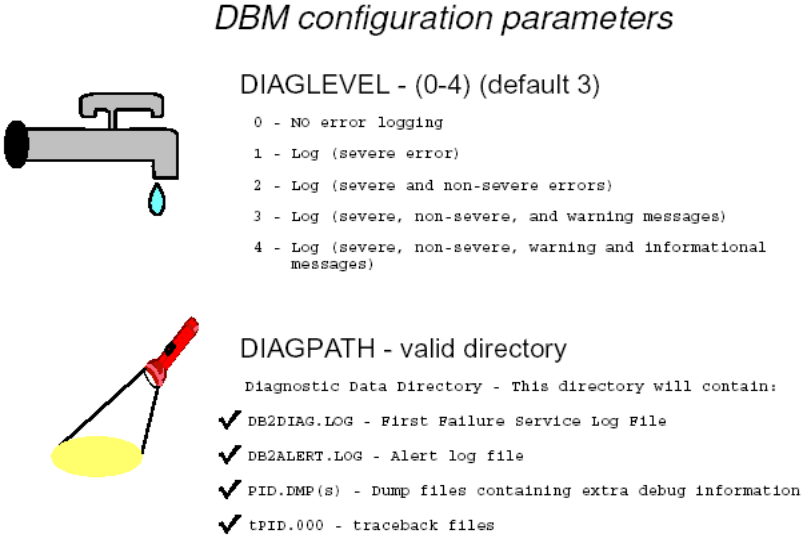


Figure 1401B ••• DIAGLEVEL과 DIAGPATH 인스턴스 구성 변수

TOPIC 1401 오류 진단 파일

Point

파일시스템 저장공간이 가득 차는 것을 방지하기 위하여 db2diag.log 파일의 크기를 제한할 수 있습니다.

1 아래의 명령을 수행하면 1024 메가 바이트 중 diaglog파일이 90%, db2<instance>.nfy파일이 10%의 비율로 생성되며 파일사이즈의 합이 1024 메가 바이트로 파일들이 생성됩니다. 파일은 rotating 방식으로 생성되며 db2diag.0.log 부터 순차적으로 10개의 파일이 생성됩니다.

```
$ db2 update dbm cfg using DIAGSIZE 1024
```

```
$ ls -l
total 8696
-rw-rw-rw- 1 db21r1 dblr1adm 172781     2009-06-03 10:02 db2diag.0.log
-rw-rw-rw- 1 db21r1 dblr1adm 172839     2009-06-03 10:07 db2diag.1.log
-rw-rw-rw- 1 db21r1 dblr1adm 172716     2009-06-03 10:12 db2diag.2.log
-rw-rw-rw- 1 db21r1 dblr1adm 172360     2009-06-03 10:18 db2diag.3.log
-rw-rw-rw- 1 db21r1 dblr1adm 172862     2009-06-03 10:23 db2diag.4.log
-rw-rw-rw- 1 db21r1 dblr1adm 172635     2009-06-03 12:07 db2diag.5.log
-rw-rw-rw- 1 db21r1 dblr1adm 93338      2009-06-03 14:16 db2diag.6.log
-rw-rw-rw- 1 db21r1 dblr1adm 3362       2009-06-03 09:21 db2diag.log
-rw-r----- 1 db21r1 dblr1adm 6291312    2009-06-03 14:16 db2eventlog.000
-rw-rw-rw- 1 db21r1 dblr1adm 2436       2009-06-03 14:16 db21r1.0.nfy
-rw-rw-rw- 1 db21r1 dblr1adm 1074       2009-06-03 09:21 db21r1.nfy
drwxrwxr-t 2 db21r1 dblr1adm 4096       2009-06-03 09:47 events
drwxrwxr-t 2 db21r1 dblr1adm 4096       2009-06-03 09:46 stmmlog
```

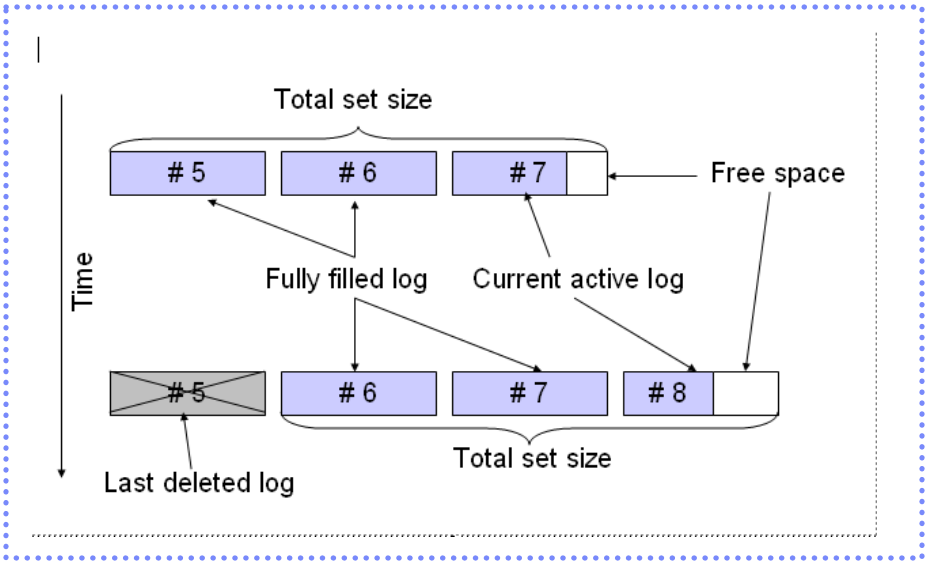


Figure 1401B • • • Rotating Diagnostic logs

TOPIC 1402 시스템 모니터 스위치

Point

모니터링 자료의 수집 여부를 결정하는 시스템 모니터 스위치는 7가지 인스턴스 구성 변수를 이용하여 조절합니다. 인스턴스를 생성한 직후에는 DFT_MON_TIMESTAMP 구성 변수를 제외한 다른 모니터 스위치의 기본값은 OFF 입니다.

Tip

DFT_MON_TIMESTAMP는 DB2 엔진이 운영 체제를 호출하여 모든 명령문의 실행 전후에 시간소인을 확보하도록 하는 스위치로 많은 비용을 소모합니다. CPU 사용율이 심하게 높은 경우에는 OFF로 설정합니다.

Tip

모든 모니터링 응용프로그램은 응용프로그램이 첫 번째 모니터링 요청을 발행할 때 기본 스위치 설정값을 적용하고, 그 설정값을 계속 상속합니다.

1 get dbm cfg 명령어를 이용하여 기본 모니터 스위치의 현재 설정값을 확인합니다. DFT_MON_BUFPOOL, DFT_MON_LOCK, DFT_MON_SORT, DFT_MON_STMT, DFT_MON_TABLE, DFT_MON_UOW, DFT_MON_TIMESTAMP 등의 7가지 기본 모니터 스위치가 제공됩니다.

```
$ login <인스턴스 사용자>
$ db2 get dbm cfg | grep DFT
```

2 시스템 모니터 스위치를 활성화시키려면 update dbm cfg 명령어로 모니터 스위치의 값을 ON 또는 OFF 로 설정합니다. 변경 이후에 시작하는 모든 세션에는 변경된 시스템 모니터 스위치의 값이 적용됩니다. 값이 ON 인 경우에는 해당 항목에 대한 추가적인 모니터링 자료가 수집되고, OFF 인 경우에는 추가적인 모니터링 자료에 대한 수집이 중지됩니다.

```
$ db2 attach to <인스턴스명>
$ db2 update dbm cfg using <기본 모니터 스위치 변수명> ON
$ db2 update dbm cfg using <기본 모니터 스위치 변수명> OFF
$ db2 detach
```

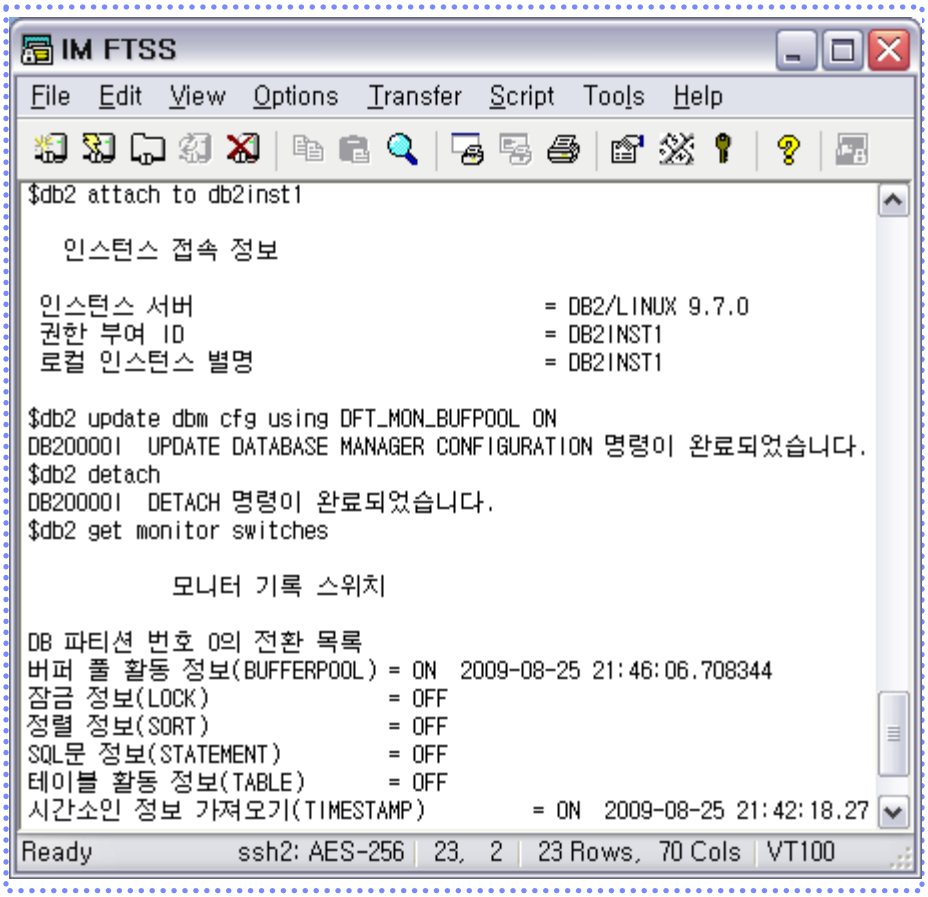


Figure 1402A ••• 시스템 모니터 스위치의 설정값

TOPIC 1403 세션별 모니터 스위치

Point

UPDATE MONITOR SWITCHES 명령어로 현재 세션의 모니터 스위치의 값을 설정하면, 인스턴스 구성 변수에 설정된 시스템 모니터 스위치의 값보다 우선적으로 적용됩니다.

Tip

- 변경된 모니터 스위치의 값은 현재의 세션에만 적용됩니다. terminate 명령으로 세션이 종료되면 모니터 스위치는 기본값으로 복원됩니다.

1 get monitor switches 명령어를 이용하여 모니터링 스위치의 현재 설정값을 확인합니다.

```
$ login <인스턴스 사용자>
$ db2 get monitor switches
```

2 새로운 세션을 시작하기 전에 update monitor switches 명령어로 7가지의 모니터 스위치 중에서 원하는 모니터 스위치를 ON 또는 OFF 로 설정합니다.

```
$ db2 update monitor switches using <모니터 스위치명> ON
$ db2 update monitor switches using <모니터 스위치명> OFF
$ db2 get monitor switches
```

3 reset monitor 명령어는 모니터링 정보 중에서 엔진이 관리하는 내부적인 누적값을 표시하는 항목의 현재값을 초기값으로 갱신합니다.

```
$ db2 reset monitor all
$ db2 reset monitor for db <데이터베이스명>
```

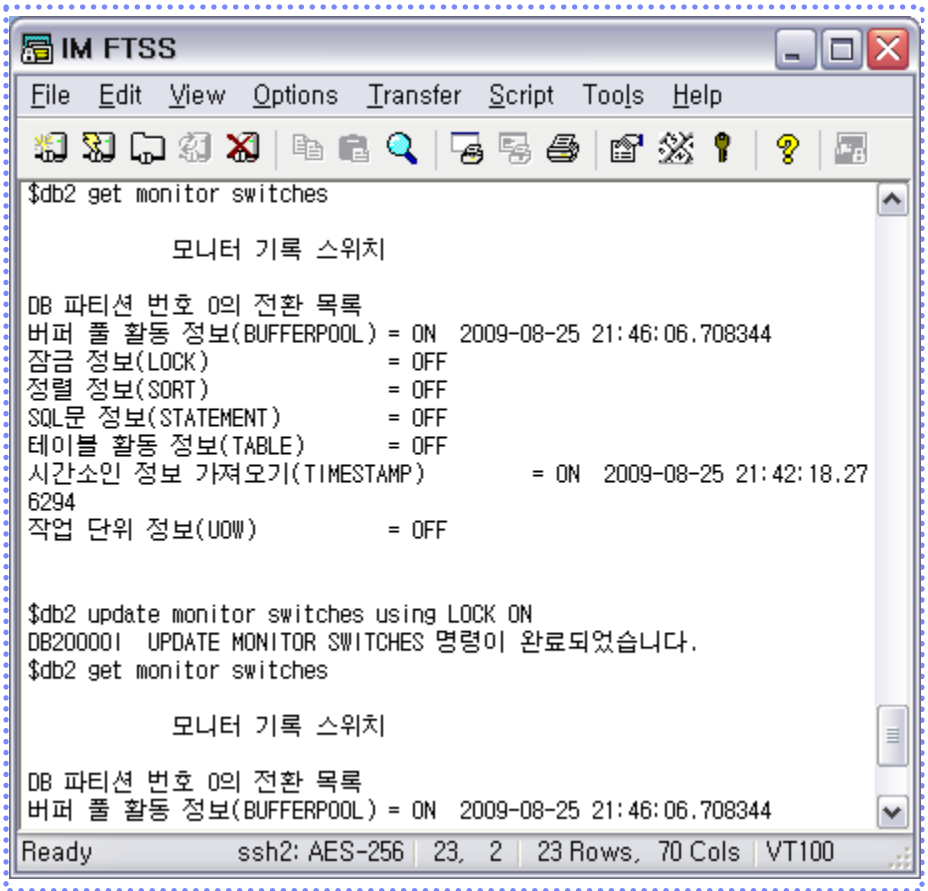


Figure 1403A ••• 현재 세션의 모니터 스위치 설정값

TOPIC 1404 스냅샷 모니터

Point

GET SNAPSHOT 명령어를 이용하여 현재 시점의 데이터베이스의 활동 상황을 모니터링할 수 있습니다. 모니터링의 대상은 인스턴스, 데이터베이스, 응용프로그램이 될 수 있습니다. 세션의 모니터 스위치의 설정값에 따라 표시되는 텍스트 정보가 달라집니다.

Tip

인스턴스, 데이터베이스, 응용프로그램, 테이블 스페이스, 테이블, 잠금, 버퍼풀, 동적 SQL문 등에 대한 모니터링 정보가 수집됩니다.

Tip

모니터링 항목은 현재값 또는 누적값을 표시합니다.

New

9.7에서는 Health Monitor는 더 이상 지원하지 않습니다. (첨부참고)

1. 인스턴스 사용자로 로그인합니다.

```
$ login <인스턴스 사용자명>
```

2. get snapshot 명령어를 이용하여 인스턴스의 활동 내역을 모니터링할 수 있습니다.

```
$ db2 get snapshot for dbm
```

3. get snapshot 명령어를 이용하여 데이터베이스의 활동 내역을 모니터링할 수 있습니다.

```
$ db2 get snapshot for all on <데이터베이스명>
$ db2 get snapshot for db on <데이터베이스명>
$ db2 get snapshot for locks on <데이터베이스명>
```

4. get snapshot 명령어를 이용하여 응용프로그램의 활동 내역을 모니터링할 수 있습니다.

```
$ db2 list applications
$ db2 get snapshot for application agentid  <응용프로그램 핸들 번호>
```

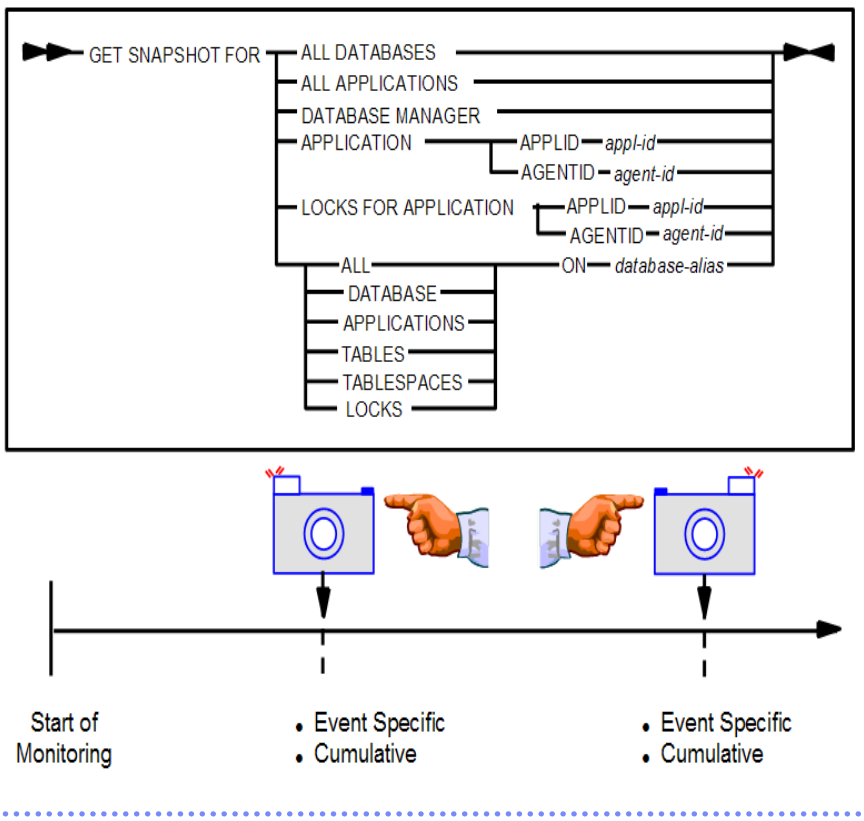


Figure 1404A... GET SNAPSHOT 명령어

TOPIC 1405 스냅샷 테이블 함수

Point

GET SNAPSHOT 명령어의 실행 결과와 동일한 스냅샷 정보를 테이블의 형태로 저장하는 테이블 함수입니다. SELECT 문의 FROM 절에서 표현되고, 일반 테이블과 동일한 방법으로 조회할 수 있습니다.

1 대표적인 스냅샷 테이블 함수의 종류는 다음과 같습니다.

함 수	설 명
SNAPSHOT_AGENT	응용프로그램 스냅샷에서 응용프로그램의 핸들 번호와 대응되는 db2agent 프로세스의 pid를 반환합니다.
SNAPSHOT_APPL	응용프로그램 스냅샷에서 응용프로그램에 대한 일반적인 정보를 반환합니다.
SNAPSHOT_APPL_INFO	응용프로그램 스냅샷에서 응용프로그램의 핸들 번호와 이름, 응용프로그램 프로세스의 pid 등을 반환합니다.
SNAPSHOT_BP	버퍼풀 스냅샷에서 버퍼풀의 활동 정보를 반환합니다.
SNAPSHOT_CONTAINER	테이블 스냅샷에서 컨테이너 구성 정보를 반환합니다.
SNAPSHOT_DATABASE	데이터베이스 스냅샷에서 정보를 반환합니다.
SNAPSHOT_DBM	인스턴스 스냅샷에서 정보를 반환합니다.
SNAPSHOT_DYN_SQL	동적 SQL 스냅샷에서 정보를 반환합니다
SNAPSHOT_FCM	FCM에 관한 정보를 반환합니다.
SNAPSHOT_LOCK	잠금 스냅샷에서 정보를 반환합니다.
SNAPSHOT_LOCKWAIT	응용프로그램 스냅샷에서 잠금 대기 정보를 반환합니다.
SNAPSHOT_STATEMENT	응용프로그램 스냅샷에서 명령문의 정보를 반환합니다.
SNAPSHOT_TABLE	테이블 스냅샷에서 활동 정보를 반환합니다.
SNAPSHOT_TBS	테이블 스페이스 스냅샷에서 활동 정보를 반환합니다.
SNAPSHOT_TBS_CFG	테이블 스페이스 스냅샷에서 구성 정보를 반환합니다

2 get dbm cfg 명령어로 시스템 모니터 스위치의 기본값을 확인합니다.

```
$ db2 get dbm cfg | grep DFT_MON
```

3 get monitor switches 명령어를 이용하여 현재 세션의 모니터링 스위치의 값을 확인합니다.

```
$ db2 get monitor switches
```

4 SELECT문의 FROM 절에서 TABLE 이라는 키워드를 사용하여 스냅샷 함수를 실행합니다. 인수로 <데이터베이스명>과 <데이터베이스 파티션 번호>를 입력합니다. 파티션 번호가 -1 이면 현재 파티션을 의미하고, -2 이면 모든 파티션을 의미합니다. <결과 테이블에 대한 별명>은 임의로 지정합니다.

```
SELECT * FROM TABLE(SNAPSHOT_APPL_INFO('SAMPLE',-1)) TFSNAP
```

TABLE() function — table function — alias

Figure 1405A ··· 스냅샷 테이블 함수의 사용 방법

Tip

스냅샷 함수의 실행 결과를 저장할 임시 테이블의 별명을 지정하지 않으면 SELECT문은 실패합니다.

Tip

결과로 반환된 테이블은 SELECT문이 실행되는 동안에만 존재합니다.

TOPIC 1406 응용프로그램 목록 확인

Point

데이터베이스에 접속하고 있는 응용프로그램에 대한 정보를 확인합니다. 응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, 응용프로그램 ID, 상태, 에이전트 개수, 데이터베이스명, 프로세스 ID 를 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$MON_APPL_ID 는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.auth_id,1,8) as authid
    ,cast(appl.agent_id as integer) as agentid
    ,substr(appl_name,1,20) as appl_name
    ,substr(appl_info.appl_id,1,30) applid
    ,case appl_info.appl_status
        when 2 then 'Connection Completed'
        when 3 then 'Executing'
        when 4 then 'UOW Wait'
        when 5 then 'Lock Wait'
        when 9 then 'Compile'
        when 24 then 'Pending remote request'
        when 26 then 'Decoupled'
        else substr(char(appl_info.appl_status),1,10)
    end as status
    ,cast(appl.num_agents as integer) num_agents
    ,substr(appl_info.db_name,1,8) dbname
    ,cast(appl_info.client_pid as integer) client_pid
from    table(snapshot_appl('$DBNAME', $DPMODE)) as appl,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where appl.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
order by appl.num_agents desc, appl_name, agentid;
```

2 실행 결과는 다음과 같습니다.

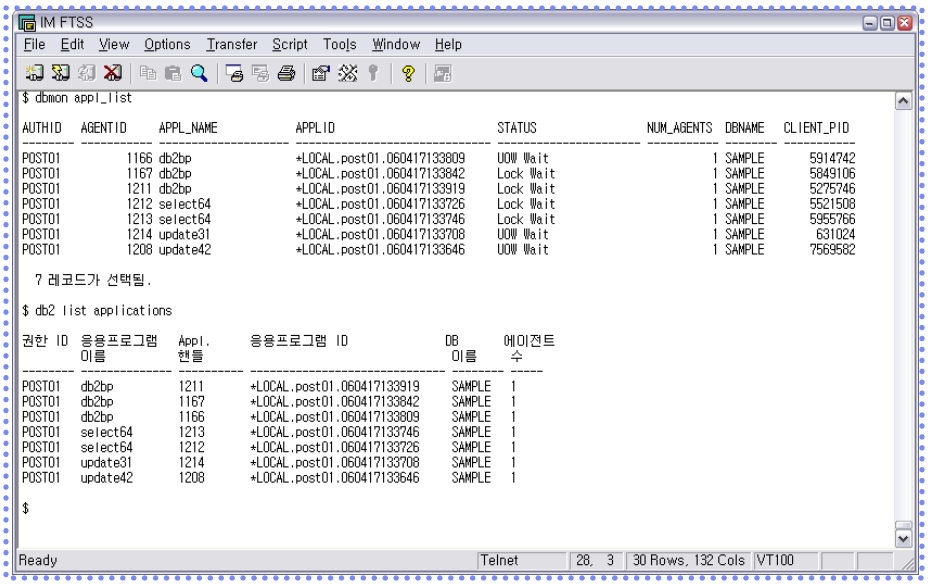


Figure 1406A ··· 응용프로그램 목록 확인

TOPIC 1407 응용프로그램이 사용한 CPU 시간

Point

응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, User CPU, System CPU, 경과 시간, 유휴 시간, 읽은 행수, 기록한 행수를 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$MON_APPL_ID는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.auth_id,1,8) as authid
    ,cast(appl.agent_id as integer) as agentid
    ,substr(appl_info.appl_name,1,20) appl_name
    ,cast(appl.agent_usr_cpu_time_s as integer) as user_cpu
    ,cast(appl.agent_sys_cpu_time_s as integer) as sys_cpu
    ,timestampdiff( 2, char( current timestamp - appl.uow_start_time ))
    as elapsed_time
    ,cast(appl_idle_time as integer) idle_time
    ,appl.rows_read
    ,appl.rows_written
from table(snapshot_appl('$DBNAME', $DPMODE)) as appl,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where appl.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
and uow_stop_time is null
order by user_cpu desc, agentid with ur;
```

2 실행 결과는 다음과 같습니다.

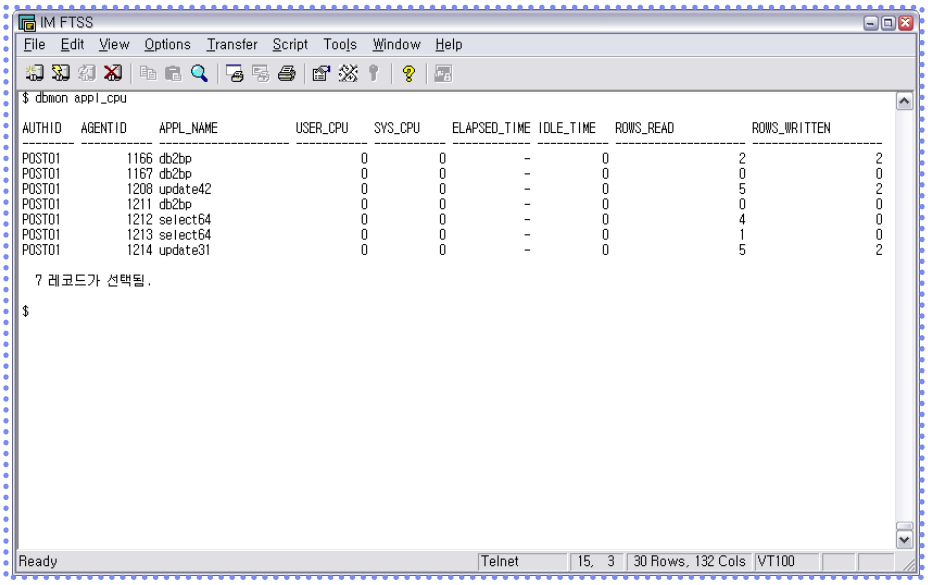


Figure 1407A ••• 응용프로그램이 사용한 CPU 시간

TOPIC 14-08 응용프로그램이 처리한 행의 수

Point

응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, 경과 시간, 조회 건수, 추가 건수, 갱신 건수, 삭제 건수, 읽은 행수, 기록한 행수를 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$MON_APPL_ID 는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.auth_id,1,8) as authid
    ,cast(appl.agent_id as integer) as agentid
    ,substr(appl_name,1,20) as appl_name
    ,timestampdiff( 2, char(current timestamp - uow_start_time))
    as elapsed_time
    ,cast(appl.rows_selected as integer) as rows_select
    ,cast(appl.rows_inserted as integer) as rows_insert
    ,cast(appl.rows_updated as integer) as rows_update
    ,cast(appl.rows_deleted as integer) as rows_delete
    ,cast(appl.rows_read as integer) rows_read
    ,cast(appl.rows_written as integer) rows_written
from table(snapshot_appl('$DBNAME', $DPMODE)) as appl,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where appl.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
and uow_stop_time is null
order by elapsed_time desc, agentid with ur;
```

2 실행 결과는 다음과 같습니다.

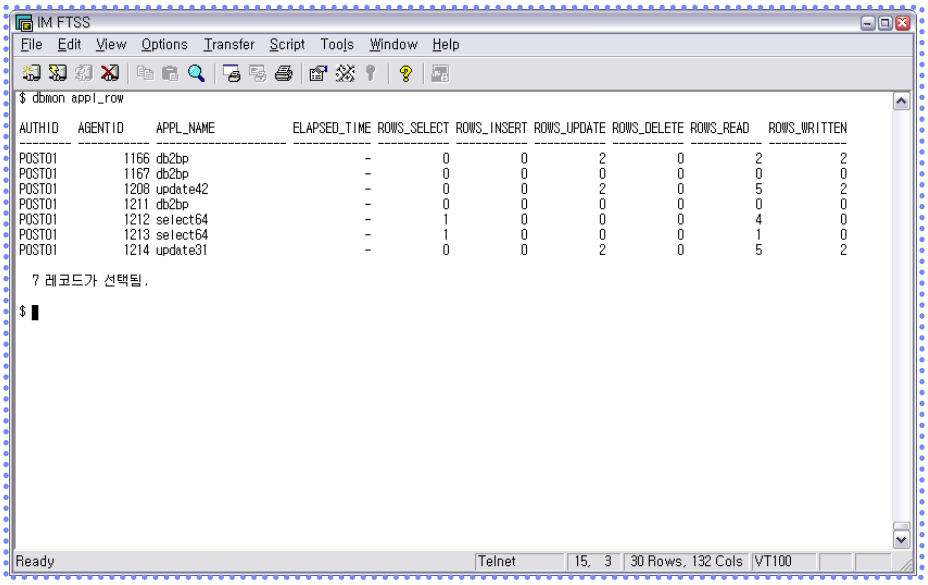


Figure 1408A ••• 응용프로그램이 처리한 행의 수

TOPIC 1409 응용프로그램별 잠금

Point

응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, 상태, 대기 잠금수, 보유 잠금수, escalation, 교착 상태 회수, 잠금 대기 여부 등을 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$MON_APPL_ID는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.auth_id,1,8) as authid
    ,cast(appl.agent_id as integer) as agentid,
    substr(appl_info.appl_name,1,14) as appl_name,
    case appl_info.appl_status
        when 2 then 'Connection Completed'
        when 3 then 'Executing'
        when 4 then 'UOW Wait'
        when 5 then 'Lock Wait'
        when 9 then 'Compile'
        when 24 then 'Pending remote request'
        when 26 then 'Decoupled'
        else substr(char(appl_info.appl_status),1,10)
    end as status,
    cast(appl.lock_waits as integer) as lock_waits,
    cast(appl.locks_held as integer) as locks_held,
    cast(appl.lock_escals as integer) as escals,
    cast(appl.x_lock_escals as integer) as x_escals,
    cast(appl.deadlocks as integer) as deadlock,
    cast(appl.locks_waiting as integer) as locks_waiting
from table(snapshot_appl('$DBNAME',$DPMODE)) as appl,
     table(snapshot_appl_info('$DBNAME',$DPMODE)) as appl_info
where appl.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
order by agentid;
```

2 실행 결과는 다음과 같습니다.

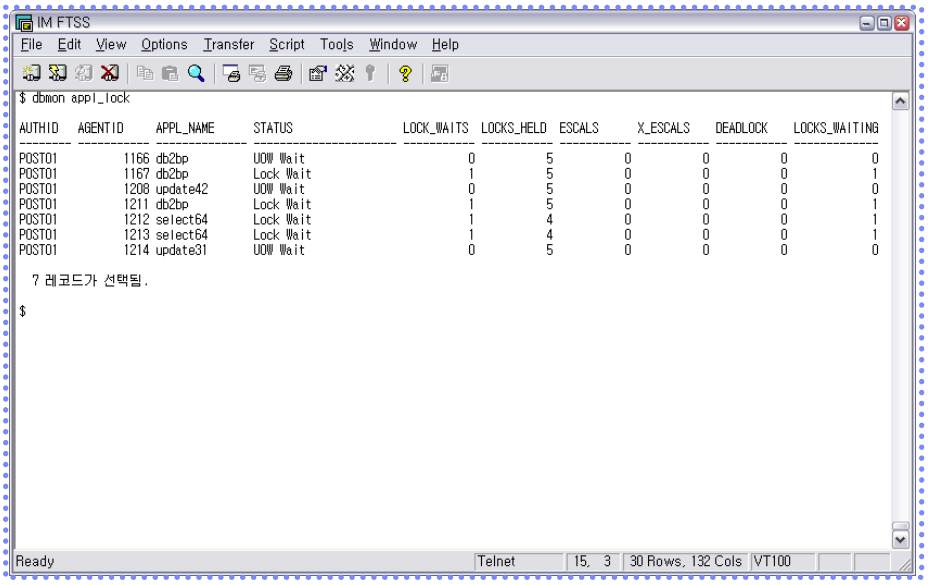


Figure 1409A ••• 응용프로그램별 잠금

TOPIC 14-10 파티션별 잠금

Point

특정 데이터베이스 파티션별로 응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, 테이블스페이스명, 테이블명, 잠금 수준, 잠금 대상, 잠금 모드, 잠금 상승 현상 발생 여부 등을 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 전체 파티션을 의미하는 -2 를 사용하지 말고, 특정 파티션 번호를 지정하도록 합니다.

Tip

단일 파티션을 사용한다면 <파티션번호> 에는 0 을 지정합니다.

Tip

$MON_APPL_ID 는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.AUTH_ID,1,10) as auth_id,
    cast(lock.agent_id as integer) as agentid,
    substr(appl_info.appl_name,1,16) as appl_name,
    substr(lock.tablespace_name,1,15) as tbsname,
    substr(lock.table_name,1,18) as tabname,
    lock.lock_object_type
        when 1 then 'Table'
        when 2 then 'Row'
        else substr(char(lock.lock_object_type),1,4)
    end  as type,
    cast(lock.lock_object_name as integer) as lockobjname,
    case cast(lock.lock_mode as smallint)
        when 3 then 'S'
        when 5 then 'X'
        when 9 then 'NS'
        else substr(char(lock.lock_mode),1,4)
    end as mode,
    case lock.lock_status
        when 1 then 'G'
        when 2 then 'C'
        else cast(lock.lock_status as char)
    end as status,
    cast(lock.lock_escalation as integer) as escal
from    table(snapshot_lock('$DBNAME', $DPMODE )) as lock,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where lock.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
order by agentid, tbsname, tabname;
```

2 실행 결과는 다음과 같습니다.

IM FTSS

File Edit View Options Transfer Script Tools Window Help

```
$ dbmon dp_lock 0

AUTH_ID    AGENTID     APPL_NAME        TBSNAME         TABNAME          TYPE                LOCKOBJNAME MODE STATUS ESCAL
---------- ----------- ---------------- --------------- ---------------- ------------------- ----------- ---- ------ -----------
POST01               7 update42         TS5             T1               Row                           5 X    G               0
POST01               7 update42         TS5             T1               Row                           7 X    G               0
POST01               7 update42         TS5             T1               Table                         2 IX   G               0
POST01               7 update42         -               -                Internal P Lock               0 S    G               0
POST01               7 update42         -               -                Internal P Lock               0 S    G               0
POST01               8 update31         TS5             T1               Row                           4 X    G               0
POST01               8 update31         TS5             T1               Row                           6 X    G               0
POST01               8 update31         TS5             T1               Table                         2 IX   G               0
POST01               8 update31         -               -                Internal P Lock               0 S    G               0
POST01               8 update31         -               -                Internal P Lock               0 S    G               0
POST01               9 select64         TS5             T1               Table                         2 IS   G               0
POST01               9 select64         -               -                Internal P Lock               0 S    G               0
POST01               9 select64         -               -                Internal P Lock               0 S    G               0
POST01              10 select53         TS5             T1               Table                         2 IS   G               0
POST01              10 select53         -               -                Internal P Lock               0 S    G               0
POST01              10 select53         -               -                Internal P Lock               0 S    G               0
POST01              11 db2bp            TS5             T1               Row                          10 X    G               0
POST01              11 db2bp            TS5             T1               Row                          11 X    G               0
POST01              11 db2bp            TS5             T1               Table                         2 IX   G               0
POST01              11 db2bp            -               -                Internal P Lock               0 S    G               0
POST01              11 db2bp            -               -                Internal P Lock               0 S    G               0
POST01              12 db2bp            TS5             T1               Table                         2 IS   G               0
POST01              12 db2bp            -               -                Internal V Lock               0 S    G               0
POST01              12 db2bp            -               -                Internal P Lock               0 S    G               0
POST01              12 db2bp            -               -                Internal P Lock               0 S    G               0
POST01              13 db2bp            TS5             T1               Table                         2 IX   G               0
```

Ready | Telnet | 30, 3 | 30 Rows, 132 Cols | VT100

Figure 1410A ··· 파티션별 잠금

TOPIC 14-11 테이블별 잠금

Point

특정 테이블별로 응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, 테이블스페이스명, 테이블명, 잠금 수준, 잠금 대상, 잠금 모드, 잠금 상승 현상 발생 여부 등을 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$TABNAME 은 <테이블명>으로 <스키마명>을 제외하고 지정합니다.

Tip

$MON_APPL_ID 는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.AUTH_ID,1,10) as auth_id,
    cast(lock.agent_id as integer) as agentid,
    substr(appl_info.appl_name,1,16) as appl_name,
    substr(lock.tablespace_name,1,15) as tbsname,
    substr(lock.table_name,1,18) as tabname,
    case lock.lock_object_type
        when 1 then 'Table'
        when 2 then 'Row' as type,
    cast(lock.lock_object_name as integer) as lockobjname,
    case cast(lock.lock_mode as smallint)
        when 3 then 'S'
        when 5 then 'X'
        when 8 then 'U'
        when 9 then 'NS'
        else substr(char(lock.lock_mode),1,4)
    end as mode,
    case lock.lock_status
        when 1 then 'G'
        when 2 then 'C'
        else cast(lock.lock_status as char)
    end as status,
    cast(lock.lock_escalation as integer) as escal
from    table(snapshot_lock('$DBNAME', $DPMODE )) as lock,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where lock.agent_id = appl_info.agent_id
    and lock.table_name = ucase('$TABNAME')
    and appl_info.appl_id <> '$MON_APPL_ID'
order by agentid, tbsname, tabname, type;
```

2 실행 결과는 다음과 같습니다.

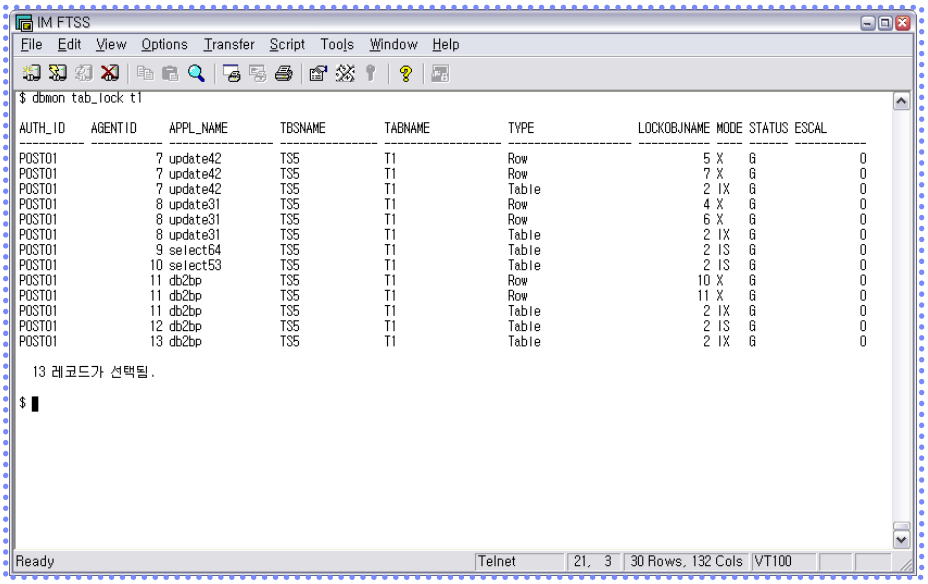


Figure 1411A ••• 테이블별 잠금

TOPIC 14-12 잠금 대기 에이전트

Point

잠금 대기 상태에 있는 에이전트의 목록을 확인합니다. 잠금 대기 응용프로그램명과 핸들 번호, 잠금 대상, 잠금 모드, 잠금을 보유하고 있는 응용프로그램명과 핸들 번호를 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

잠금에 대한 정확한 스냅샷 정보를 확인하려면, 시스템 기본 모니터 스위치인 DFT_MON_LOCK을 ON으로 설정하는 것이 좋습니다.

1 아래의 SQL문을 실행합니다.

```
select
    cast(lockwait.agent_id as integer) as wait_agent,
    substr(appl_info1.appl_name,1,10) as wait_appl,
    substr(lockwait.table_schema,1,18) as tabschema ,
    substr(lockwait.table_name,1,18) as tabname ,
    case lockwait.lock_object_type
        when 1 then 'Table'
        when 2 then 'Row'
        else substr(char(lockwait.lock_object_type),1,4)
    end as lock_type
    ,case cast(lockwait.lock_mode_requested  as smallint)
        when 3 then 'S'
        when 5 then 'X'
        else substr(char(lockwait.LOCK_MODE_REQUESTED),1,4)
    end as lock_mode
    ,cast(partition_number as smallint) partition
    ,cast(lockwait.agent_id_holding_lk as integer) as hold_agent
    ,substr(appl_info2.appl_name,1,10) as hold_appl
    ,timestampdiff( 2, char( current timestamp -
        lockwait.LOCK_WAIT_START_TIME)) as wait_time_s
from
    table(snapshot_lockwait('$DBNAME',$DPMODE)) as lockwait,
    table(snapshot_appl_info('$DBNAME',$DPMODE)) as appl_info1,
    table(snapshot_appl_info('$DBNAME',$DPMODE)) as appl_info2
where
    lockwait.agent_id = appl_info1.agent_id
    and   lockwait.agent_id_holding_lk = appl_info2.agent_id
order by wait_agent,lock_type, hold_agent;
```

2 실행 결과는 다음과 같습니다.

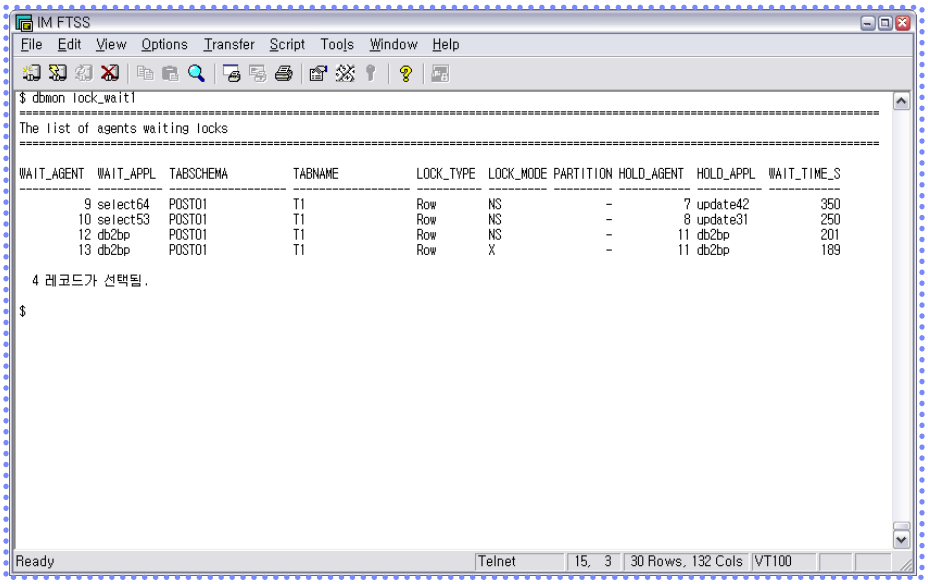


Figure 1412A ··· 잠금 대기 에이전트

TOPIC 14-13 잠금 대기 에이전트의 정적 SQL문

Point

잠금 대기 중인 에이전트가 실행할 정적 SQL문을 확인합니다. 해당 SQL문은 잠금을 보유한 응용프로그램이 UOW 를 종료할 때까지 실행되지 못하고 대기해야 합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select
    cast(lockwait.agent_id as integer) wait_agent
    ,substr(appl_info.appl_name,1,10) wait_appl
    ,substr(statement.creator,1,8) pkg_schema
    ,substr(statement.package_name,1,8) package
    ,cast(statement.section_number as smallint) section
    ,substr(cat_statements.text,1,63) SQL
    ,cast(lockwait.agent_id_holding_lk as integer) hold_agent
from    table(snapshot_lockwait('$DBNAME',$DPMODE)) lockwait
    ,table(snapshot_appl_info('$DBNAME',$DPMODE)) appl_info
    ,table(snapshot_statement('$DBNAME',$DPMODE)) statement
    ,syscat.statements cat_statements
where
    lockwait.agent_id = appl_info.agent_id                  and
    lockwait.agent_id = statement.agent_id                  and
    statement.stmt_text is null                             and
    cat_statements.pkgname = upper(statement.package_name)  and
    cat_statements.sectno = statement.section_number;
```

2 실행 결과는 다음과 같습니다.

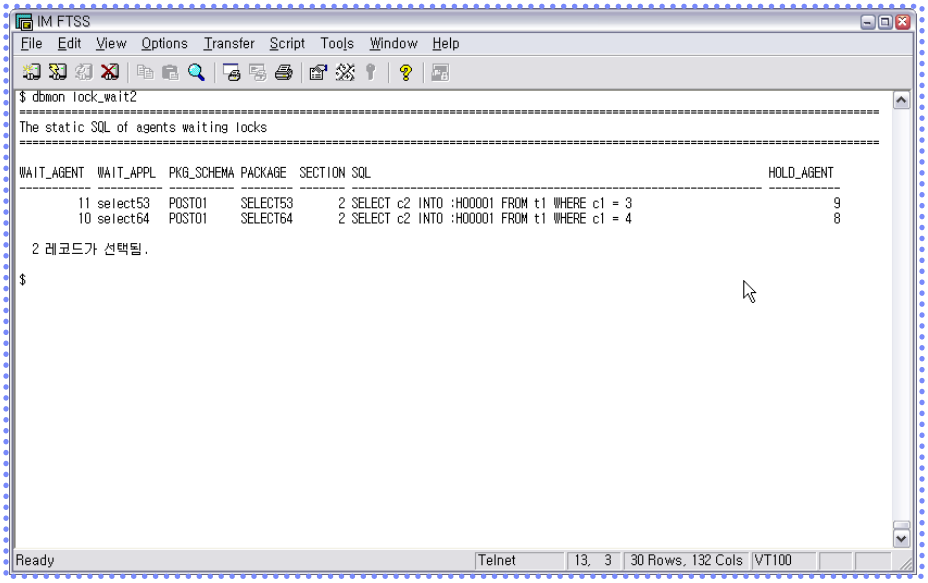


Figure 1413A ••• 잠금 대기 에이전트의 정적 SQL문

TOPIC 14-14 잠금 대기 에이전트의 동적 SQL문

Point

잠금 대기 중인 에이전트가 실행할 동적 SQL문을 확인합니다. 해당 SQL문은 잠금을 보유한 응용프로그램이 UOW 를 종료할 때까지 실행되지 못하고 대기해야 합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2 를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select
    cast(lockwait.agent_id as integer) wait_agent
    ,substr(appl_info.appl_name,1,10) wait_appl
    ,cast(NULL as char(8)) pkg_schema
    ,cast(NULL as char(8)) package
    ,cast(NULL as smallint) section
    ,substr(statement.stmt_text,1,63) SQL
    ,cast(lockwait.agent_id_holding_lk as integer) hold_agent
from    table(snapshot_lockwait('$DBNAME',$DPMODE)) lockwait
    ,table(snapshot_appl_info('$DBNAME',$DPMODE)) appl_info
    ,table(snapshot_statement('$DBNAME',$DPMODE)) statement
where
    lockwait.agent_id = appl_info.agent_id              and
    lockwait.agent_id = statement.agent_id              and
    statement.stmt_text is not null
order by wait_agent, hold_agent;
```

2 실행 결과는 다음과 같습니다.

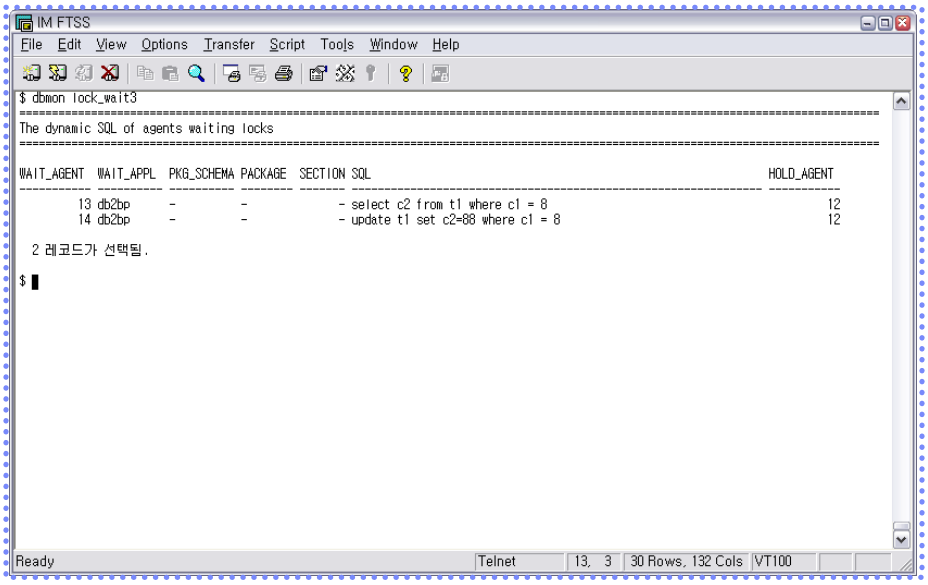


Figure 1414A ••• 잠금 대기 에이전트의 동적 SQL문

TOPIC 14-15 잠금 보유 에이전트의 정적 SQL문

Point

잠금을 보유하고 있는 에이전트가 실행한 정적 SQL 문을 확인합니다. 해당 SQL문의 실행이 완료되거나, UOW를 종료할 때까지 잠금을 보유합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select
    cast(lockwait.agent_id_holding_lk as integer) as hold_agent
    ,substr(appl_info.appl_name,1,10) as hold_appl
    ,substr(statement.creator,1,8) pkg_schema
    ,substr(statement.package_name,1,8) package
    ,cast(statement.section_number as smallint) last_section
    ,cast(cat_statements.sectno as smallint) section
    ,substr(cat_statements.text,1,61) last_SQL
from
    table(snapshot_lockwait('$DBNAME',$DPMODE)) as lockwait
    ,table(snapshot_appl_info('$DBNAME',$DPMODE)) as appl_info
    ,table(snapshot_statement('$DBNAME',$DPMODE)) as statement
    ,syscat.statements cat_statements
where
    lockwait.agent_id_holding_lk = appl_info.agent_id       and
    lockwait.agent_id_holding_lk = statement.agent_id       and
    cat_statements.pkgschema = upper(statement.creator)     and
    cat_statements.pkgname = upper(statement.package_name)  and
    cat_statements.sectno <= statement.section_number
order by hold_agent, section;
```

2 실행 결과는 다음과 같습니다.

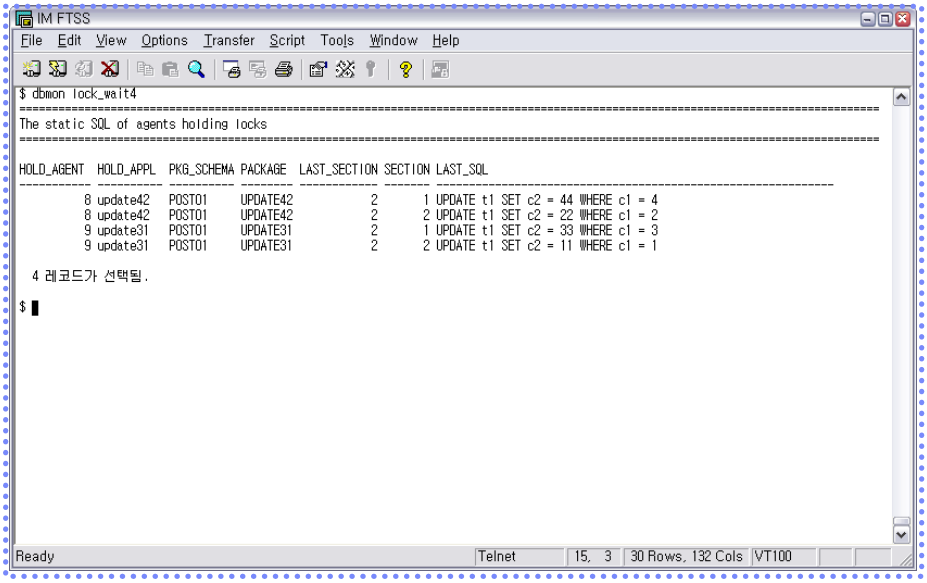


Figure 1415A • • • 잠금 보유 에이전트의 정적 SQL문

TOPIC 14-16 잠금 보유 에이전트의 동적 SQL문

Point

잠금을 보유하고 있는 에이전트가 실행한 마지막 SQL문을 확인합니다. 해당 SQL문의 실행이 완료되거나, UOW를 종료할 때까지 잠금을 보유합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select  distinct
    cast(lockwait.agent_id_holding_lk as integer) as hold_agent
    ,substr(appl_info.appl_name,1,10) as hold_appl
    ,cast(NULL as char(8)) pkg_schema
    ,cast(NULL as char(8)) package
    ,cast(NULL as smallint) last_section
    ,cast(NULL as smallint) section
    ,cast(substr(statement.stmt_text,1,61) as char(61)) SQL
from
    table(snapshot_lockwait('$DBNAME',$DPMODE)) as lockwait
    ,table(snapshot_appl_info('$DBNAME',$DPMODE)) as appl_info
    ,table(snapshot_statement('$DBNAME',$DPMODE)) as statement
where
    lockwait.agent_id_holding_lk = appl_info.agent_id       and
    lockwait.agent_id_holding_lk = statement.agent_id       and
    statement.stmt_text is not null
order by hold_agent, last_section;
```

2 실행 결과는 다음과 같습니다.

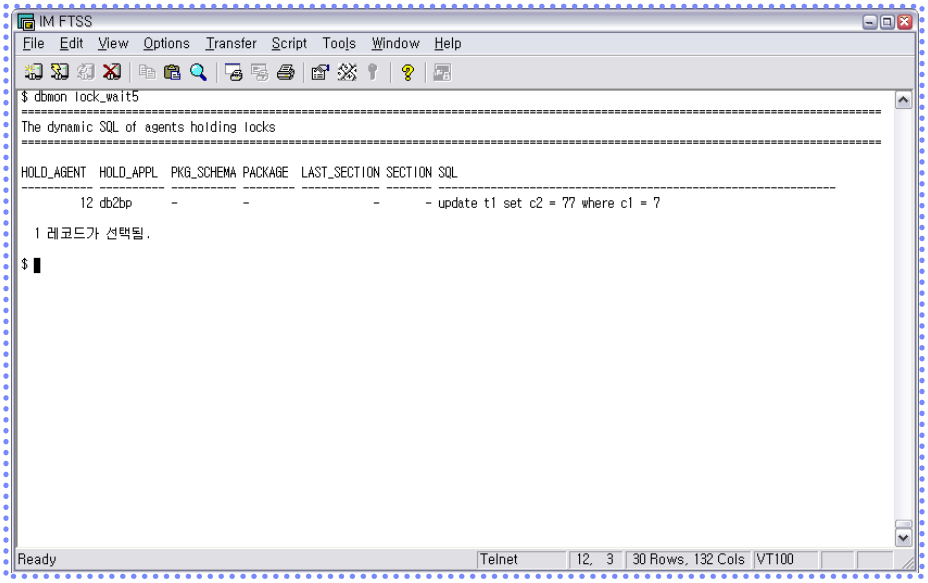


Figure 1416A ··· 잠금 보유 에이전트의 동적 SQL문

TOPIC 14-17 응용프로그램별 로그 사용량

Point

특정 응용프로그램이 사용하고 있는 로그의 사용량에 대한 정보를 확인합니다. 로그 사용량이 가장 많은 응용프로그램을 파악할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$MON_APPL_ID는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.auth_id,1,8) as authid
    ,cast(appl.agent_id as integer) as agentid
    ,substr(appl_info.appl_name,1,20) appl_name
    ,appl.uow_log_space_used
    ,cast(appl.agent_usr_cpu_time_s as integer) user_cpu
    ,timestampdiff( 2, char( current timestamp - appl.uow_start_time ))
        as elapsed_time
    ,appl.rows_read
    ,appl.rows_written
from table(snapshot_appl('$DBNAME', $DPMODE )) as appl,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where appl.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
order by uow_log_space_used desc, rows_written desc, agentid;
```

2 실행 결과는 다음과 같습니다.

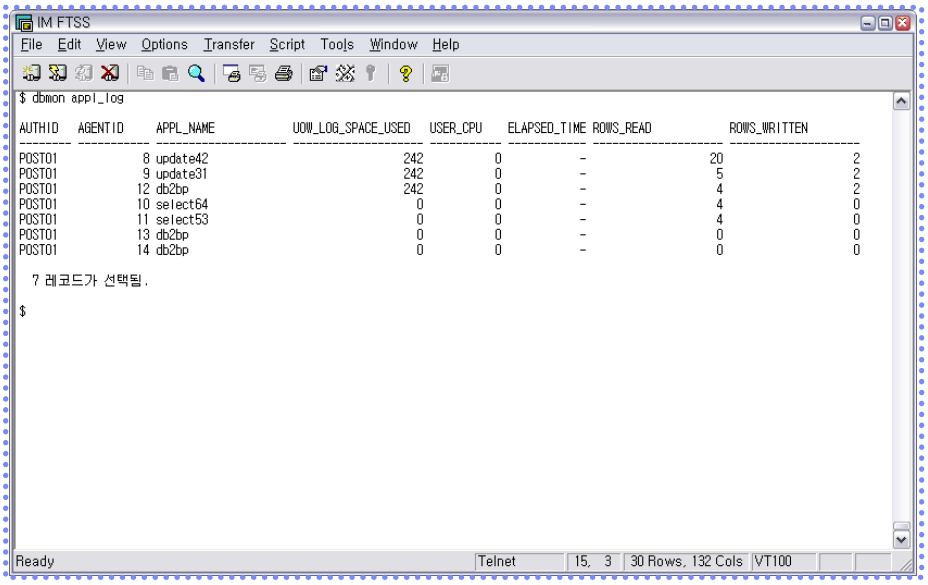


Figure 1417A ••• 응용프로그램별 로그 사용량

TOPIC 14-18 데이터베이스별 로그 사용량

Point

한 데이터베이스 전체 로그 사용량과 가장 오래된 트랜잭션을 가진 에이전트 ID 를 확인할 수 있습니다. 데이터베이스의 로그 사용량 비율이 심하게 높은 경우에는 가장 오래된 트랜잭션에 대한 점검이 필요합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2 를 사용합니다.

Tip

$MON_APPL_ID 는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID 를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select
    total_log_used
    ,total_log_available
    ,tot_log_used_top
    ,cast(sec_logs_allocated as integer) sec_logs_allocated
    ,sec_log_used_top
    ,cast(appl_id_oldest_xact as integer) oldest_tx_agent
    ,substr(appl_info.appl_name,1,12) appl_name
from table(snapshot_database('$DBNAME', $DPMODE )) as database,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where database.appl_id_oldest_xact = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID';
```

2 실행 결과는 다음과 같습니다.

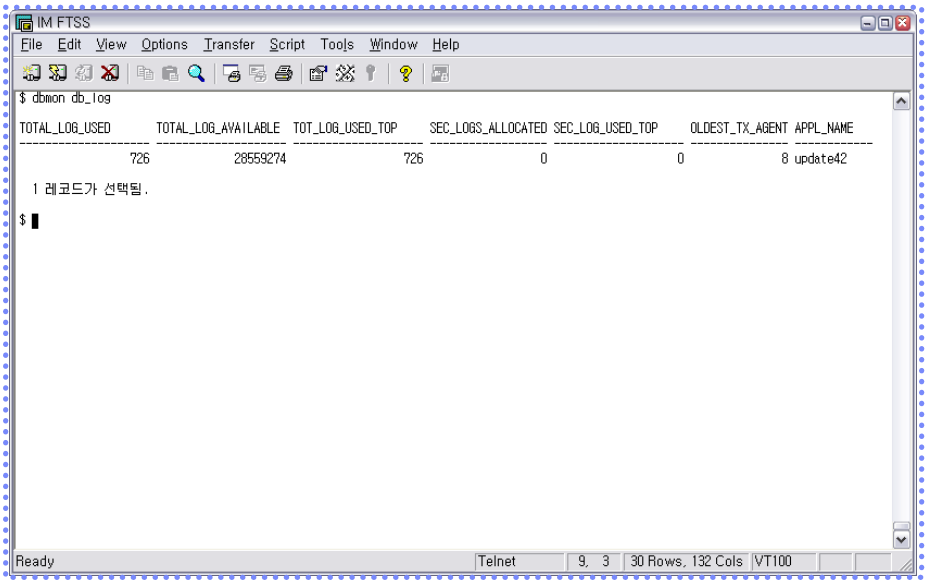


Figure 1418A ••• 데이터베이스별 로그 사용량

TOPIC 14-19 테이블 스페이스 사용량

Point

테이블 스페이스의 사용량과 사용율을 확인합니다. SMS 테이블스페이스는 필요할 때마다 공간을 할당하므로 거의 100%의 사용 비율을 보입니다. DMS 테이블스페이스의 경우에 70 ~ 80% 이상을 사용했다면, 컨테이너를 확장하는 등의 관리가 필요합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(tbs_cfg.tablespace_name,1,20) tablespace
    ,cast(tbs_cfg.tablespace_state as smallint) as state
    ,case tbs_cfg.tablespace_type
        when 1 then 'SMS'
        else 'DMS'
    end type
    ,cast(page_size as integer) page_size
    ,(total_pages * page_size) / 1024 / 1024 as total_size_mb
    ,(used_pages * page_size) / 1024 / 1024 as used_size_mb
    ,(free_pages * page_size) / 1024 / 1024 as free_size_mb
    ,case tablespace_type
        when 1 then 100
        else dec((tbs_cfg.used_pages * 100.00)/tbs_cfg.total_pages,5,2)
    end  as ratio
from table(snapshot_tbs_cfg('$DBNAME',$DPMODE)) as tbs_cfg
order by tablespace;
```

2 실행 결과는 다음과 같습니다.

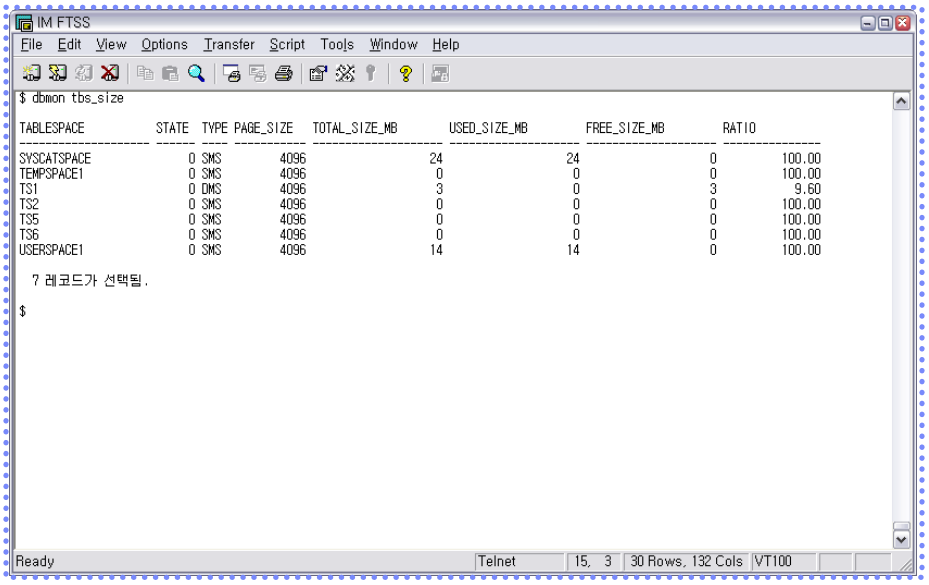


Figure 1419A ••• 테이블 스페이스 사용량

TOPIC 14-20 테이블 스페이스 컨테이너 사용량

Point

테이블스페이스의 각 컨테이너별 사용량을 확인합니다. 데이터는 기본적으로 라운드 로빈 방식으로 각 컨테이너에 균등하게 저장되므로, 한 테이블스페이스에 속한 각 컨테이너는 유사한 사용 비율을 보이게 됩니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select  distinct substr(container.tablespace_name,1,20) as tablespace
    ,substr(container.container_name,1,50) as container_name
    ,cast(container.total_pages as integer) as total_pages
    ,cast(container.usable_pages as integer) as usable_pages
    ,cast(container.total_pages*tbs_cfg.page_size/1024/1024 as int)
         as total_size_M
    ,cast(container.usable_pages*tbs_cfg.page_size/1024/1024 as int)
         as usable_pages_M
    ,case container.accessible
        when 1 then 'YES'
        when 0 then 'NO'
        else cast(container.accessible as char(1))
    end access
from    table(snapshot_container('$DBNAME', $DPMODE)) as container,
    table(snapshot_tbs_cfg('$DBNAME', $DPMODE)) as tbs_cfg
where container.tablespace_id = tbs_cfg.tablespace_id
order by tablespace, container_name;
```

2 실행 결과는 다음과 같습니다.

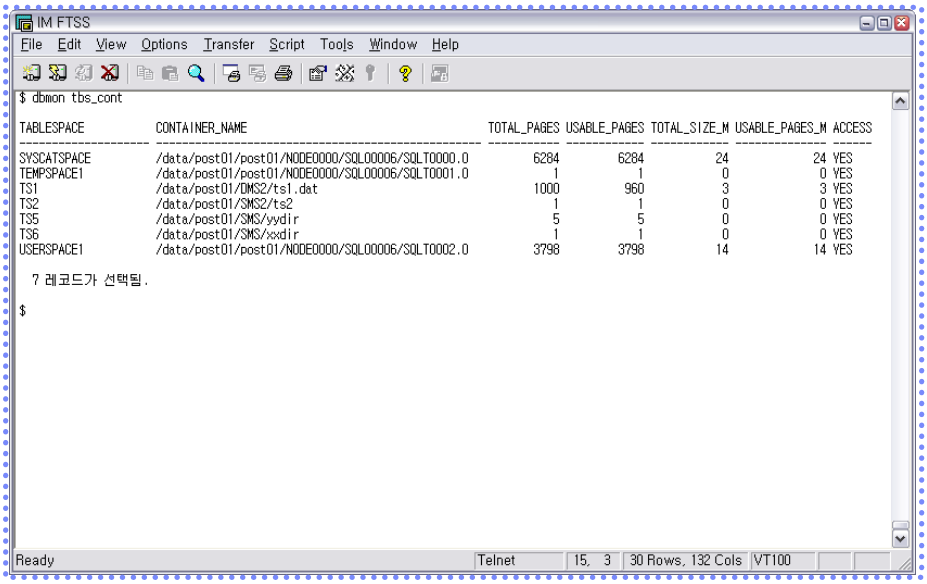


Figure 1420A ••• 테이블 스페이스 사용량

TOPIC 14-21 테이블 스페이스 적중률

Point

테이블스페이스의 데이터가 버퍼풀에 있는 비율을 확인합니다. 테이블의 데이터와 인덱스의 데이터에 대한 버퍼 적중율이 높을수록 유리합니다. 특정 테이블에 대한 적중률이 심하게 낮은 경우에는 버퍼풀의 크기를 조절하거나, 별도의 절 할당하도록 합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(tbs.tablespace_name,1,20) as tablespace
    ,cast(tbs.pool_data_l_reads as integer)  pool_data_l_reads
    ,cast(tbs.pool_data_p_reads as integer) pool_data_p_reads
    ,case tbs.pool_data_l_reads
        when 0 then null
        else dec(((tbs.pool_data_l_reads - tbs.pool_data_p_reads)
             * 100.00 / tbs.pool_data_l_reads) ,5,2)
    end data_hit_ratio
    ,cast(tbs.pool_index_l_reads as integer) pool_index_l_reads
    ,cast(tbs.pool_index_p_reads as integer) pool_index_p_reads
    ,case tbs.pool_index_l_reads
        when 0 then null
        else    dec(((tbs.pool_index_l_reads - tbs.pool_index_p_reads)
             * 100.00 / tbs.pool_index_l_reads),5,2)
    end index_hit_ratio
from table(snapshot_tbs('$DBNAME', $DPMODE)) as tbs
order by tablespace;
```

2 실행 결과는 다음과 같습니다.

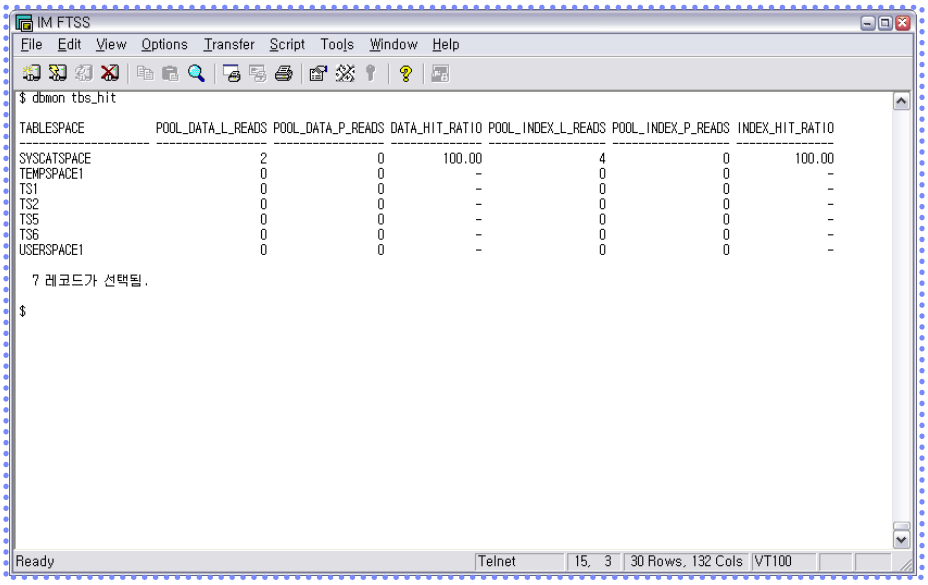


Figure 1421A ••• 테이블 스페이스 적중률

TOPIC 14-22 이벤트 모니터

Point

일정 기간 동안의 데이터베이스의 활동 내역에 대한 모니터링 정보를 수집하여 파일 또는 테이블에 저장하는 도구입니다. CREATE EVENT MONITOR 명령어, SET EVENT MONITOR 명령어, DROP EVENT MONITOR 명령어를 이용하여 관리합니다.

Tip

CONNECTIONS,STATEMENTS, TRANSACTIONS 유형은 수집되는 정보의 양이 너무 많게 되므로 모니터링의 범위를 줄일 수 있는 조건을 지정합니다.

Tip

테이블 이벤트 모니터를 사용할 때는 별도의 테이블스페이스를 생성하는 것이 좋습니다.

1. 특정한 이벤트가 발생할 때마다 이벤트 유형별로 관련된 이벤트 정보를 수집합니다. 이벤트의 유형은 다음과 같이 6가지로 분류됩니다.

이벤트유형	설명
DATABASE	마지막 응용프로그램이 접속을 종료할 때, 데이터베이스에 관련된 모든 모니터링 항목을 기록합니다.
TABLES	마지막 응용프로그램이 접속을 종료할 때, 변경이 발생했던 테이블에 대해 읽혀진 행의 수와 기록된 행의 수 등을 기록합니다.
TABLESPACES	마지막 응용프로그램이 접속을 종료할 때, 각각의 테이블스페이스에 대해 버퍼풀 모니터링 항목과 프리페처, 페이지 클리너, 직접 I/O 회수 등의 정보를 기록합니다.
BUFFERPOOLS	마지막 응용프로그램이 접속을 종료할 때, 각각의 버퍼풀에 대한 기본 모니터링 항목과 프리페처, 페이지 클리너, 직접 I/O 회수 등의 정보를 기록합니다.
CONNECTIONS	응용프로그램이 접속을 종료할 때, 응용프로그램과 관련된 모든 모니터링 항목을 기록합니다.
STATEMENTS	SQL문이 실행을 종료할 때마다 SQL문의 시작과 종료 시간, 사용한 CPU, 동적 SQL문의 텍스트 등을 정보를 기록합니다.
DEADLOCKS	DB2 9.7부터 LOCKING 이벤트 모니터 사용을 권장합니다,
TRANSACTIONS	DB2 9.7부터 UOW 이벤트 모니텅 사용을 권장합니다.

2. 이벤트 모니터가 수집한 정보는 파일, 테이블, 파이프 등에 저장됩니다. 저장 방식에 따라 수집된 정보를 분석하는 방법이 다릅니다.

3. 특정 이벤트가 발생할 때마다 관련된 모니터링 정보를 수집하려면 데이터베이스에 접속한 후에 create event monitor 명령어를 이용하여 이벤트 모니터를 생성합니다.

4. set event monitor 명령어는 원하는 시점에 이벤트 모니터의 상태값을 ON 또는 OFF 로 설정하여 이벤트 모니터의 모니터링 기간을 조절할 수 있습니다. 상태값이 ON 이면 이벤트 모니터는 활성화되어 이벤트의 발생 정보를 수집합니다. 상태값이 OFF 이면 이벤트가 발생해도 이벤트 모니터는 정보를 수집하지 않습니다.

5. 이벤트 모니터를 제거하려면 drop event monitor 명령어를 이용합니다. 상태값인 ON 으로 설정된 이벤트 모니터는 제거할 수 없습니다.

6. 이벤트 모니터에 대한 정보는 SYSCAT.EVENTMONITORS 뷰를 이용해서 확인합니다.

```
$ db2 "select * from syscat.eventmonitors"
```

TOPIC 14-22 이벤트 모니터

Point

Deadlock 및 Transaction에 대한 이벤트 모니터가 9.7 이후 Locking 및 UOW로 기능 대체됩니다.

7 Deadlock 모니터링 방법입니다.

9.7 이전)
CREATE EVENT MONITOR FOR DEADLOCKS
CREATE EVENT MONITOR FOR DB2DETAILDEADLOCK
9.7 이후)
CREATE EVENT MONITOR FOR LOCKING

8 트랜잭션 모니터링 방법입니다.

9.7 이전)
CREATE EVENT MONITOR FOR TRANSACTIONS
9.7 이후)
CREATE EVENT MONITOR FOR UNIT OF WORK

TOPIC 14.23 CREATE EVENT MONITOR 문

Point

특정한 데이터베이스에 이벤트 모니터를 생성하는 SQL문입니다. 8가지 이벤트 유형을 선택적으로 지정할 수 있고, 수집된 정보를 저장할 파일, 테이블, 파이프의 정보를 지정합니다.

Tip

SYSADM 또는 DBADM 권한이 필요합니다.

1 create event monitor 의 형식은 다음과 같습니다.

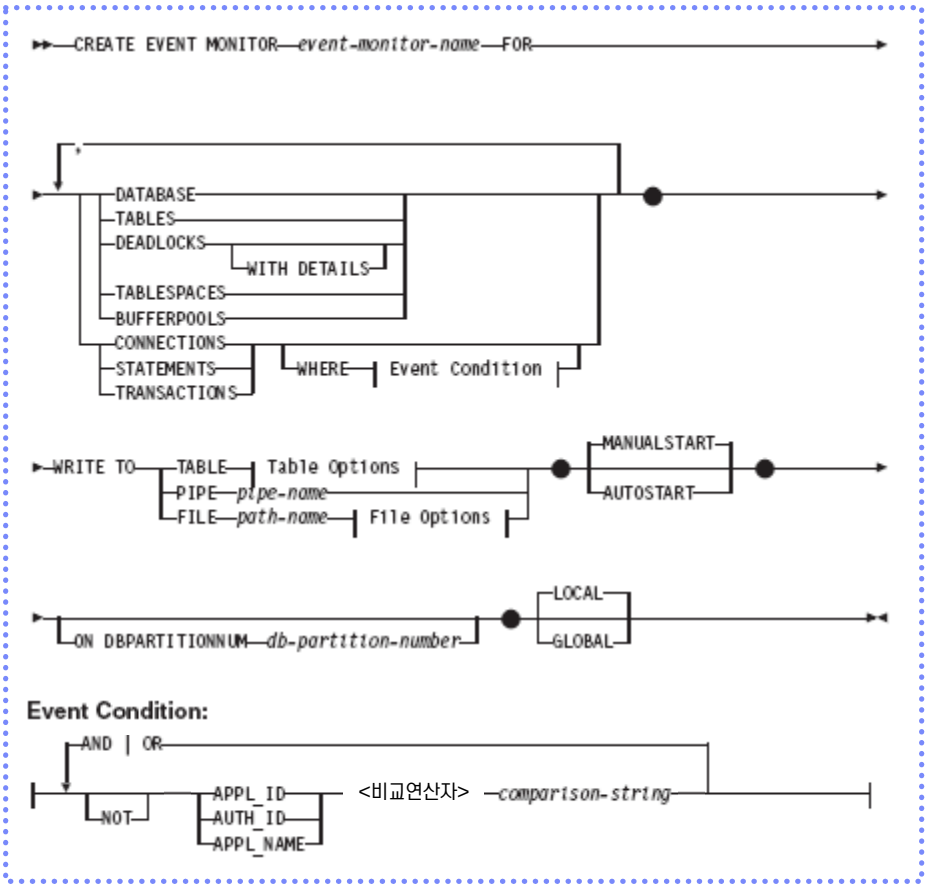


Figure 1423A CREATE EVENT MONITOR 문

2 옵션에 대한 설명은 다음과 같습니다.

옵 션	설 명
<이벤트모니터명>	임의의 고유한 값으로 지정합니다.
FOR <이벤트유형>	8가지 유형 중에서 한 개 이상을 선택합니다.
WHERE	CONNECTIONS, STATEMENTS, TRANSACTIONS 유형은 수집되는 정보의 양이 너무 많게 되므로 모니터링의 범위를 줄일 수 있는 조건을 지정합니다. APPL_ID, AUTH_ID, APPL_NAME 을 이용하여 조건식을 표현합니다.
WRITE TO TABLE	수집된 정보를 테이블에 저장합니다.
WRITE TO FILE	수집된 정보를 파일에 저장합니다.
WRITE TO PIPE	수집된 정보를 파이프에 저장합니다.
MANUALSTART	데이터베이스가 활성화되어도 자동적으로 시작되지 않습니다. SET EVENT MONITOR STATE 문으로 조절합니다.
AUTOSTART	데이터베이스가 활성화되면 자동적으로 시작됩니다.

TOPIC 14-24 파일 이벤트 모니터

Point

이벤트 모니터가 수집한 정보를 파일에 저장합니다. 파일이 생성될 디렉토리, 파일의 크기, 파일의 개수, 버퍼의 크기 등의 옵션을 추가적으로 지정할 수 있습니다. 생성된 파일은 db2evmon 명령어로 변환하여야 합니다.

1 create event monitor 의 형식은 다음과 같습니다.

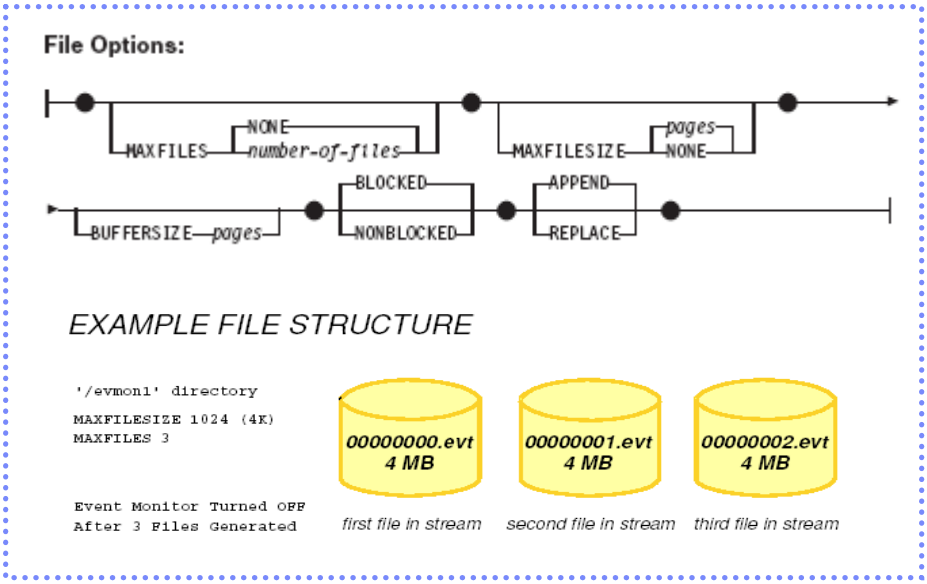


Figure 1424A • • • 파일 이벤트 모니터의 옵션

Tip

수집된 정보는 출력용 디렉토리에 *.evt 파일에 저장됩니다.

2 이벤트 모니터가 수집하는 정보를 저장할 출력용 디렉토리를 생성합니다.

```
$ mkdir –p <출력용 디렉토리>
```

Tip

출력 파일의 개수가 MAXFILES 옵션의 값을 초과하거나, 데이터베이스가 비활성화되면 이벤트 모니터는 자동적으로 중지됩니다.

3 create event monitor 명령어로 파일 이벤트 모니터를 생성합니다.

```
$ db2 “create event monitor <이벤트모니터명> for <이벤트유형명> write to
    file <출력용 디렉토리명> MAXFILES <최대 파일 개수> MANUALSTART”
```

4 이벤트 모니터를 활성화시킵니다.

```
$ db2 “set event monitor <이벤트모니터명> state = 1”
```

5 일정한 시간동안 이벤트 정보를 수집한 후에 이벤트 모니터를 비활성화시킵니다.

```
$ db2 “set event monitor <이벤트모니터명> state = 0”
```

Tip

파일 이벤트 모니터의 출력 파일은 db2evmon 명령어로 분석합니다. 이벤트 모니터가 생성한 출력 파일인 *.evt 는 제거해도 됩니다.

6 db2evmon 명령어를 이용하여 파일로 수집된 이벤트 정보를 텍스트 형태로 변환합니다.

```
$ db2evmon -db <데이터베이스명> -evm <이벤트모니터명> > <출력파일명>
$ rm -Rf  <출력용 디렉토리>/*.evt
```

7 불필요한 이벤트 모니터는 제거합니다.

```
$ db2 drop event monitor <이벤트모니터명>
```

TOPIC 14-25 테이블 이벤트 모니터

Point

이벤트 모니터의 결과를 테이블에 저장합니다. 수집된 데이터 스트림을 한 개 이상의 논리적인 그룹으로 분할하여 해당 테이블에 INSERT 합니다. SELECT 문을 이용하여 분석합니다.

Tip

테이블 이벤트 모니터를 사용할 때는 별도의 테이블스페이스를 생성하는 것이 좋습니다.

Tip

테이블 이벤트 모니터의 정보는 테이블에 저장되므로, 일반 테이블과 동일한 방법으로 조회하면 됩니다.

Tip

pctdeactivate 옵션은 DMS 테이블스페이스에만 적용되며, 지정한 비율 이상을 사용하게 되면 이벤트 모니터를 중지합니다.

1 옵션은 다음과 같습니다.

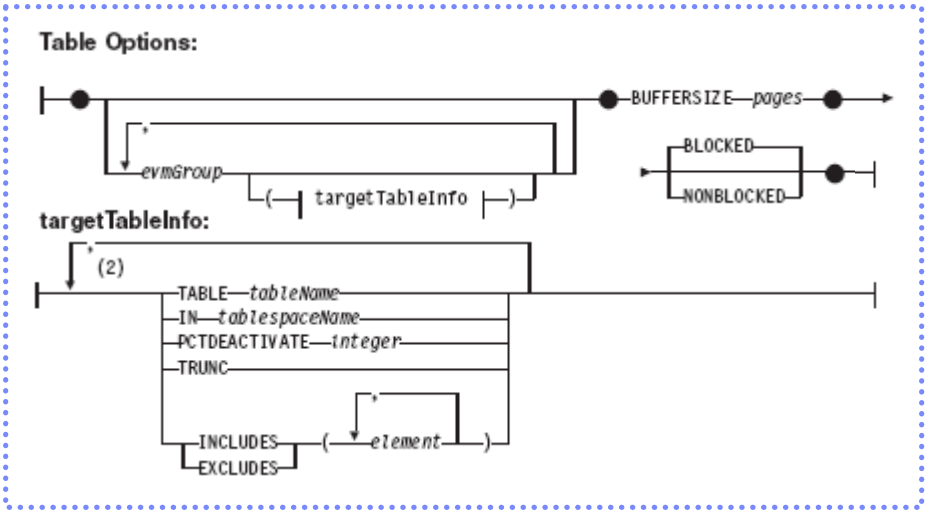


Figure 1425A ••• 파일 이벤트 모니터의 옵션

2 이벤트 모니터가 수집할 정보를 저장할 테이블스페이스를 생성합니다.

```
db2 "create tablespace <TS명> managed by database using (...)"
```

3 이벤트 모니터를 생성합니다.

```
db2 +p -t << EOF
    create event monitor <이벤트모니터명> for <이벤트유형>
    write to table
    connheader (table mon2.connheader, in <TS명>,
                pctdeactivate 90),
    stmt (table mon2.statements,  in <TS명>,  pctdeactivate 90),
     control (table mon2.control, in <TS명>, pctdeactivate 90)
    manualstart;
EOF
```

4 이벤트 모니터를 활성화시킵니다. 일정한 시간 동안 이벤트 정보를 수집한 후에 이벤트 모니터를 비활성화시킵니다.

```
$ db2 “set event monitor <이벤트모니터명> state = 1”
$ db2 “set event monitor <이벤트모니터명> state = 0”
```

5 불필요한 이벤트 모니터는 제거합니다.

```
$ db2 drop event monitor <이벤트모니터명>
```

TOPIC 14-26 시간 소요 모니터

Point

포괄적인 시간 기반 모니터 요소 세트를 사용하면 어디에, 어떻게 시간이 사용되었는지를 쉽게 이해할 수 있습니다.

Tip

- 정밀한 성능 튜닝을 위해 9.7에서 시간 기반 모니터링이 개선되었습니다.

1. 포괄적인 시간 소요 모니터링 요소를 통하여, 시간이 어디에 소요되는지를 정확하게 표시하는 기능으로 문제점의 잠재적인 원인을 쉽게 찾고 성능 개선을 위해 조정을 수행할 수 있을지 여부를 확인할 수 있습니다.

전체 시간 소요 처리 요청 및 DB2 데이터베이스 관리 프로그램 내의 전체 대기 시간
자원(예: 잠금, 버퍼 풀 또는 로깅)에 의한 대기 시간
DB2 데이터베이스 관리 프로그램 외부의 시간 소요 측정(client_idle_wait_time)

2. 시간 모니터링 결과값을 수집하여 아래와 같은 도표를 산출할 수 있습니다. 아래 예제는 전체 대기 시간 중 Lock 대기 시간이 가장 큰 부분을 차지함을 알 수 있습니다.

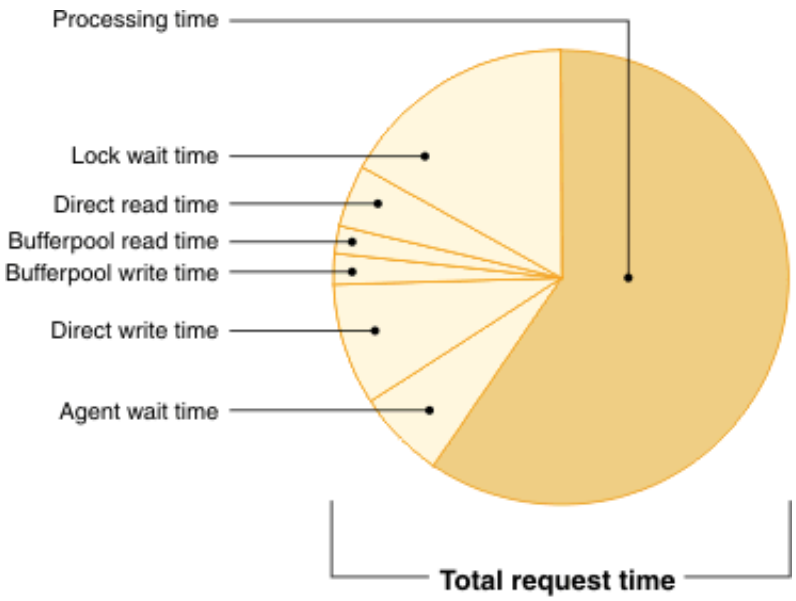


Figure 1426A ••• 시스템 전체 요청시간 예제

TOPIC 14-27 DB2PD 모니터링

Point

DB2PD는 메모리로부터 수집되어 스냅샷 모니터에 비해 보다 가볍고 신속한 데이터베이스 모니터을 제공합니다.

Tip

Db2pd는 메모리 세트의 빠르고 즉각적인 정보를 포함하는 문제점 판별 도구입니다. 반면에 스냅샷 모니터는 래치를 확보하거나 엔진자원을 사용합니다.

1 db2pd 로 데이터베이스 시스템 메모리 세트에서 정보를 검색합니다.

Figure 1427A ··· DB2PD 사용법

옵션	설명
-inst	모든 인스턴스 레벨의 정보를 보여줍니다.
-h | help	온라인 도움말을 보여줍니다.
-v | -version	설치된 DB2제품의 현재버전과 서비스레벨을 보여줍니다.
-database | -db<DB명>	지정된 데이타베이스의 메모리집합에 접근하도록 지정합니다.
-alldatabases | -alldbsw	모든 데이타베이스 메모리 집합에 접근하도록 지정합니다.
-everything	모든 데이타베이스의 모든 옵션을 수행합니다.
-file <파일명>	결과값을 지정된 파일에 기록합니다.
-applications	애플리케이션에 대한 정보를 보여줍니다. 애플리케이션 ID가 지정될 경우 그 애플리케이션 정보만을 보여줍니다.
-appinfo	현재 UOW의 Dynamic SQL 구문의 실행을 포함하여 애플리케이션에 대한 자세한 정보를 제공합니다.

Figure 1427B ··· DB2PD 옵션 설명

TOPIC 14-28 db2top

Point

db2top은 db2 데이터베이스의 포괄적인 모니터링을 위해 기본으로 제공되는 도구입니다.

Tip

db2top은 별도 제공되었으나, 9.7 이후 설치 시 기본 제공됩니다.

Tip

구체적인 사용 설명은 http://www.ibm.com/developerworks/data/library/techarticle/dm-0812wang/을 참조하십시오.

1 인스턴스 사용자로 로그인 하여 db2top 을 입력합니다.

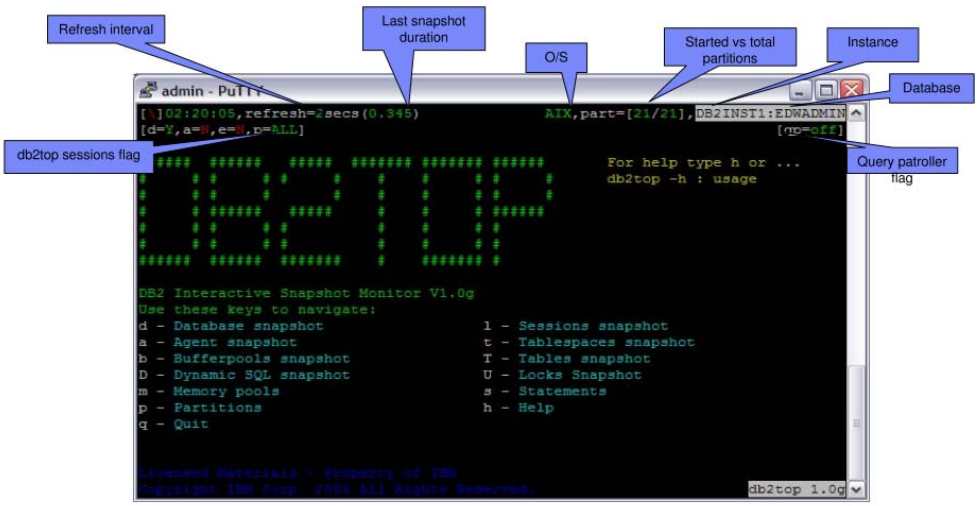


Figure 1428A ••• db2top 초기화면

2 db2top의 옵션은 아래와 같습니다. db2top의 어떠한 화면에서도 h를 누르게 되면 도움말이 나타납니다.

d - Database	l - Sessions
a - Details for agent <agentid>	t - Tablespaces
b - Bufferpools	
T - Tables	
D - Dynamic SQL	U - Locks
m - Memory pools	s - Statements
u - Utilities	p - Partitions
C - Toggle collector on/off	W - Watch user/agent
/ - Set regexp	
g - Toggle graph on/off	
i - Toggle idle objects on/off	G - Toggle local/global sn
P - Select db partition	X - Toggle extended mod
k - Toggle actual/delta values	z - Descending sort
Z - Ascending sort	+ - Longer default sort
- - Shorter default sort	I - Set new snapshot inter
R - Reset snapshot monitor	S - Run native DB2 snap
> - Move right	< - Move left
c - Change columns order	f - Freeze display
! - Goto to system prompt	V - Set default explain sc
O - Display settings	w - Write parms to .db2t
h - Help	q - Quit

TOPIC 14-29 db2top - Application

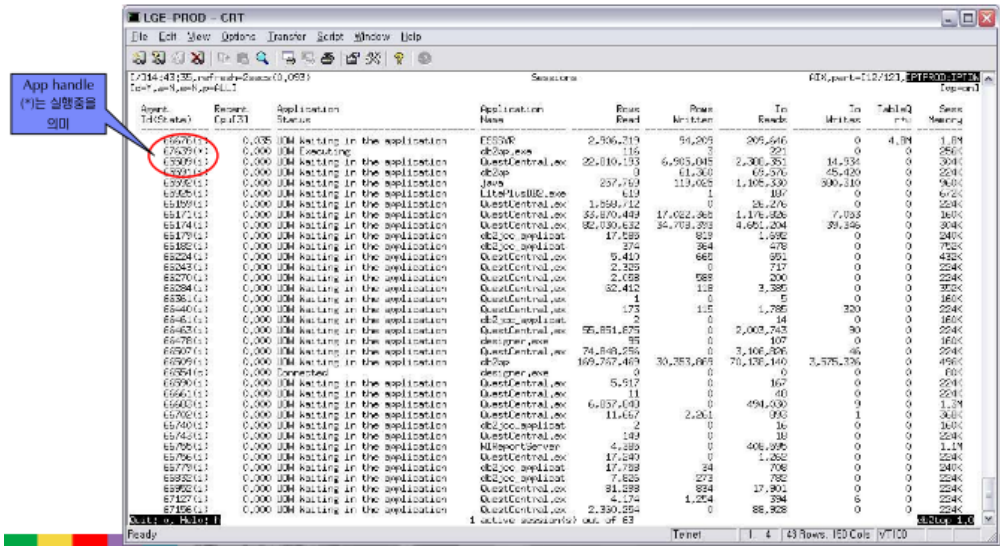
Point

db2top을 이용한 응용프로그램 모니터링 수행 방법입니다.

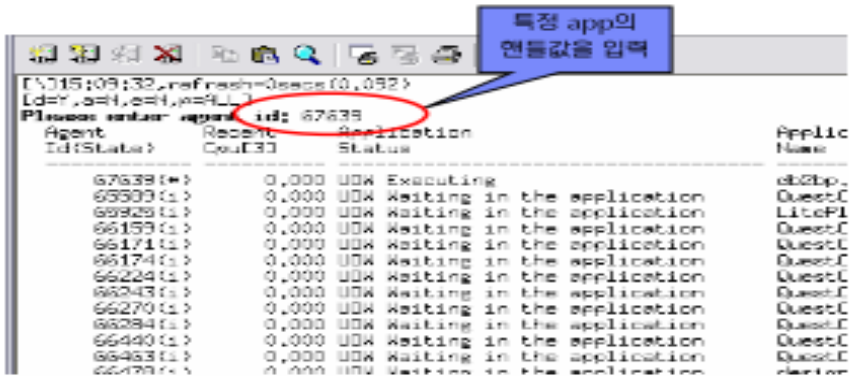
Tip

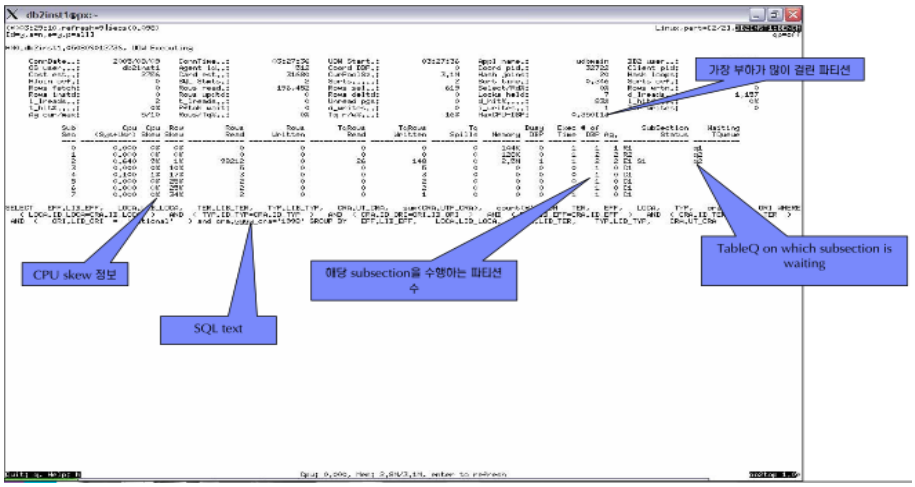
- f 키를 사용하면 특정 Appliction을 작업 중지시킬 수 있습니다.

1 l 키를 사용하여 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.



2 a 키를 사용하여 특정 application의 핸들값을 입력하면 자세한 정보를 확인 할 수 있습니다.





TOPIC 14-29 db2top - Application

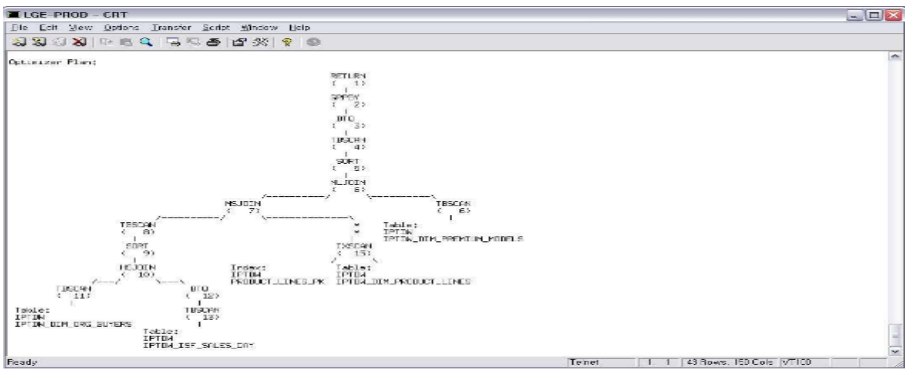
Point

db2top을 이용한 세부 응용프로그램 모니터링 수행 방법입니다.

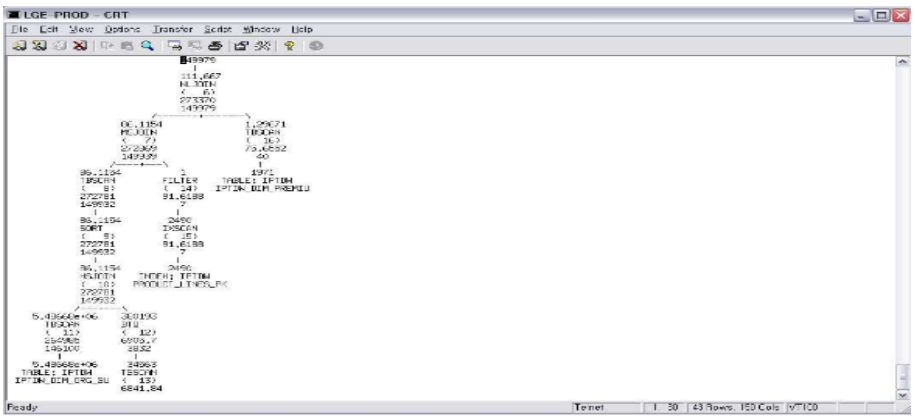
Tip

- vi editor를 통해 처리되기
- 때문에 :q를 통해 완료됩니다.

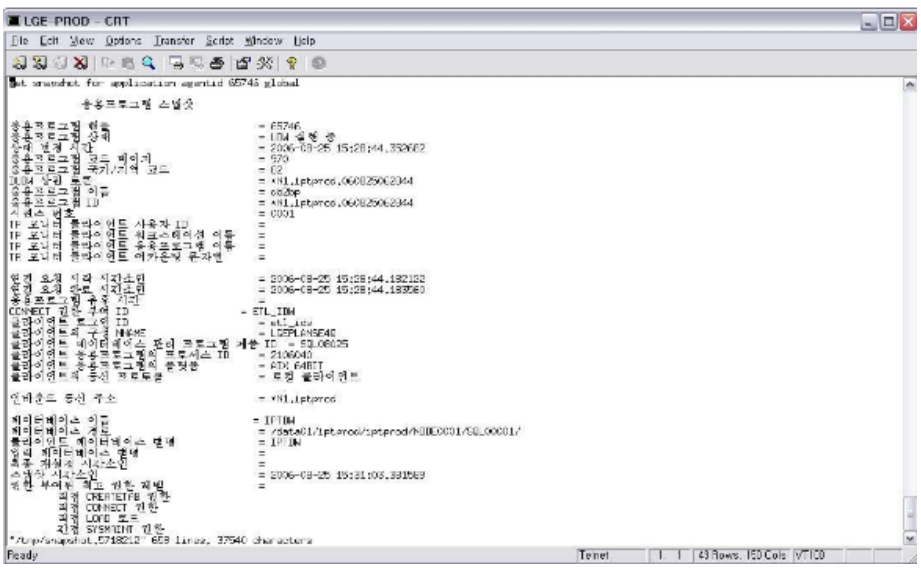
3 e 키를 사용하면 db2expln을 사용하여 해당 app의 access plan을 검증할 수 있습니다. vi editor를 통해 처리되기 때문에 :q를 통해 완료됩니다.



4 x 키를 사용하면 db2exfmt를 사용하여 해당 app의 access plan을 검증할 수 있습니다.



5 s 키는 해당 app에 대한 db2 snapshot을 수행할 수 있습니다. 이를 통해 보다 자세한 데이터를 수집하고 모니터링 할 수 있습니다.



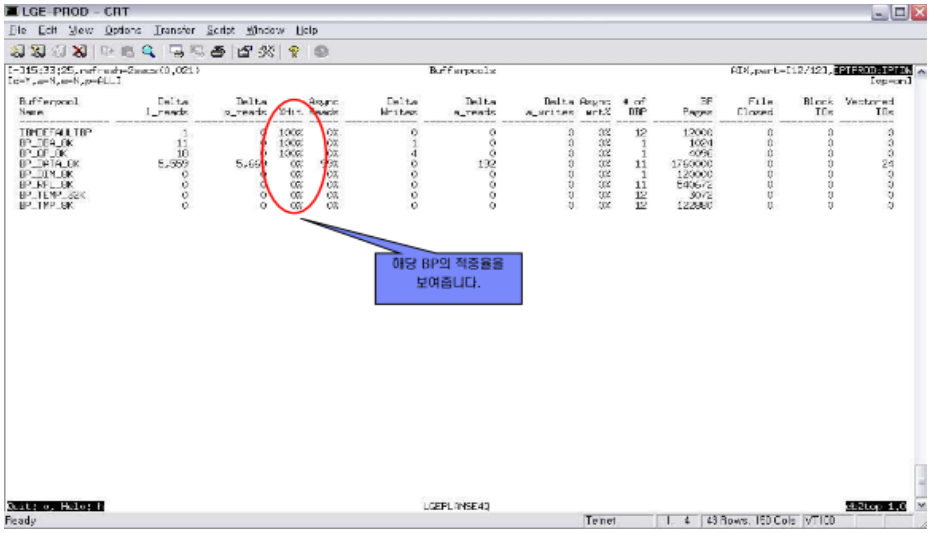
TOPIC 14-30 db2top - Memory

Point

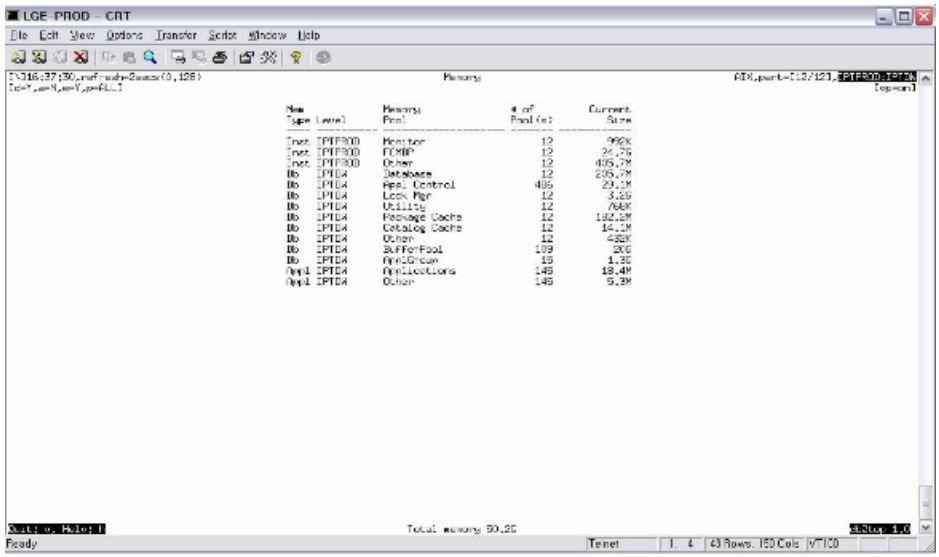
db2top을 이용한 메모리 모니터링 수행 방법입니다.

1 b 키를 사용하여 bufferpool을 모니터링 할 수 있습니다.

좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.



2 m 키를 사용하여 DB2가 사용하는 전체 메모리를 모니터링 할 수 있습니다.

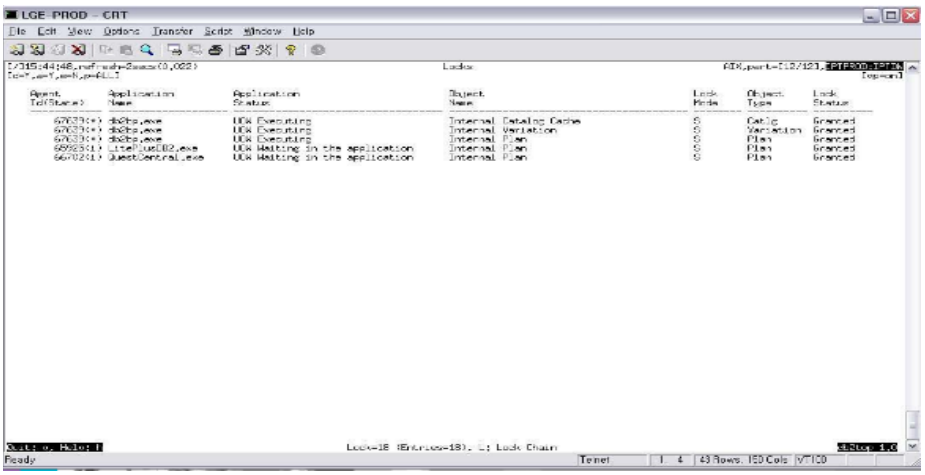


TOPIC 14-31 db2top – Lock

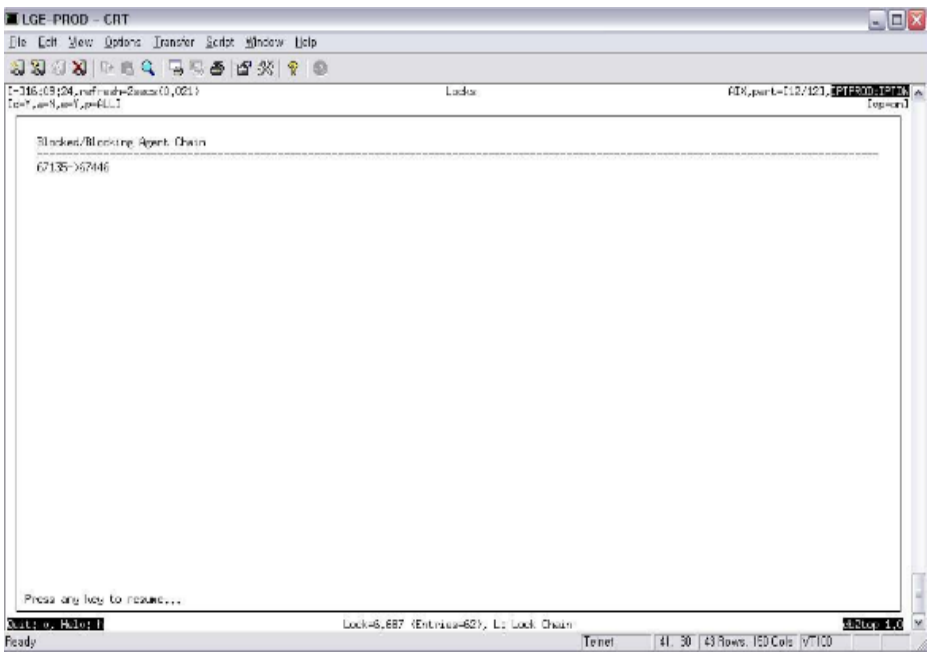
Point

db2top을 이용한 Lock 모니터링 수행 방법입니다.

1 U 키를 사용하여 lock을 모니터링 할 수 있습니다. 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다



2 L 키를 사용하여 lock chain을 확인할 수 있습니다.



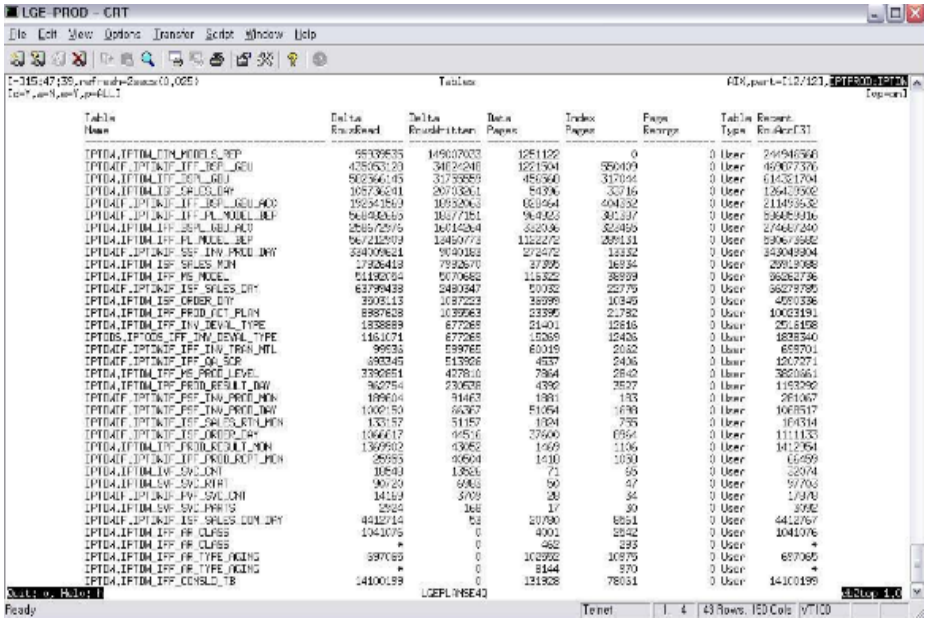
TOPIC 14-32 db2top – Table

Point

db2top을 이용한 테이블 모니터링 수행 방법입니다.

1 T 키를 사용하여 테이블을 모니터링 할 수 있습니다.

좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.

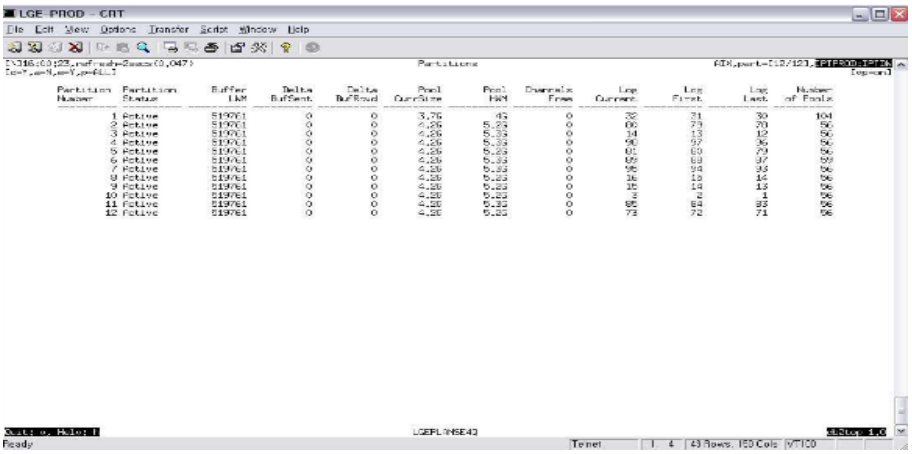


TOPIC 14-33 db2top – Partitioning

Point

db2top을 이용한 파티셔닝 모니터링 수행 방법입니다.

1 p 키를 사용하여 partition을 모니터링 할 수 있습니다. 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.

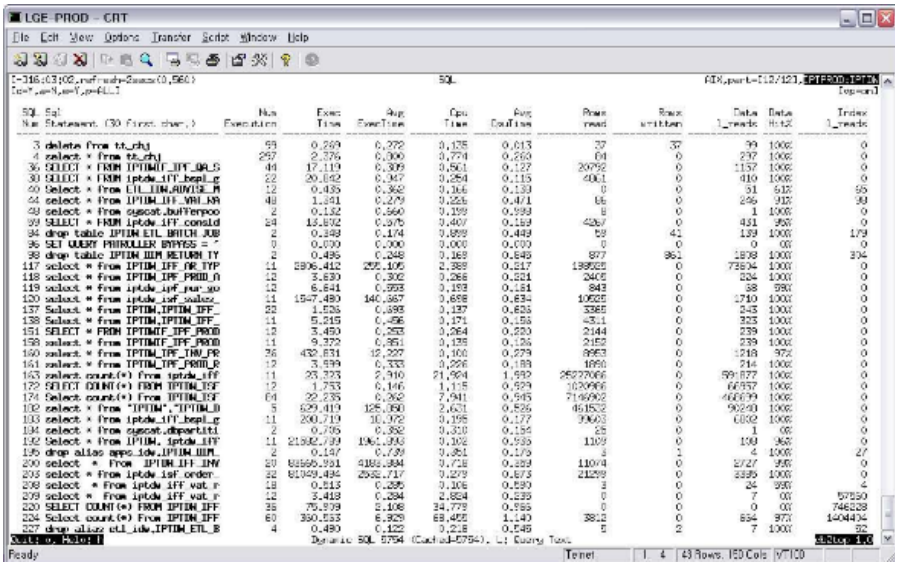


TOPIC 14-34 db2top – Dynamic SQL

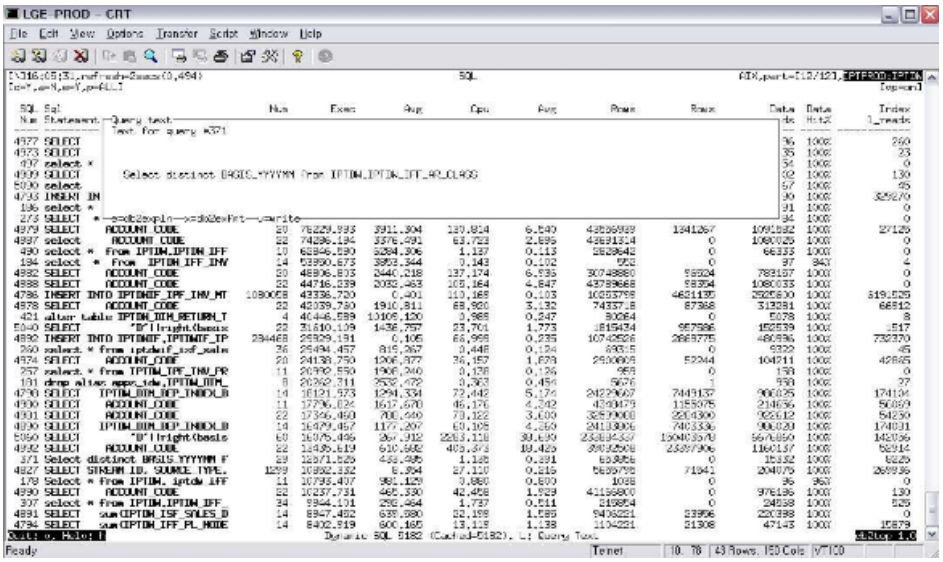
Point

db2top을 이용한 파티셔닝 모니터링 수행 방법입니다.

1 D 키를 사용하여 Dynamic SQL을 모니터링 할 수 있습니다. 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.



2 L 키를 사용하여 SQL 전체 문장을 확인할 수 있습니다. w 키를 사용하여 SQL을 파일로 저장할 수 있습니다. 파일은 홈디렉토리에 dynsql.sql라는 이름으로 생성됩니다.

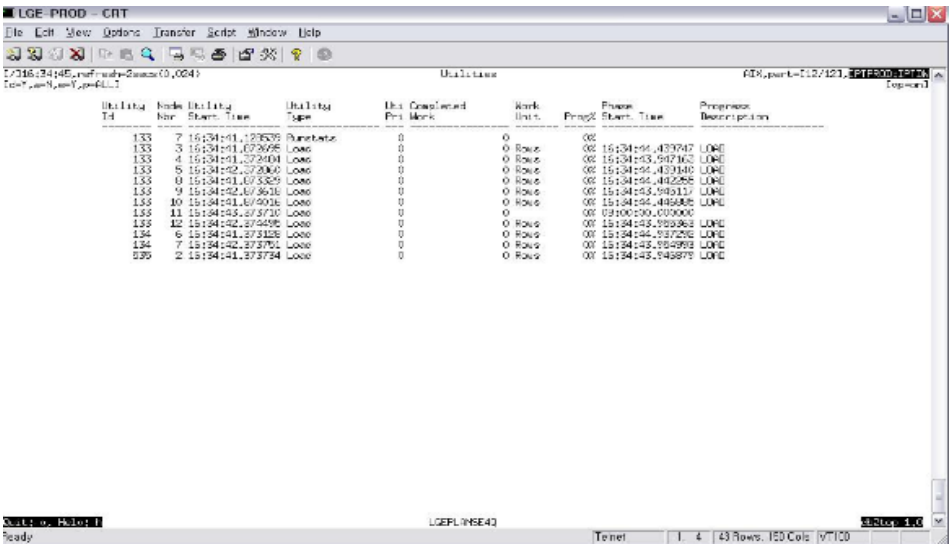


TOPIC 14-35 db2top – Utility

Point

db2top을 이용한 유틸리티 모니터링 수행 방법입니다.

1 u 키를 사용하여 Utility를 모니터링 할 수 있습니다. 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.

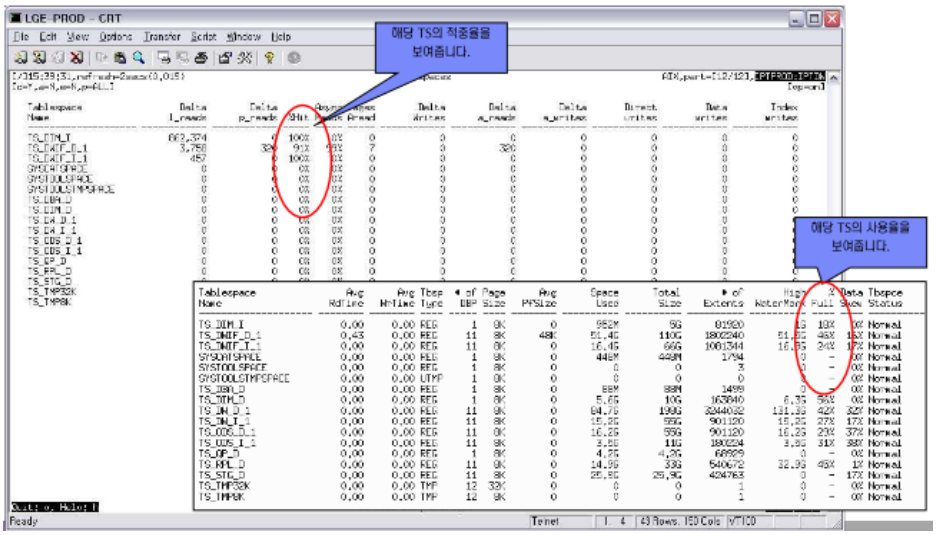


TOPIC 14-36 db2top - Tablespace

Point

db2top을 이용한 Tablespace 모니터링 수행 방법입니다.

1 t 키를 사용하여 tablespace를 모니터링 할 수 있습니다. 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다



UNIT 15

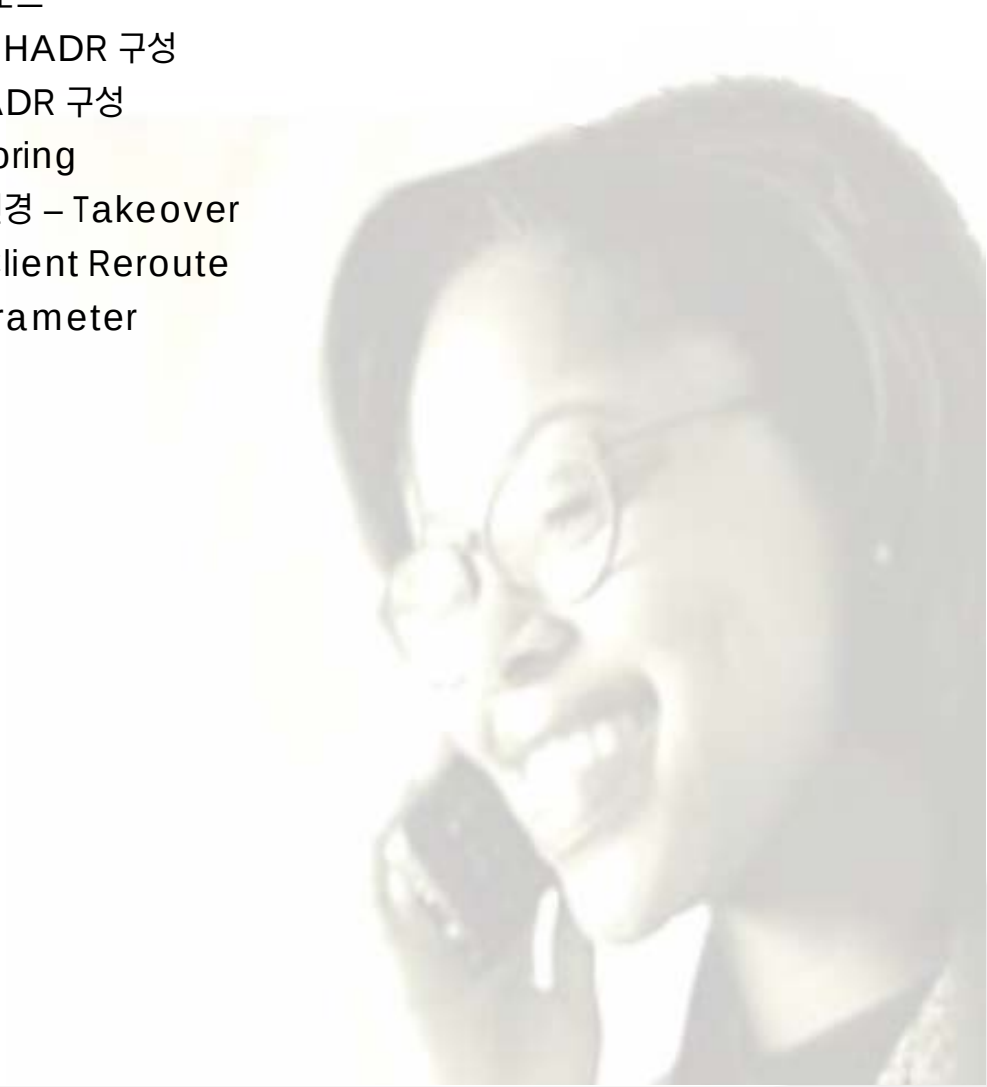
HADR

DB2의 HADR (High Availability Disaster Recovery)은 데이터 복제를 통해 구현되는 고가용 솔루션으로 H/W, Network, S/W 문제 등의 장애 종류와 상관없이 응용프로그램이 최소한의 중지 시간만으로 서비스를 재개할 수 있도록 합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- HADR
- HADR Read On Standby
- HADR 구성 개요
- HADR 동기화 모드
- Wizard를 통한 HADR 구성
- CLP를 통한 HADR 구성
- HADR Monitoring
- HADR Role 변경 – Takeover
- Automatic Client Reroute
- HADR튜닝 Parameter



TOPIC 1501 HADR

Point

HADR 은 두 대(Primary, Standby)의 DBMS의 동기화를 위해 Log Shipping을 사용합니다.

Tip

DB2의 HADR은 8.2 (V8.1 FikPack7이상)부터 사용 가능합니다.

Tip

Log Shipping :
전체 로그 파일을 archive device, 또는 Primary DB에 대해 실행 중인 User Exit 을 통하여 standby server로 복사하는 프로세스입니다.

1 HADR 이란 ?

DB2 HADR은 Database 복제 기술을 이용하여, Site의 부분 또는 전체적인 장애 상황을 대비합니다. HADR환경에서는 Primary와 Standby 두 대의 Server를 운영하게 됩니다.

Server	설 명
Primary Server	Database 소스가 입/출력 되는 곳으로, Transaction이 Primary Server에서 발생하면, Database Log가 자동으로 Standby Server에 전달됩니다.
Standby Server	Primary Server로부터 Backup후 Restore된 복제(clone) Database를 가지게 됩니다.

Tip

HDAR 구성 순서는 다음과 같습니다.

1. Backup & Restore를 통한 Standby Database구성
2. HDAR Parameter 구성을 통한 Database 동기화
3. 사용자 Transaction 처리

2 HADR 가동

Primary DB에서 Capture된 Log Record가 Standby DB에 전송되며, 수신된 Log Record는 Standby DB에서 재실행 됩니다. 이러한 반복적인 재실행을 통해서, Standby DB는 Primary DB에 대한 동기화된 복제 DB로서 장애를 대비하게 됩니다.

Tip

Standby DB가 새로운 DB역할을 인계 받는 것은 “ takeover ” 명령을 통해 이루어집니다.

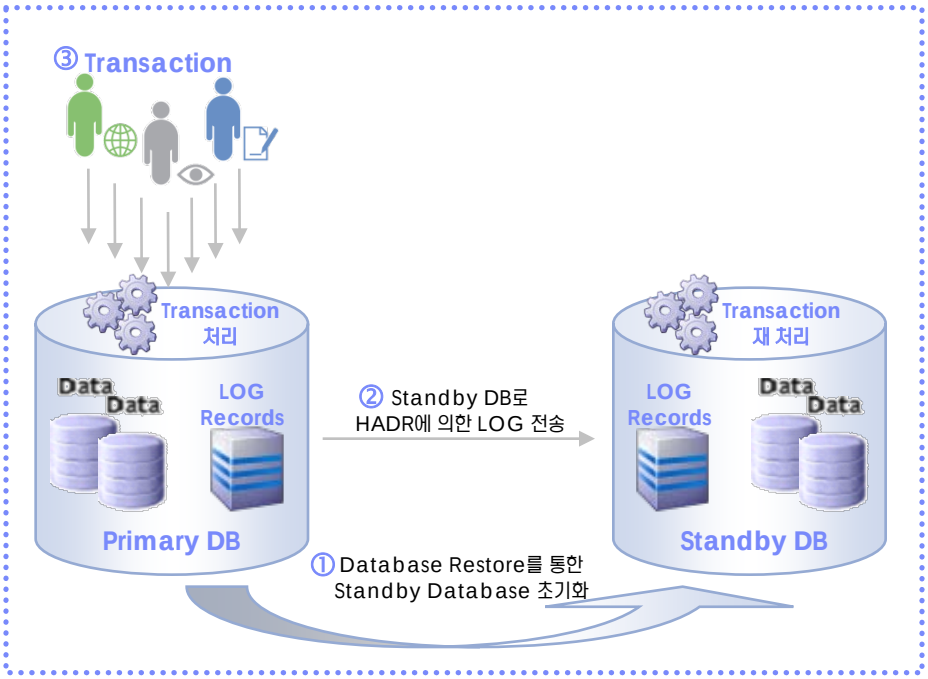


Figure 1501A HADR 의 흐름

TOPIC 1501 HADR

Point

HADR 을 적용하므로 인해 Application은 Primary Server에 장애가 발생하면 자동으로 Standby Server 로 경로를 변경하여 데이터의 손실 없이 수행가능 합니다.

Tip

HADR Take-Over :

사용자의 Take-Over 명령 또는 자동화된 Tool에 의한 Take-Over명령을 수행 할 수 있습니다.

Tip

HADR Catch-up :

Primary과 Standby DB간의 Log Gap에 대해, Standby DB의 Log 재 처리 과정

Tip

Primary와 Standby 서버가 서로 연결되어 통신 가능한 시점부터, HADR 프로세서에 의해 자동으로 동기화 됩니다. 이때, 장애기간 동안 발생된 모든 Transaction Log는 보존되어야 하며, 이를 기반으로 동기화가 수행됩니다.

3 HADR Takeover

Primary DB에 장애가 발생하면, Standby DB가 새로운 Primary DB역할을 인계 받아 Transaction 을 수행하게 됩니다. 또, 장애가 발생한 서버가 복구가 되면, 새로운 Primary DB에서 발생한 Transaction Catch-UP을 통해 재동기화 됩니다. 이 시점부터, 이전의 Primary DB는 새로운 Standby DB가 됩니다.

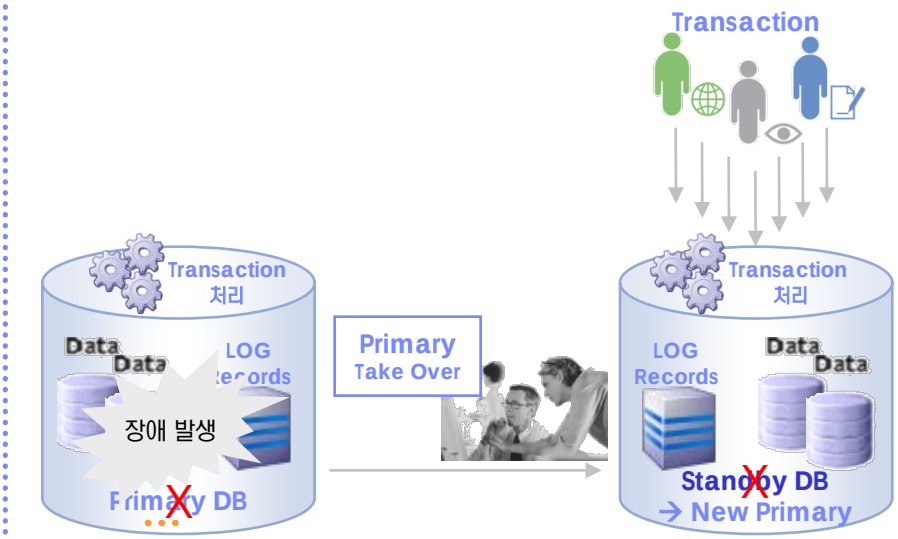


Figure 1501B • • • Standby Database의 Primary Role 수행

4 HADR 재 동기화

새로운 Standby DB가 새로운 Primary와 Catch-up이 종료되고, Transaction의 동기화가 정상적으로 수행되는 중에는, 관리자의 조작(Takeover 명령 또는 또는 GUI)에 의해 Primary와 Standby의 Role을 변경할 수 있습니다.

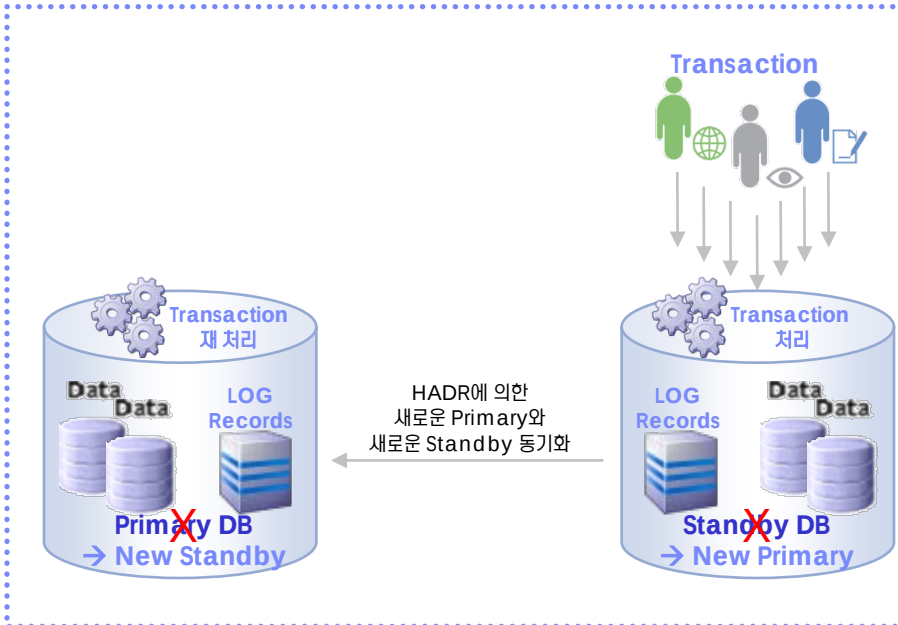


Figure 1501C • • • New Standby DB의 New Primary DB에 대한 Catch-up

TOPIC 1502 HADR Read On Standby

Point

HADR 은 두 대(Primary, Standby)의 DBMS의 동기화를 위해 Log Shipping을 사용합니다.

Tip

- HADR RoS 기능은 9.7 FixPack1부터 지원됩니다.

1 9.7 이후, HADR환경에서 Standby 서버는 조회 업무를 처리할 수 있습니다.

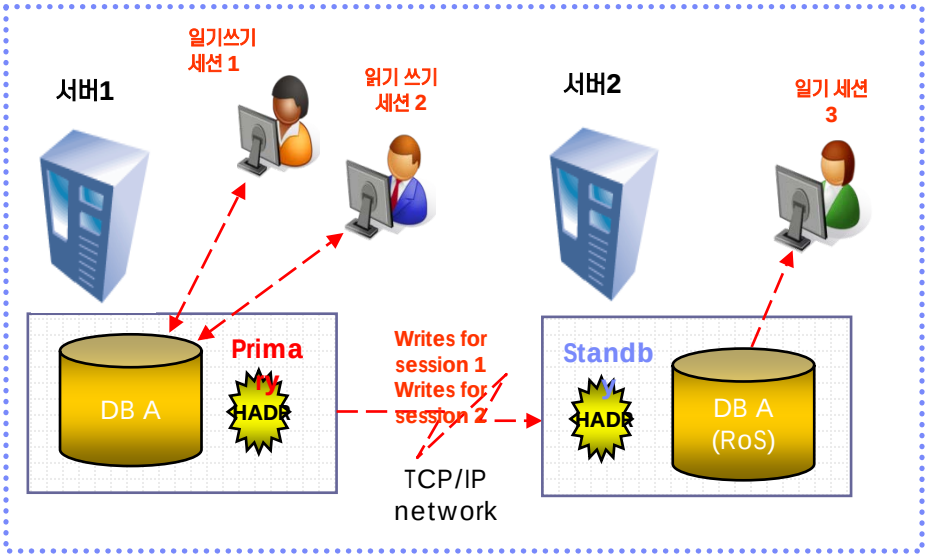


Figure 1502A ••• HADR Reads On Standby

2 RoS 사용을 위해 레지스트리 변수를 아래와 같이 설정합니다. 변경시에는 Instance 재시작이 요구됩니다.

```
$ db2set DB2_HADR_ROS = Y
```

Tip

- Standby에서 지원되지 않는 몇가지 사항이 있습니다.
- 1. LOB, XML, LONG VARCHAR,LONG GRAPHIC 조회
- 2. STMM and WLM 사용
- 3. 사용자 정의 임시 테이블

3 Standby서버에서 지원되는 사항은 아래와 같습니다.

```
- 모든 동기화 모드
        sync
        async
        nearsync
- 모든 연결 상태
        peer to peer
        remote catch up
        local catch up
- 복잡한 DML
        joins
        nested queries
        index scans
- 내부 임시 테이블, 버퍼 풀, 동시에 인덱스 사용 및 인덱스 트리 구조 변경
```

TOPIC 15-03 HADR 구성 개요

Point

HADR은 매우 간단하고 쉽게 구성할 수 있습니다.

Tip

Log Archive 모드 :
동기화를 수행하지 못하는 환경에서도 Primary는 지속적으로 Transaction 처리를 하며, 향후, Catch-Up시 Log (Archive 포함)를 사용하기 때문에, DB에 대한 Logging 방법은 반드시 Archive Mode로 운영되어야 합니다.

1 HADR 구성 준비 (HADR을 위한 Primary와 Standby DB준비)

- HADR의 Pair (Primary & Standby)를 결정합니다.
- Primary DB에 대한 Archive Log mode를 설정합니다.
- HADR를 위한 DATABASE Configuration Parameter를 Primary와 Standby 에 설정합니다.

DB CFG parameter	설 명
HADR_LOCAL_HOST	HADR 통신을 위한 Local Host정보 (TCP/IP server정보)를 정의합니다. - IP 주소 또는 HOSTNAME이 사용됩니다.
HADR_LOCAL_SVC	HADR Process가 연결을 Accept할 TCP/IP Service 또는 Port Number를 정의합니다.
HADR_REMOTE_HOST	HADR 2차 Node에 대한 TCP/IP HOSTNAME이나 IP 주소를 정의합니다.
HADR_REMOTE_SVC	HADR 2차 Node에 대한 TCP Service 이름 또는 Port 이름을 정의합니다.
HADR_REMOTE_INST	2차 서버에 대한 Instance명을 정의합니다.
HADR_TIME_OUT	HADR Pair 서버간의 Communication 장애를 결정하기 전까지 프로세스가 기다리는 시간(초 단위)을 정의합니다.
HADR_SYNCMODE	동기화 모드를 정의합니다. - 서버가 서로 Peer 상태일 때, Primary Log가 Standby DB에 동기화 되는 모드를 정의합니다. - SYNC / NEARSYNC / ASYNC
HADR_DB_ROLE	Database의 현재(Standard, Primary, Standby)의 역할을 나타냅니다.
HADR_PEER_WINDOW	둘간의 연결이 끊어져도 지정된 시간만큼 여전이 Peer상태로 간주하고 동작. 데이터 일관성 확인에 도움이 됩니다.

Tip

Tablespace 및 Container :
Standby DB의 Tablespace및 Container 구성은 Primary DB와 같은 구조를 가져야 합니다. 즉, Container의 이름, PATH 및 크기 까지 Primary DB와 반드시 일치하여야 합니다.

2 Primary DB 복제

Primary DB의 Full Backup 이미지를 Standby Server에 Copy하여, Restore합니다. Standby 시스템에서 DB Restore한 이후에 Standby DB는 “Roll Forward Pending ” 상태가 됩니다. 즉, Standby DB는 Fail-Over에 의해 Primary 역할을 수행하기 전까지는 활성화 되지 않으며, READ & WRITE를 허용하지 않습니다.

- Primary Database Full Backup
- Standby 서버로 Copy
- Standby 서버에서 Restore
- HADR configuration Parameter 변경 및 적용

TOPIC 1503 HADR 구성 개요

Point

Primary DB 복제 후 Standby DB –> Primary DB 순으로 반드시 가동해야 합니다.

3 Standby DB Start

Standby DB가 시작되면, Local Catch-Up 상태가 됩니다.

Pending Log Record가 Standby DB에서 재 수행됩니다.

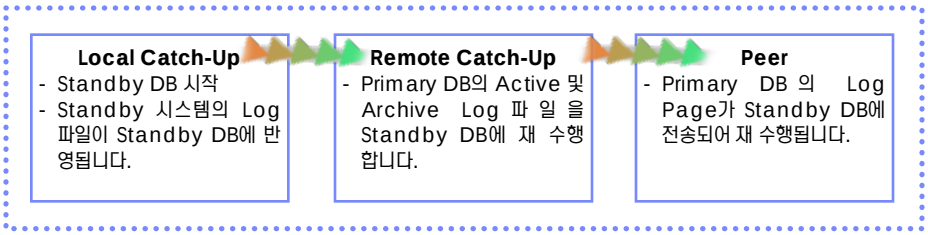


Figure 1503A Standby Database의 상태 변화

Standby DB 를 시작하면 Local Catch-Up 상태가 됩니다.

Standby 시스템 의 Local Pending Log를 모두 처리하면, Standby DB는 Remote Catch-Up상태가 되며, 이때 Standby DB 는 Primary DB 의 Log를 모두 재 수행하고, Standby DB 가 Remote Catch-Up 상태가 되기 위해서는 Primary DB 가 반드시 활성화 되어야 합니다.

Primary DB 의 모든 Log가 적용되고 나면, Standby DB 는 Peer 상태가 되고, Primary DB 의 Log가 Disk로 쓰여질 때마다 Log Record가 Standby DB 에 전송되어 적용(재 수행)됩니다.

Tip

Standby DB 에 Log가 적용되는 방식은 다음과 같습니다.

1. SYNC
2. NEARSYNC
3. ASYNC

4 Primary DB Start

Primary DB 가 시작되면, Standby DB 가 접속을 대기합니다. Standby DB 가 정해진 임계시간 내에 Primary DB 에 접속하지 못하면, HADR은 시작되지 못합니다.

HADR_TIMEOUT Configuration Parameter를 이용해 Time-Out 기간을 정의가능합니다.

Primary Database 복제

↓

HADR Parameter 구성

↓

Standby Database 시작

↓

Primary Database 시작

TOPIC 1504 HADR 동기화 모드

Point

HADR은 3가지 동기화 모드를 제공합니다. Sync, Nearsync, Async이며, NearSync가 기본 옵션입니다.

Tip

가용성 정도 또는 성능 조건에 따라 동기화 모드를 선택합니다.

1 동기화 (Synchronization) 모드

Primary DB와 Standby DB 사이에 Log 적용방식을 관리하기 위해 동기화 (Synchronization) 모드를 정의합니다.

- SYNC (Synchronous)
- NEARSYNC (Near Synchronous)
- ASYNC (Asynchronous)

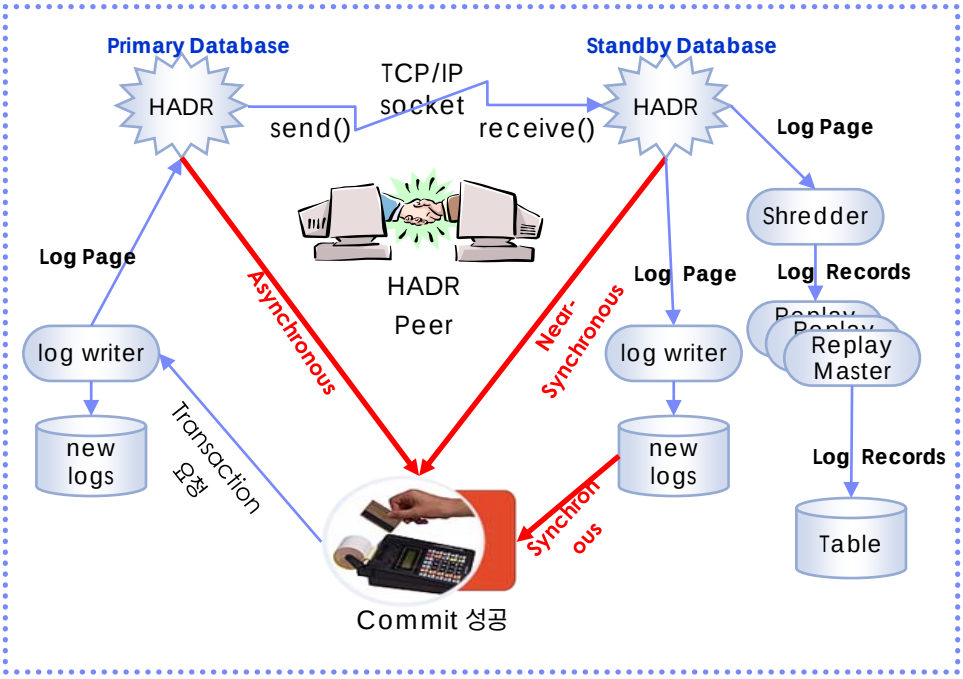


Figure 1504A ••• HADR 동기화 모드

2 Sync 모드 - Commit 조건

- Primary DB 의 Disk의 Log 파일에 Log Record가 저장됨.
- Standby DB에 Log가 정상적으로 적용되었음을 Standby DB로부터 응답을 받음.
- Log Data가 Primary DB와 Standby DB에 모두 저장되었음을 보장함.

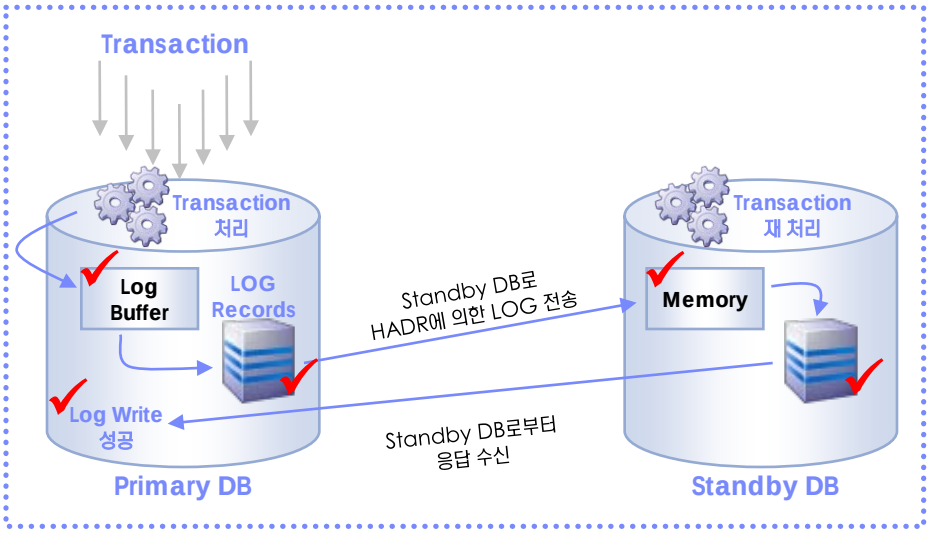


Figure 1504B ••• 동기화 모드 : SYNC

TOPIC 1504 HADR 동기화 모드

Point

Sync모드는 HA요건인 경우, Aync모드는 원격 DR요건에 주로 사용됩니다.

Tip

만일, Main Memory의 Log Record가 Disk의 Log에 쓰여지지 않은 상태에서 Primary DB와 Standby DB 모두 장애가 발생하면, 양 DB에 Log Record 불일치하는 사항이 발생 할 수 있음.

3 NearSync 모드 - Commit 조건

- Primary DB의 Disk의 Log 파일에 Log Record가 저장됨.
- Standby DB의 Main Memory에 Log Record가 정상적으로 쓰여졌음을 Standby DB로부터 Primary DB 응답을 받음.

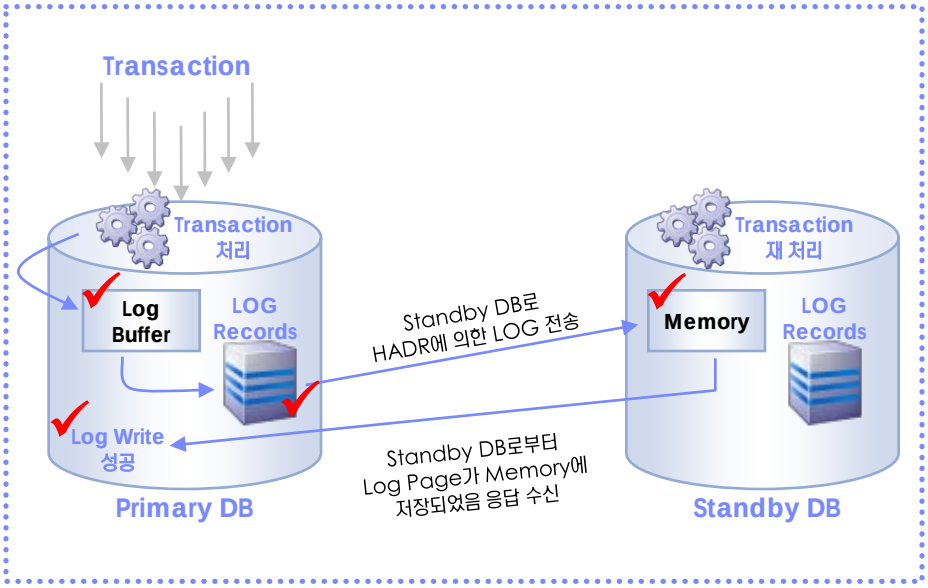


Figure 1504C ••• 동기화 모드 : NEARSYNC

Tip

Primary Database, Network 또는 Standby Database의 장애 시, Log 파일의 처리 중인 Transaction에 대해 손실될 가능성이 있음.

4 Async 모드 - Commit 조건

- Primary Database의 Disk의 Log 파일에 Log Record가 저장됨.
- Log Record가 Standby Database에 전달됩니다. HADR 프로세스로부터 Socket을 통한 Log Record를 수신 받았음을 응답함.

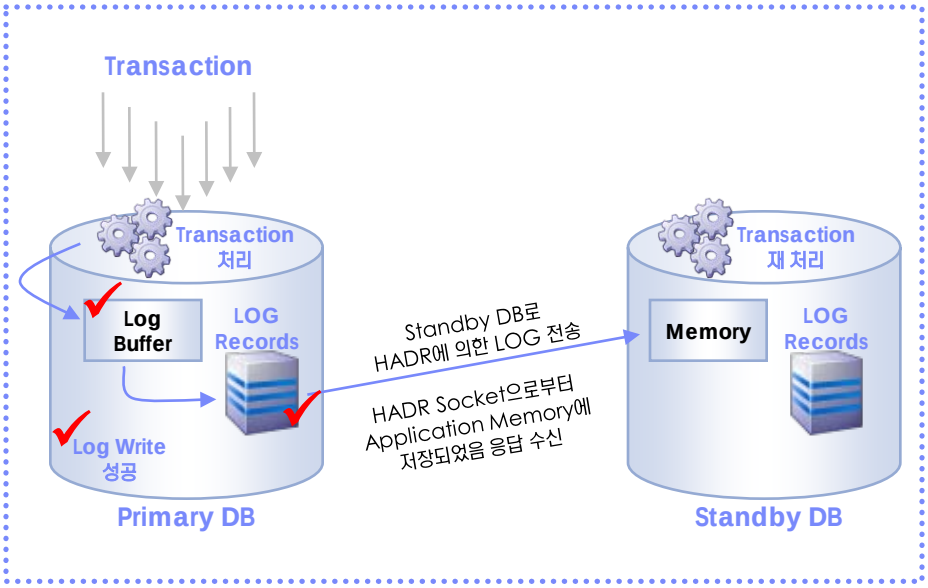


Figure 1504D ••• 동기화 모드 : ASYNC

TOPIC 1505 Wizard를 통한 HADR 구성

Point

기본으로 제공하는 GUI를 통해서 HADR을 손쉽게 구성할 수 있습니다.

Tip

HADR을 구성하고자 하는 Primary DB 가 Client 에서 Catalog 되어 있어야 합니다.

1 HADR Wizard 시작하기

DB2에서 제공하는 GUI(HADR 구성마법사)를 통해 HADR 구성환경을 손쉽게 구현하는 방법을 소개합니다. HADR 구성을 포함한, Start / Stop, Database Role 전환 등의 관리 업무를 수행 가능합니다.

마법사를 시작하려면, DB2 제어센터에서 HADR을 위한 Database에서 오른쪽을 클릭하여 “고가용성 재해 복구” 를 수행합니다.

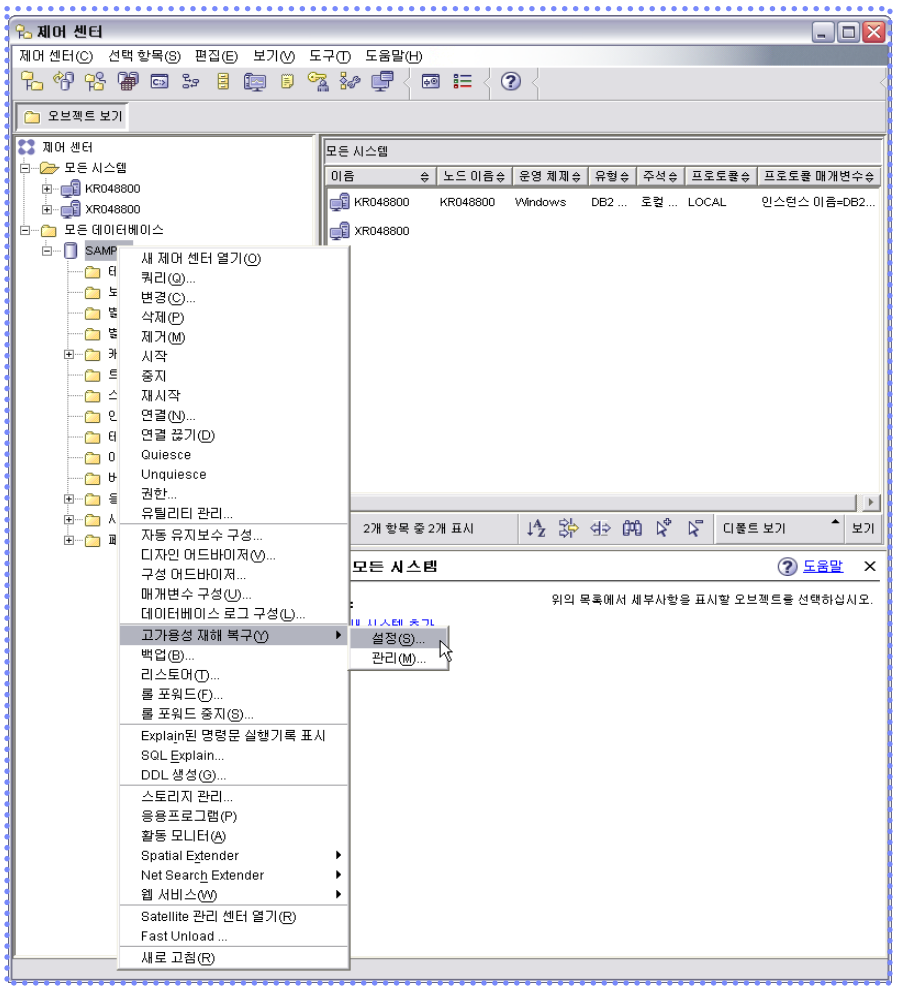


Figure 1505A ••• 제어센터에서 HADR Wizard 수행

TOPIC 1505 Wizard를 통한 HADR 구성

Point

우선 데이터베이스 로그마법사를 통해 archive 로그 모드로 전환합니다.

"HADR 데이터베이스 설정" GUI Wizard를 통해 간단하게 구현할 수 있습니다.

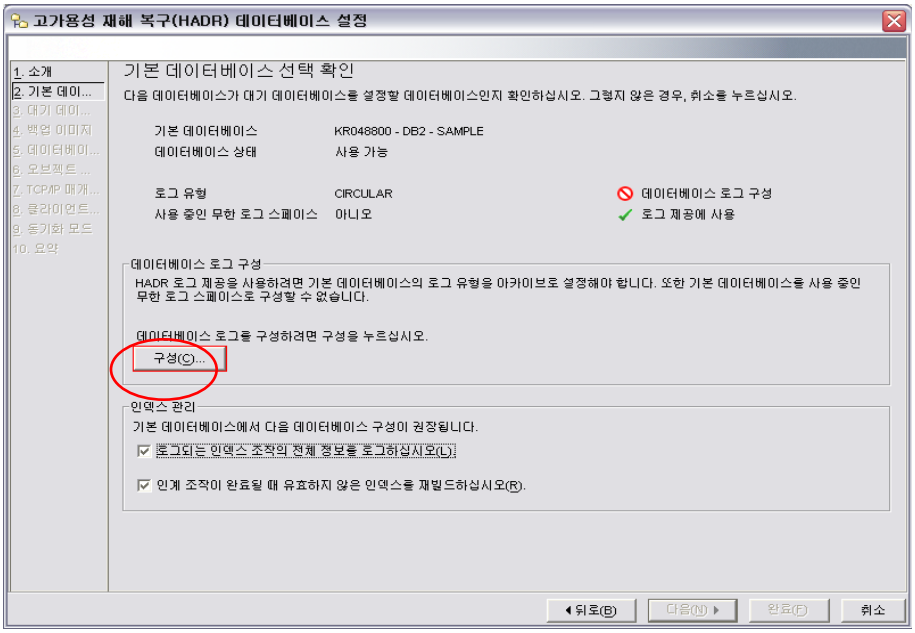


Figure 1505B ••• 제어센터에서 HADR Wizard 수행

2 데이터베이스 로그 마법사 구성

➲ HADR환경을 운영하기 위해서는 Log 관리를 Archive모드로 운영해야 합니다.

➲ "데이터베이스 로그 마법사" 를 통해 순환(Circular)로그를 아카이브(Archive)로그 유형으로 전환 할 수 있습니다.

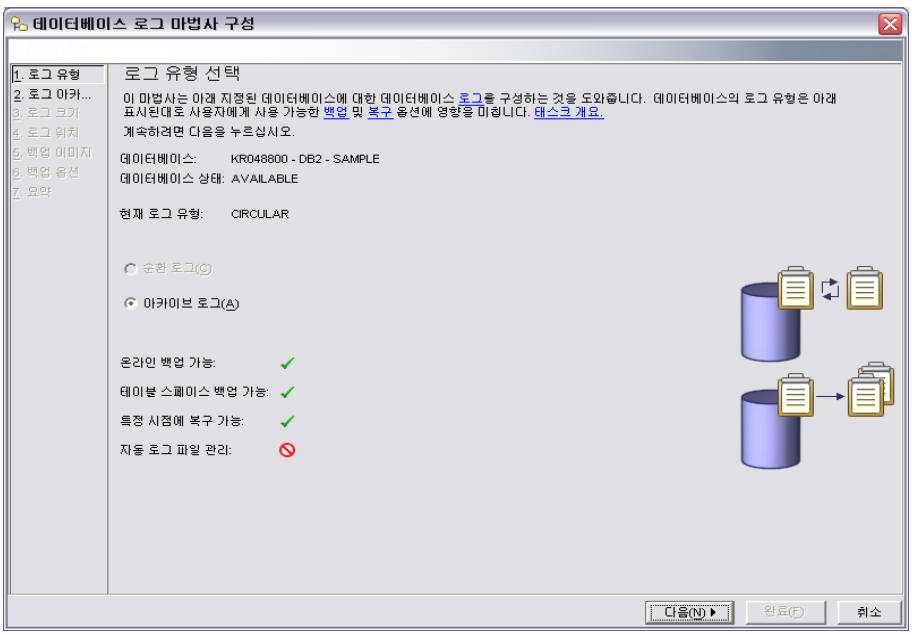


Figure 1505C ••• 데이터베이스 로그마법사구성 - 로그유형

TOPIC 1505 Wizard를 통한 HADR 구성

Point

우선 데이터베이스 로그마법사를 통해 archive 로그 모드 전환 및 로그 크기를 지정해줍니다.

Tip

HADR 을 DB2 V8.2 이상에서 사용할 수 있듯이 logarchmeth1도 DB2 V8.2 이상에서 나오는 DB CFG 매개변수입니다.

Tip

LogArchMeth1: HADR환경에서 LogArchMeth1를 사용하여 Archive를 관리하면, Remote Catch-Up 시점에 DBA가 별도로 Archive 파일을 로그 경로로 복사해 줄 필요가 없습니다.

Tip

아카이브 로깅에서는 Remote Catch-Up 할 때, Archive Log를 DBA가 복사해야 하는 추가적인 관리가 필요합니다.

Tip

USER EXIT
sqllib/samples/c 폴더에 있는 db2uext2.* 예제 파일을 수정하여 적절한 위치에 저장하면 Archive를 실행하게 합니다.

Database의 Log를 Archive 하는 방법 선택

- 수동 아카이브 : Archive Log를 DBA가 관리.
- User Exit 루틴 : User Exit Program을 사용하여 Archive Log를 관리.
- DB2를 사용하여 자동으로 로그파일 아카이브 : DB CFG의 logarchmeth1를 통해 Log를 Archive 하도록 합니다. 미디어 유형 및 아카이브 로그 경로를 지정합니다.

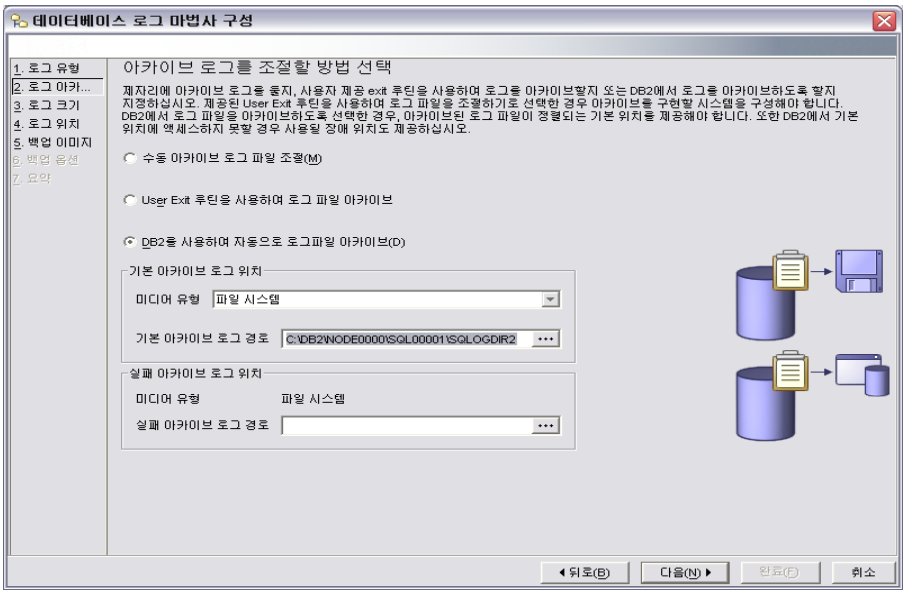


Figure 1505D • • • 데이터베이스 로그마법사구성 – 로그 아카이브

➲ 로그 파일의 수와 크기(1차 로그 파일 및 2차 로그 파일 수와 각 로그 파일의 크기) 를 정의 합니다.

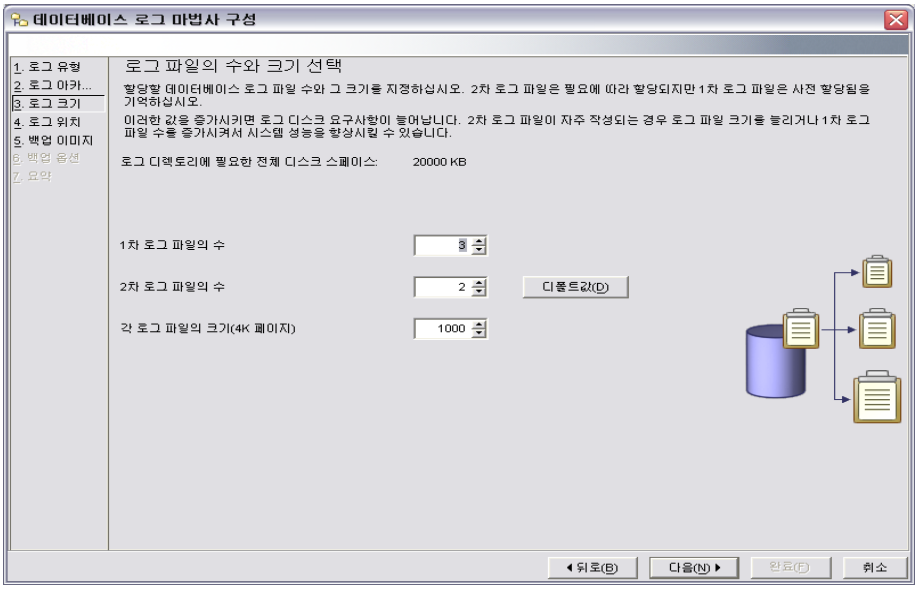


Figure 1505E • • • 데이터베이스 로그마법사구성 – 로그크기

TOPIC 1505 Wizard를 통한 HADR 구성

Point

다음 데이터베이스 로그마법사를 통해 로그위치 및 백업이미지 위치를 지정해줍니다.

➲ 사용 중인 Log 경로를 정의합니다. 또한 Log 파일의 손상을 대비하여 로그 파일 미러(Mirror) 경로를 정의 할 수 있습니다.

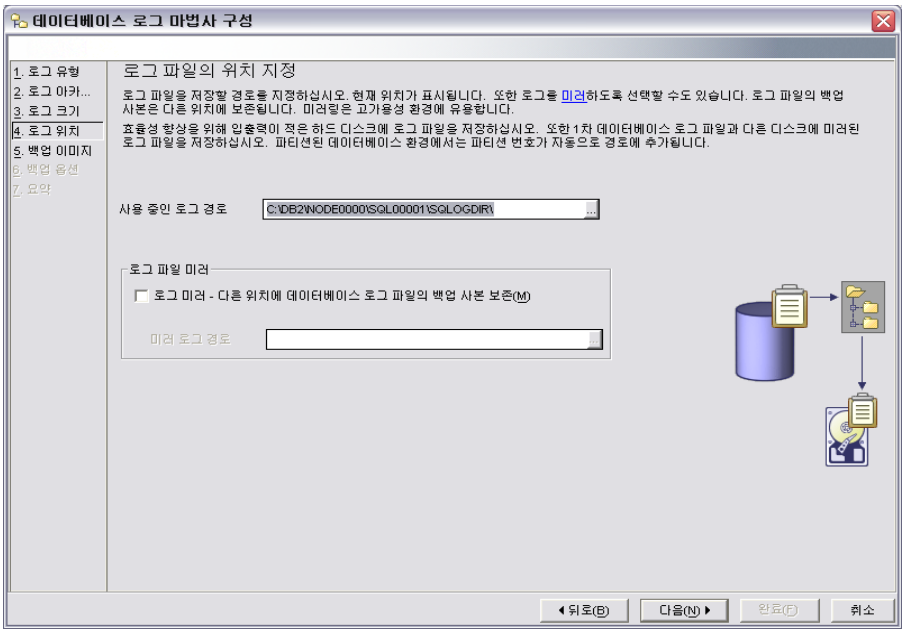


Figure 1505F ••• 데이터베이스 로그마법사구성 – 로그 위치

➲ Circular Log환경에서 Archive Log환경으로 전환 시, DB CFG Parameter를 변경한 후, Database Full Backup을 수행하여야 합니다. Backup 이미지를 저장할 유형과 경로를 지정합니다.

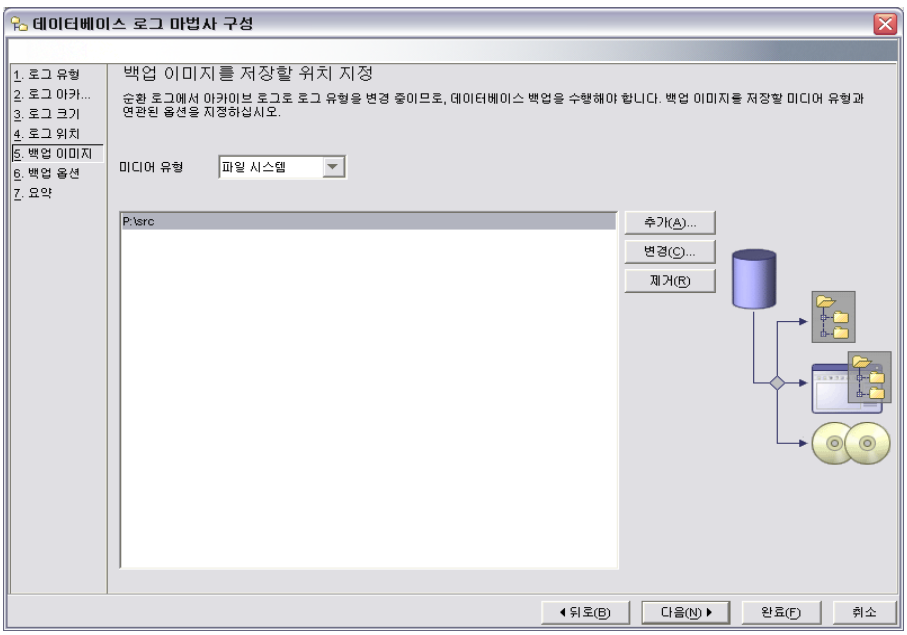


Figure 1505G ••• 데이터베이스 로그마법사구성 – 백업 이미지 관련

TOPIC 1505 Wizard를 통한 HADR 구성

Point

Backup 에 대한 옵션을 지정한 후 다시 한번 설정을 확인합니다.

Tip

Backup에 대한 병렬처리 수, Buffer 수/크기를 지정은 Backup 성능에 영향을 미칩니다.

➲ Backup에 대한 병렬처리 수, Buffer 수/크기를 지정합니다. Offline full Backup을 위해서는 데이터베이스를 Quiesce모드에서 Backup을 하여야 합니다. 또한 Backup 이미지를 압축할 것인지를 결정합니다.

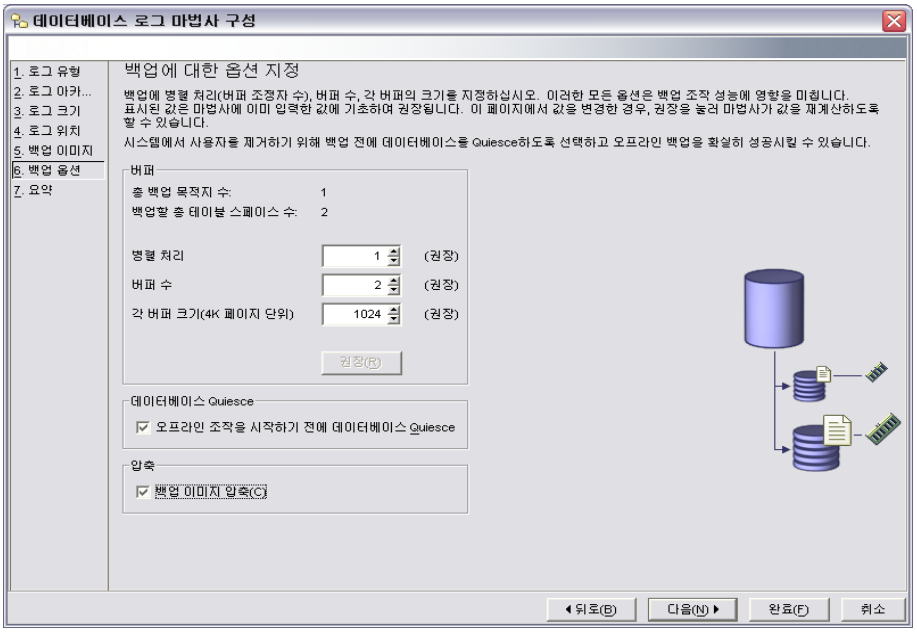


Figure 1505H ••• 데이터베이스 로그마법사구성 – 백업 옵션

Tip

"명령표시" 를 클릭하면 실행명령을 직접 확인할 수 있으며, 이 명령은 GUI를 사용하지 않고 CLP에서 바로 사용가능 합니다.

➲ 위의 모든 설정을 지정한 후, 완료를 Click하여 Offline Backup을 수행합니다.

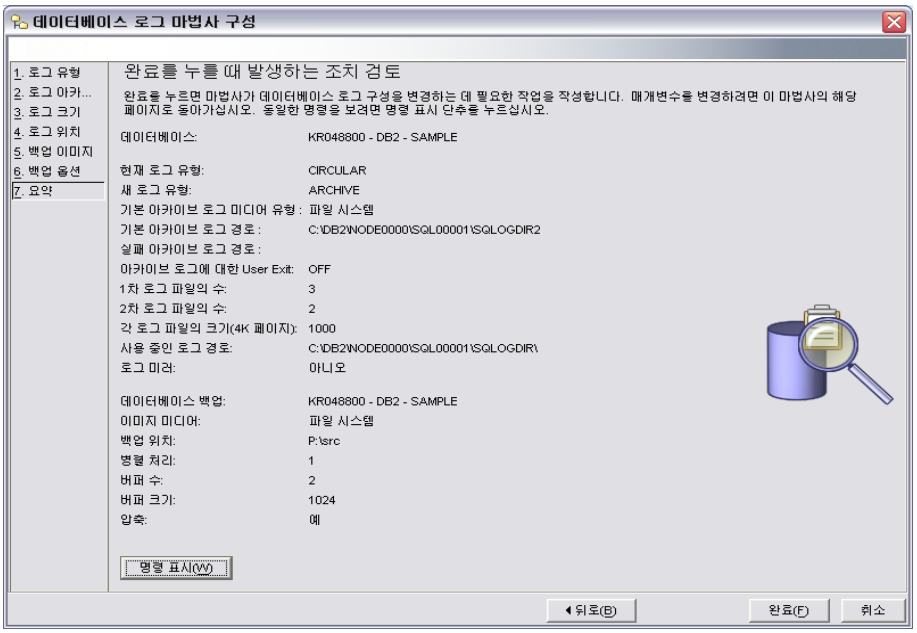


Figure 1505I ••• 데이터베이스 로그마법사구성 – 요약

TOPIC 1505 Wizard를 통한 HADR 구성

Point

“ 명령표시” 를 통해 실제 DB2 명령을 확인할 수 있습니다.

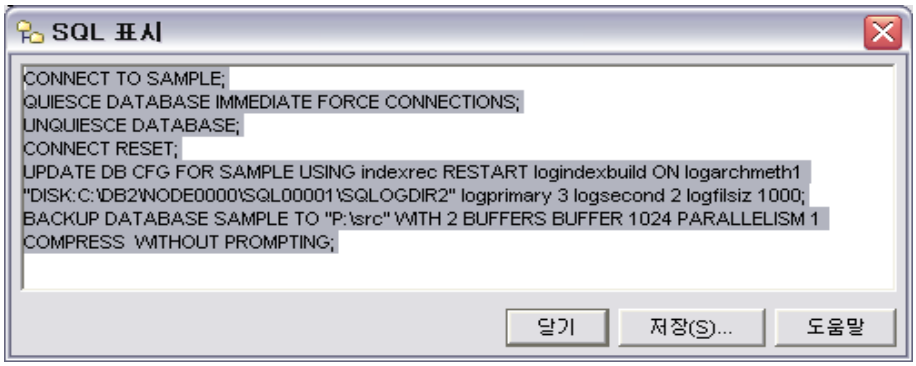


Figure 1505J ••• 데이터베이스 로그마법사구성 – 요약 -> 명령표시

➲ 다음은 Archive Log 환경을 위한 DB CFG 변경 내용 및 Backup을 수행하는 Script 예 입니다.

```
CONNECT TO SAMPLE;


QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;

UNQUIESCE DATABASE;

CONNECT RESET;


UPDATE DB CFG FOR SAMPLE USING
      indexrec RESTART
      logindexbuild ON
      logarchmeth1 "DISK:C:\DB2\NODE0000\SQL00001\SQLOGDIR2"
      logprimary 3
      logsecond 2
      logfilsiz 1000;


BACKUP DATABASE SAMPLE TO "P:\src"
      WITH 2 BUFFERS
      BUFFER 1024
      PARALLELISM 1
      COMPRESS
      WITHOUT PROMPTING;
```

TOPIC 1505 Wizard를 통한 HADR 구성

Point

데이터베이스 로그 마법사구성이 정상적으로 완료되었는지 확인합니다.

➲ 데이터베이스 로그 마법사 구성에서 “완료” 를 Click하게 되면 다음과 같은 Message와 함께 Backup을 수행하게 됩니다.

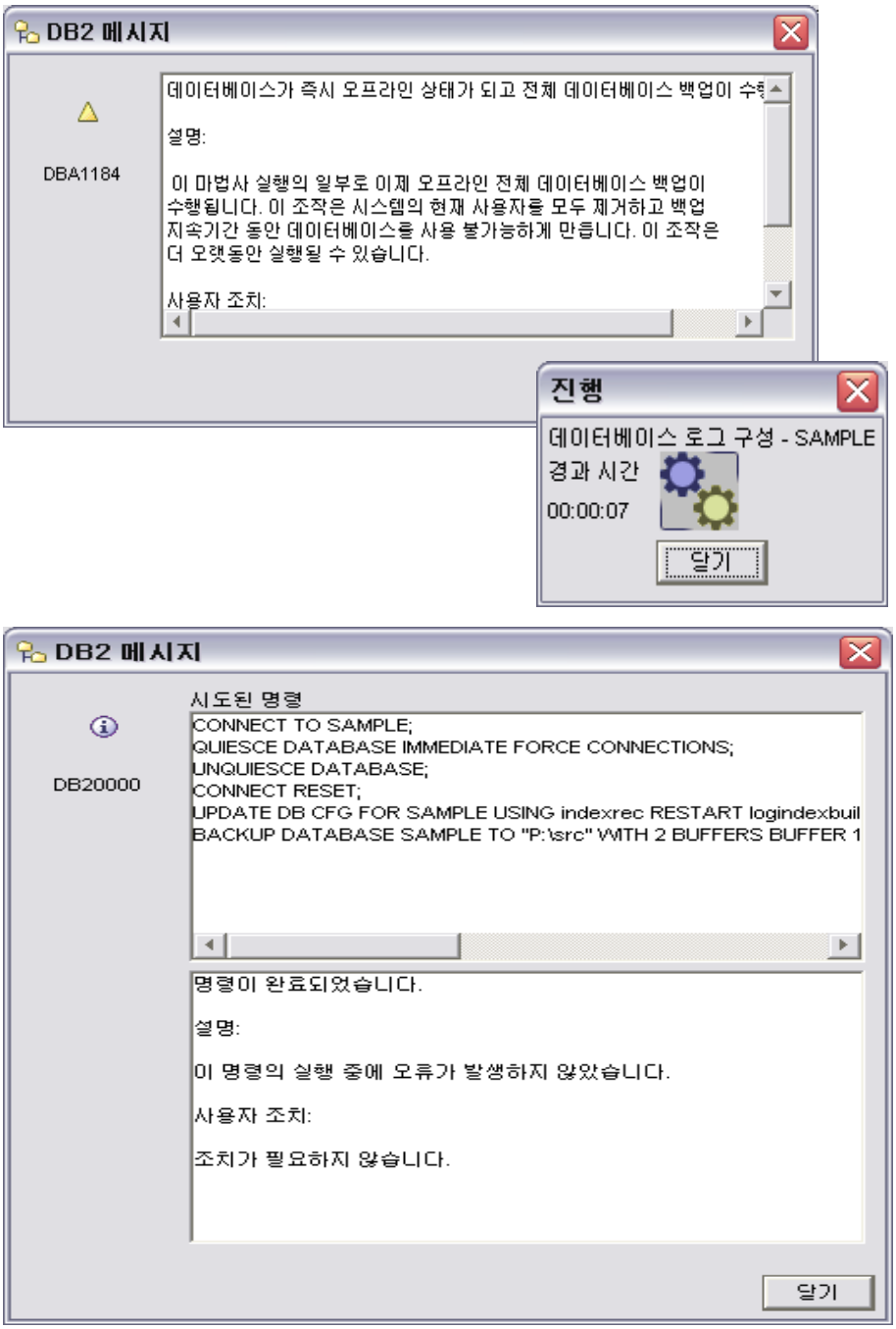


Figure 1505K • • • 데이터베이스 로그마법사구성 – 진행 및 메시지 확인

TOPIC 1505 Wizard를 통한 HADR 구성

Point

이제 Wizard를 통해 본격적인 HADR 을 구성합니다.

3 데이터베이스 로그 구성을 마치면, 기본데이터베이스 선택화면이 다음과 같이 화면이 바뀝니다.

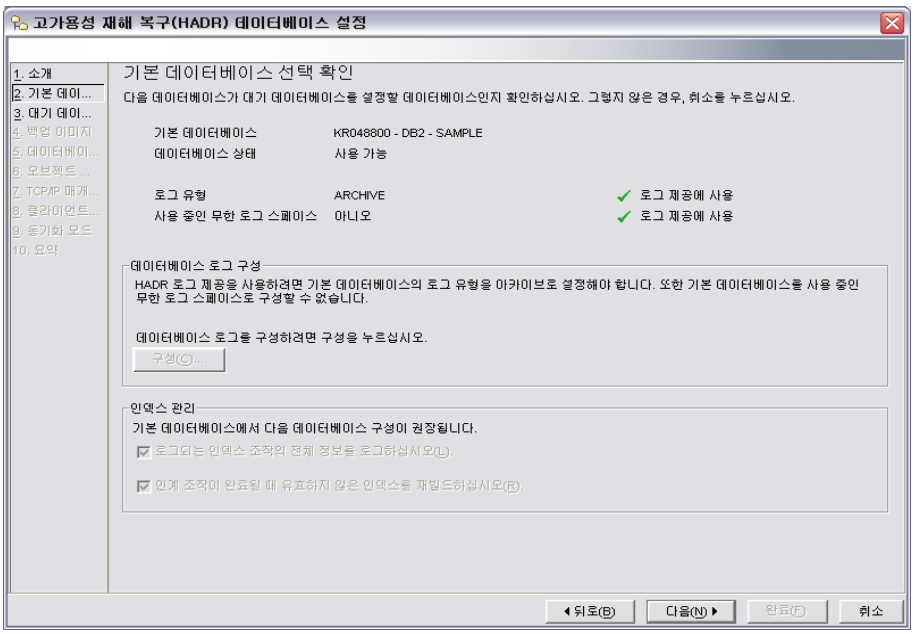


Figure 1505L ••• HADR 데이터베이스 설정 – Primary DB 선택

➲ HADR 구성을 취소하거나 재구성을 시도하면, 기본 데이터베이스 (Primary DB)가 Archive Log 상태에 있으면 다음과 같은 화면이 표시됩니다.

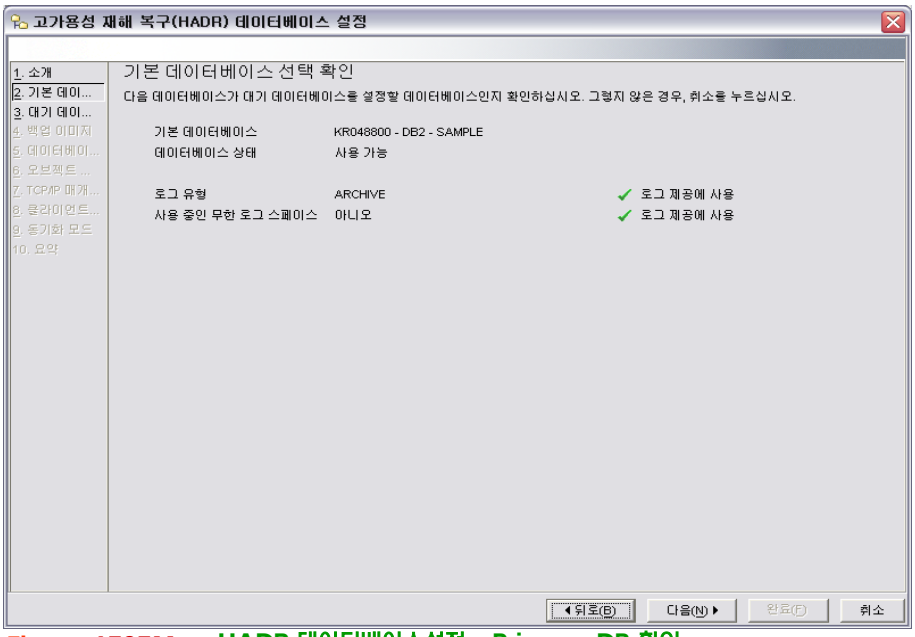


Figure 1505M ••• HADR 데이터베이스설정 – Primary DB 확인

TOPIC 1505 Wizard를 통한 HADR 구성

Point

이제 Wizard를 통해 본격적인 HADR 을 구성합니다.

➲ 대기 데이터베이스 (Standby DB)를 구성하는 화면입니다.

➢ 시스템이름과 인스턴스 이름을 선택.

➢ Standby DB 초기화를 위해 기본데이터베이스 백업 이미지를 선택.

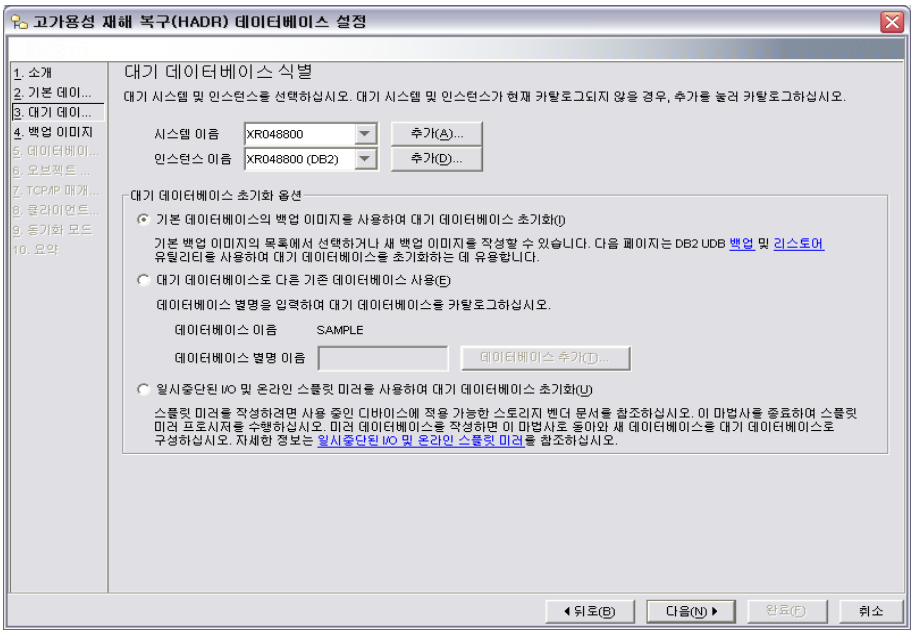


Figure 1505N • • • HADR 데이터베이스 설정 – 대기 DB 식별

Tip

Primary DB에서 Backup을 수행한 후, 이를 사용할 수 있습니다.

➲ Primary DB의 Backup 이미지를 지정합니다. 이미 Backup을 수행한 경우에는 List에서 Backup 이미지를 선택할 수 있습니다.

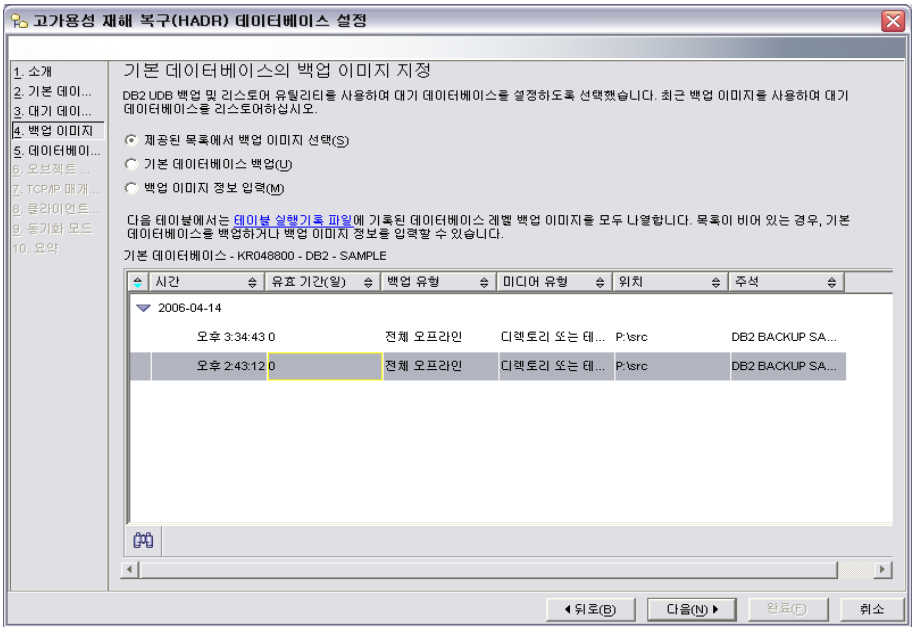


Figure 1505O • • • HADR 데이터베이스 설정 – 백업이미지 지정

TOPIC 1505 Wizard를 통한 HADR 구성

Point

이제 Wizard를 통해 본격적인 HADR 을 구성합니다.

➲ 대기 데이터베이스 (Standby DB)를 Restore하는 방법을 정의합니다.

➲ Backup 이미지 전송을 위해 Primary DB와 Standby DB의 Path을 정의합니다.

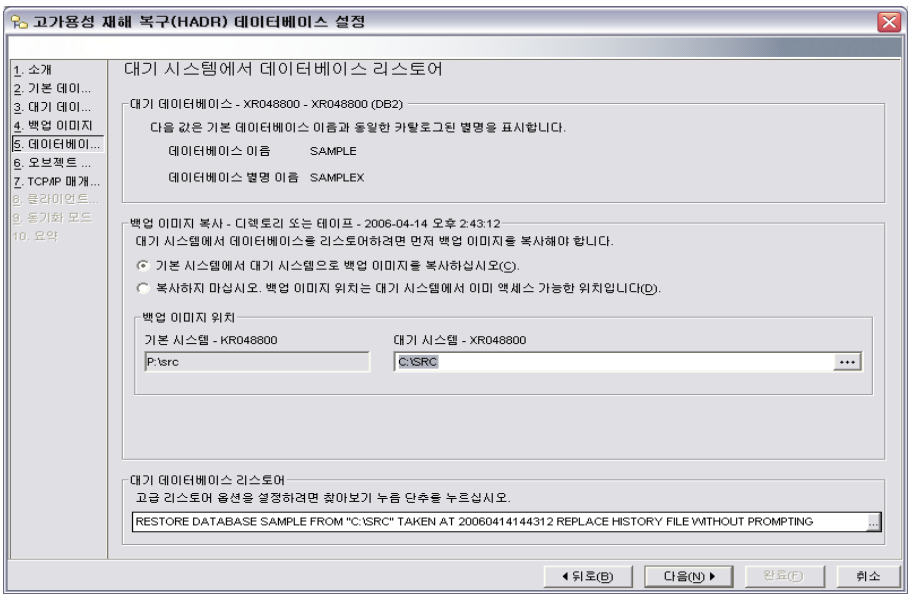


Figure 1505P • • • HADR 데이터베이스 설정 – 대기시스템에서 DB restore

```
RESTORE DATABASE SAMPLE FROM "C:\SRC"
      TAKEN AT 20060414144312
      REPLACE HISTORY FILE WITHOUT PROMPTING
```

Tip

UDF 및Stored Procedure와 같은 외부에 저장되어 있는 Object에 대한 이동 방법을 정의합니다.

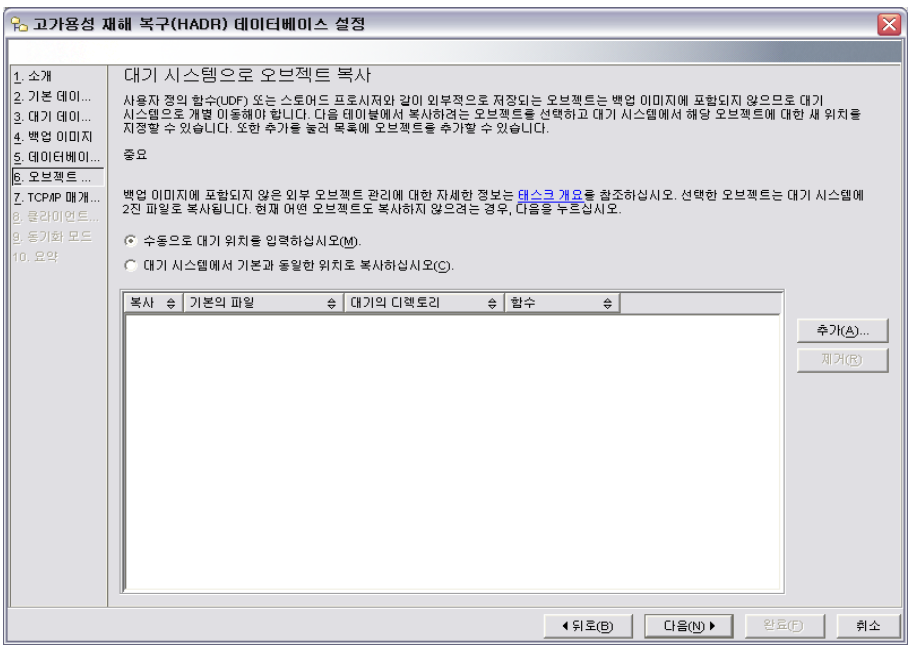


Figure 1505Q • • • HADR 데이터베이스 설정 – 오브젝트복사

TOPIC 1505 Wizard를 통한 HADR 구성

Point

이제 Wizard를 통해 본격적인 HADR 을 구성합니다.

Tip

services 파일과 hosts을 사용하거나, 직접 IP Address와 Port를 지정할 수 있습니다.

Tip

N/W IP & Service Port : HADR을 위해서 Log Shipping 및 Health Check을 위한 IP와 Port가 필요합니다. 이를 services 및 hosts 파일에 등록하여 사용합니다. 이때, Service Port가 다른 Port와 충돌이 발생하지 않도록 선정에 유의하여야 합니다.

Tip

services 파일과 hosts 파일의 위치는 Unix / Linux 에서는 /etc 이고, Windows 시스템에서는 c:\windows\system32\drivers\etc 입니다.

➲ Primary DB와 Standby DB에 대한 TCP/IP 통신 Parameter를 지정합니다.

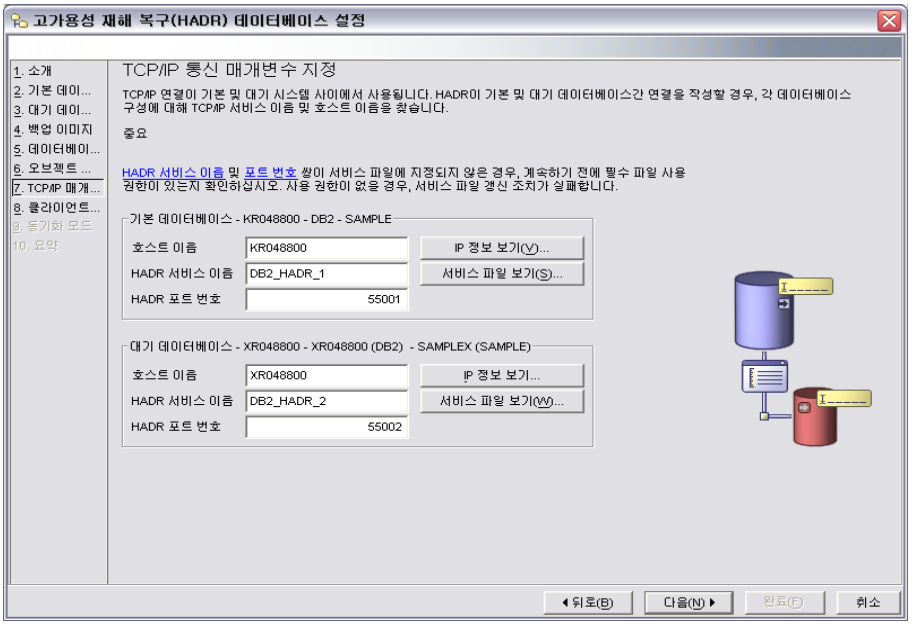


Figure 1505R ••• HADR 데이터베이스 설정 – TCP/IP 지정

➲ 장애 시, Automatic Client Reroute를 위한 대체서버 (Alternate Server)에 대한 정보를 입력합니다.

➲ Primary DB와 Standby DB에 대한 HOST 정보와 PORT정보를 설정합니다.

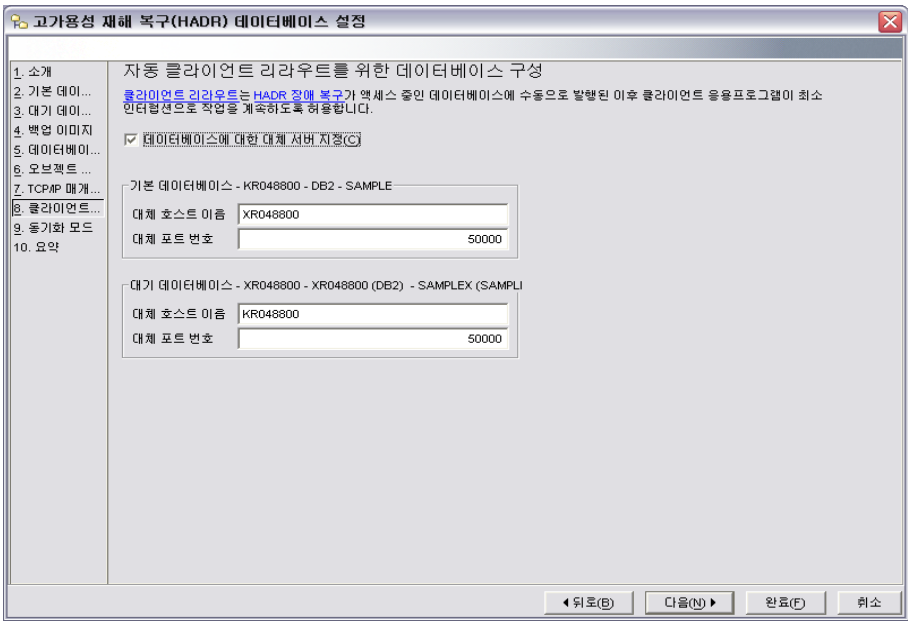


Figure 1505S ••• HADR 데이터베이스 설정 – 리라우트를 위한 구성

TOPIC 1505 Wizard를 통한 HADR 구성

Point

Wizard를 통한 HADR 구성을 완료합니다.

➲ Primary Database와 Standby Database가 PEER상태일 때, Log동기화를 위한 방법을 지정합니다.

- 동기 : SYNC / 동기에 가까움 : NEARSYNC / 비동기 : ASYNC

➲ HADR Pair 서버간의 통신장애를 결정하기 전까지 프로세스가 기다리는 시간(초 단위)을 정의합니다.

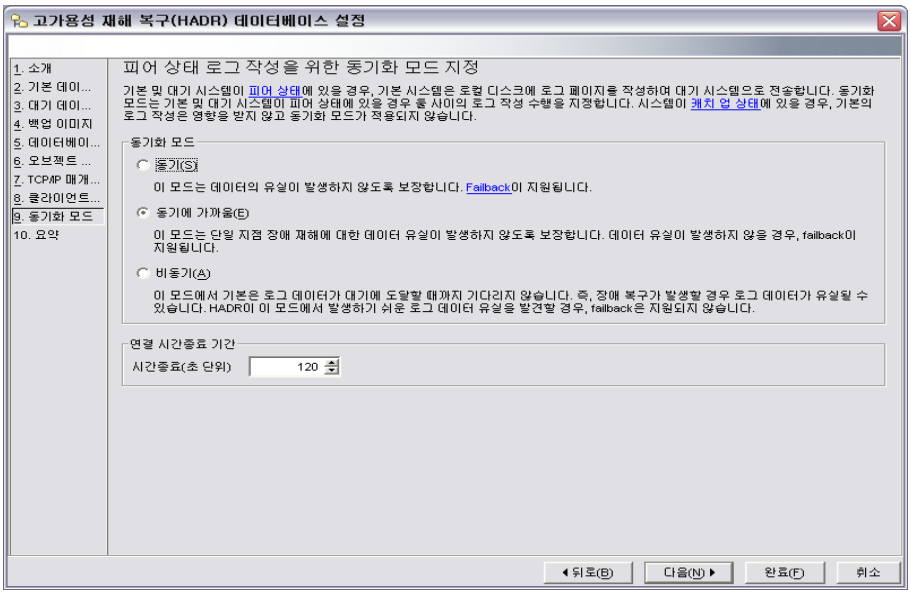


Figure 1505T ••• HADR 데이터베이스 설정 – 동기화 모드 지정

➲ 완료 를 누르면, HADR을 위한 구성을 시작합니다.

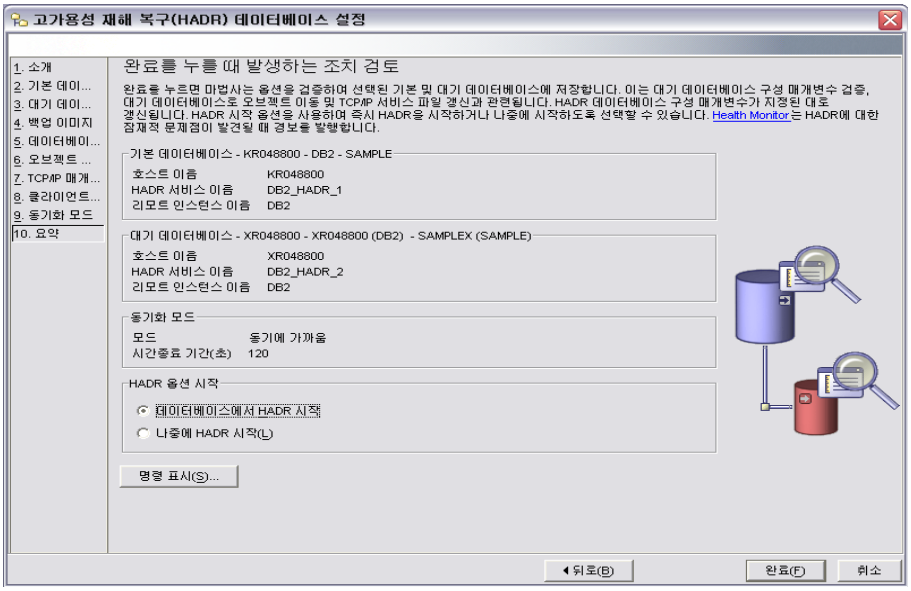


Figure 1505U ••• HADR 데이터베이스 설정 – 완료

TOPIC 1505 Wizard를 통한 HADR 구성

Point

Wizard를 통해 수행될 HADR 구성관련 명령을 확인할 수 있습니다.

➲ 명령표시를 통해 다음과 같은 내용을 확인할 수 있습니다.

```
-- 백업 이미지를 기본에서 대기 시스템으로 복사하십시오.
--
-- 기본 시스템의 위치 : P:\SRC
-- 대기 시스템의 위치 : C:\SRC
--
-- 대기 시스템에서 데이터베이스 리스토어 - XR048800 - XR048800 (DB2)  - SAMPLEX (SAMPLE)
--
RESTORE DATABASE SAMPLE FROM "C:\SRC"
       TAKEN AT 20060414144312 REPLACE HISTORY FILE WITHOUT PROMPTING
--
-- 클라이언트 리라우트를 위한 데이터베이스 구성 - KR048800 - DB2 - SAMPLE
--
UPDATE ALTERNATE SERVER FOR DATABASE SAMPLE USING HOSTNAME XR048800 PORT
      50000
--
-- 클라이언트 리라우트를 위한 데이터베이스 구성-XR048800-XR048800 (DB2)-SAMPLEX (SAMPLE)
--
UPDATE ALTERNATE SERVER FOR DATABASE SAMPLE USING HOSTNAME KR048800 PORT
      50000
--
-- 기본 시스템에서 서비스 파일 갱신 - KR048800
-- 서비스 이름 : DB2_HADR_1
-- 포트 번호 : 55001
-- 서비스 이름 : DB2_HADR_2
-- 포트 번호 : 55002
--
-- 대기 시스템에서 서비스 파일 갱신 - XR048800
-- 서비스 이름 : DB2_HADR_1
-- 포트 번호 : 55001
-- 서비스 이름 : DB2_HADR_2
-- 포트 번호 : 55002
--
-- 기본 데이터베이스에서 HADR 구성 매개변수 갱신 - KR048800 - DB2 - SAMPLE
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_HOST KR048800
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_SVC DB2_HADR_1
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_HOST XR048800
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_SVC DB2_HADR_2
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_INST DB2
UPDATE DB CFG FOR SAMPLE USING HADR_SYNCMODE NEARSYNC
UPDATE DB CFG FOR SAMPLE USING HADR_TIMEOUT 120
CONNECT TO SAMPLE
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS
UNQUIESCE DATABASE
CONNECT RESET
--
-- 대기 데이터베이스에서 HADR 구성 매개변수 갱신-XR048800-XR048800 (DB2)-SAMPLEX (SAMPLE)
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_HOST XR048800
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_SVC DB2_HADR_2
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_HOST KR048800
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_SVC DB2_HADR_1
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_INST DB2
UPDATE DB CFG FOR SAMPLE USING HADR_SYNCMODE NEARSYNC
UPDATE DB CFG FOR SAMPLE USING HADR_TIMEOUT 120
--
-- 대기 데이터베이스에서 HADR 시작 - XR048800 - XR048800 (DB2)  - SAMPLEX (SAMPLE)
DEACTIVATE DATABASE SAMPLE
START HADR ON DATABASE SAMPLE AS STANDBY
--
-- 기본 데이터베이스에서 HADR 시작 - KR048800 - DB2 - SAMPLE
DEACTIVATE DATABASE SAMPLE
START HADR ON DATABASE SAMPLE AS PRIMARY
```

TOPIC 1505 Wizard를 통한 HADR 구성

Point

Wizard를 통한 HADR 구성을 완료합니다.

➲ "고가용성 재해 복구(HADR) 데이터베이스 설정" 에서 "완료" 를 Click하면 다음과 같이 HADR 구성을 진행합니다.

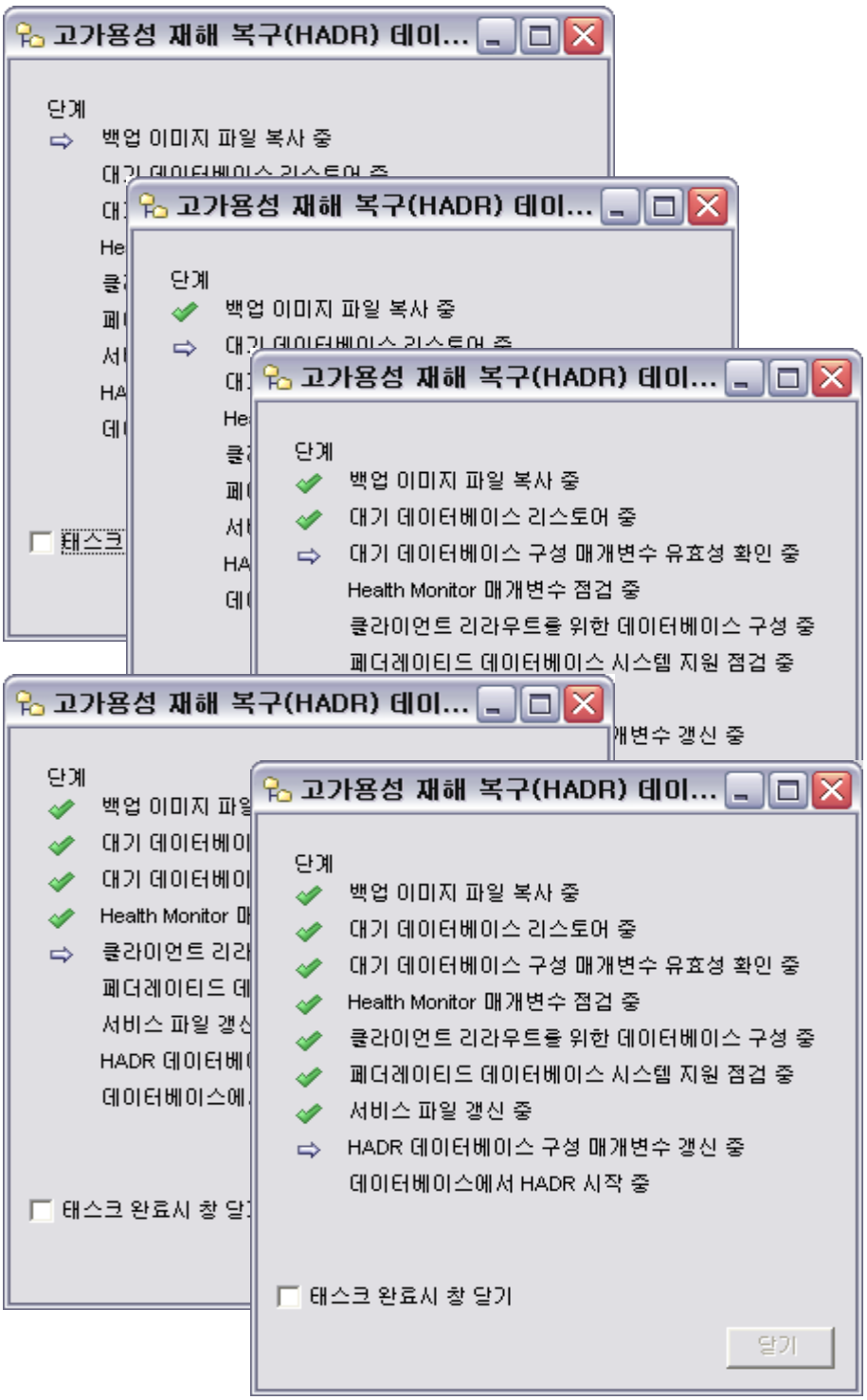


Figure 1505V ••• HADR 데이터베이스 설정 – 완료

TOPIC 1506 CLP를 통한 HADR 구성

Point

CLP (Command Line Process)를 통한 HADR 구성을 합니다.

1 CLP를 통한 HADR구성

두 대의 HADR Pair를 위해 다음과 같은 N/W IP 및 Service Port로 구성된 시스템을 대상으로 Command Line Process (CLP) 명령을 통해 구성할 수 있습니다.

KR048800	DB CFG Parameter	XR048800
192.1.1.10	HADR_LOCAL_HOST	192.1.1.20
55001	HADR_LOCAL_SVC	55002
192.1.1.20	HADR_REMOTE_HOST	192.1.1.10
55002	HADR_REMOTE_SVC	55001
DB2	HADR_REMOTE_INST	DB2
120	HADR_TIME_OUT	120
NEARSYNC	HADR_SYNCMODE	NEARSYNC
PRIMARY	HADR_DB_ROLE	STANDBY

Figure 1506A • • • N/W IP 및 Service Port 구성 내역

KR048800	작업내용	XR048800

CONNECT TO SAMPLE;
QUIESCE DB IMMEDIATE FORCE CONNECTIONS;
UNQUIESCE DATABASE;
CONNECT RESET;
UPDATE DB CFG FOR SAMPLE USING
  INDEXREC RESTART
  LOGINDEXBUILD ON
  LOGARCHMETH1
  "DISK:C:\DB2\NODE0000\SQL00001\SQLOGDIR2"
  LOGPRIMARY 3
  LOGSECOND 2
  LOGFILSIZ 1000;
BACKUP DATABASE SAMPLE TO "P:\SRC"
  WITH 2 BUFFERS
  BUFFER 1024
  PARALLELISM 1
  COMPRESS
  WITHOUT PROMPTING;
	Primary DB
	Archive Log을 위한
	DB CFG 변경	

	Backup 이미지
	복사 및 Restore
		- FTP 또는 COPY을 통한 Backup 이미지 복사
		- Database Restore
		RESTORE DATABASE SAMPLE
		  FROM "C:\SRC"
		  REPLACE HISTORY FILE
		  WITHOUT PROMPTING

TOPIC 15-06 CLP를 통한 HADR 구성

Point

CLP (Command Line Process)를 통한 HADR 구성을 확인합니다.

KR048800	작업내용	XR048800
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_HOST 192.1.1.10 HADR_LOCAL_SVC 55001 HADR_REMOTE_HOST 192.1.1.20 HADR_REMOTE_SVC 55002 HADR_REMOTE_INST DB2 HADR_SYNCMODE NEARSYNC HADR_TIMEOUT 120;	Primary DB HADR을 위한 DB CFG 변경	
	Standby DB HADR을 위한 DB CFG 변경	UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_HOST 192.1.1.20 HADR_LOCAL_SVC 55002 HADR_REMOTE_HOST 192.1.1.10 HADR_REMOTE_SVC 55001 HADR_REMOTE_INST DB2 HADR_SYNCMODE NEARSYNC HADR_TIMEOUT 120;
	Standby HADR Start	START HADR ON DB SAMPLE AS STANDBY;
START HADR ON DB SAMPLE AS PRIMARY;	Primary HADR Start	

HADR로 구성된 DB에 대한 구성을 다음과 같이 확인합니다.

DB	'db2 get db cfg for sample | grep HADR'
Primary DB CFG	HADR 데이터베이스 역할 = PRIMARY HADR 로컬 호스트 이름 (HADR_LOCAL_HOST) = 192.1.1.10 HADR 로컬 서비스 이름 (HADR_LOCAL_SVC) = 55001 HADR 리모트 호스트 이름 (HADR_REMOTE_HOST) = 192.1.1.20 HADR 리모트 서비스 이름 (HADR_REMOTE_SVC) = 55002 리모트 서버의 HADR 인스턴스 이름 (HADR_REMOTE_INST) = DB2 HADR 시간종료 값 (HADR_TIMEOUT) = 120 HADR 로그 쓰기 동기화 모드 (HADR_SYNCMODE) = NEARSYNC
Standby DB CFG	HADR 데이터베이스 역할 = STANDBY HADR 로컬 호스트 이름 (HADR_LOCAL_HOST) = 192.1.1.20 HADR 로컬 서비스 이름 (HADR_LOCAL_SVC) = 55002 HADR 리모트 호스트 이름 (HADR_REMOTE_HOST) = 192.1.1.10 HADR 리모트 서비스 이름 (HADR_REMOTE_SVC) = 55001 리모트 서버의 HADR 인스턴스 이름 (HADR_REMOTE_INST) = DB2 HADR 시간종료 값 (HADR_TIMEOUT) = 120 HADR 로그 쓰기 동기화 모드 (HADR_SYNCMODE) = NEARSYNC

TOPIC 1507 HADR Monitoring

Point

HADR 상태를 Monitoring하는 방법은 여러 가지가 있습니다. – db2pd 이용

Tip

State의 종류
- Local Catch Up
- Remote Catch Up Pending
- Remote Catch Up
- Peer
- Disconnected

Tip

Connection Status의 종류
- Connected
- Disconnected
- Congested

1 db2pd 이용

Primary DB

db2pd -db sample -hadr

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:52

HADR Information:
Role    State          SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
Primary Peer           Nearsync 0                  0

ConnectStatus ConnectTime                          Timeout
Connected     Sun Apr 16 18:47:32 2006 (1145180852) 120

LocalHost                                LocalService
192.1.1.10                               55001

RemoteHost                               RemoteService      RemoteInstance
192.1.1.20                               55002              DB2

PrimaryFile  PrimaryPg  PrimaryLSN
S0000002.LOG 0          0x0000000001770000

StandByFile  StandByPg  StandByLSN
S0000002.LOG 0          0x0000000001770000
```

Standby DB

db2pd -db sample -hadr

```
C:\SRC>db2pd -db sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 02:51:57

HADR Information:
Role    State          SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
Standby Peer           Nearsync 0                  0

ConnectStatus ConnectTime                          Timeout
Connected     Sun Apr 16 18:47:32 2006 (1145180852) 120

LocalHost                                LocalService
192.1.1.20                               55002

RemoteHost                               RemoteService      RemoteInstance
192.1.1.10                               55001              DB2

PrimaryFile  PrimaryPg  PrimaryLSN
S0000002.LOG 0          0x0000000001770000

StandByFile  StandByPg  StandByLSN
S0000002.LOG 0          0x0000000001770000
```

TOPIC 1507 HADR Monitoring

Point

HADR 상태를 Monitoring하는 방법은 여러 가지가 있습니다. - get snapshot 을 이용

2 db2 get snapshot for db ... 이용

Primary DB

db2 get snapshot for db on sample

```
HADR 상태
 역할                        = 1차
 상태                        = 피어
 동기화 모드                   = Nearsync
 연결 상태                    = 연결됨 , 2006-04-16 18:47:32.460644
 누락된 하트비트(Heartbeats)      = 0
 로컬 호스트                   = 192.1.1.10
 로컬 서비스                   = 55001
 리모트 호스트                  = 192.1.1.20
 리모트 서비스                  = 55002
 리모트 인스턴스                 = DB2
 시간종료(초)                  = 120
 1차 로그 위치(파일, 페이지, LSN)  = S0000002.LOG, 0, 0000000001770000
 대기 로그 위치(파일, 페이지, LSN) = S0000002.LOG, 0, 0000000001770000
 평균 실행 중 로그 갭(바이트)     = 0
```

Standby DB

db2 get snapshot for db on sample

```
HADR 상태
 역할                        = 대기
 상태                        = 피어
 동기화 모드                   = Nearsync
 연결 상태                    = 연결됨 , 2006-04-16 18:47:32.475740
 누락된 하트비트(Heartbeats)      = 0
 로컬 호스트                   = 192.1.1.20
 로컬 서비스                   = 55002
 리모트 호스트                  = 192.1.1.10
 리모트 서비스                  = 55001
 리모트 인스턴스                 = DB2
 시간종료(초)                  = 120
 1차 로그 위치(파일, 페이지, LSN)  = S0000002.LOG, 0, 0000000001770000
 대기 로그 위치(파일, 페이지, LSN) = S0000002.LOG, 0, 0000000001770000
 평균 실행 중 로그 갭(바이트)     = 0
```

TOPIC 1508 HADR Role 변경 - Takeover

Point

HADR을 운영하는 중에 “TAKEOVER ” 명령을 통해 Primary Role과 Standby Role을 변경할 수 있습니다.

Tip

USER … USING …
TAKEOVER를 수행하는 USER가 권한이 있는 경우에는 사용자ID와 비밀번호를 생략할 수 있습니다.

1 TAKEOVER

“ TAKEOVER HADR ON DB sample USER db2admin USING db2admin ”

➲ Primary DB 와 Standby DB 가 서로 “PEER ” 상태에서 H/W, S/W, 그 외 여러 정비작업을 이유로 DB의 Role을 변경할 수 있습니다.

➲ Standby DB에서 TAKEOVER명령을 사용하여 Primary Role을 획득하여 새로운 Primary DB로서의 역할을 합니다.

➲ 이전 Primary DB에서는 정비작업을 수행할 수 있습니다.

➲ 작업 이후, Primary DB를 새로운 Standby DB로 Start한 후, Log 변경사항에 대한 Catch-Up을 수행합니다.

➲ Catch-Up이 종료되고, “ PEER ”상태가 되면, TAKEOVER 명령을 사용하여 새로운 Primary로서의 역할을 수행 합니다.

2 TAKEOVER ~ BY FORCE

“ TAKEOVER HADR ON DB sample USER db2admin USING db2admin BY FORCE ”

➲ H/W, S/W, N/W 및 Storage 장애로 인해, Primary DB가 정상적인 작업을 수행할 수 없고, Primary와 Standby DB사이에 서로 통신이 되지 않은 상황에서 Standby DB에게 강제적으로 Primary DB로서의 역할을 수행하게 할 때 사용합니다.

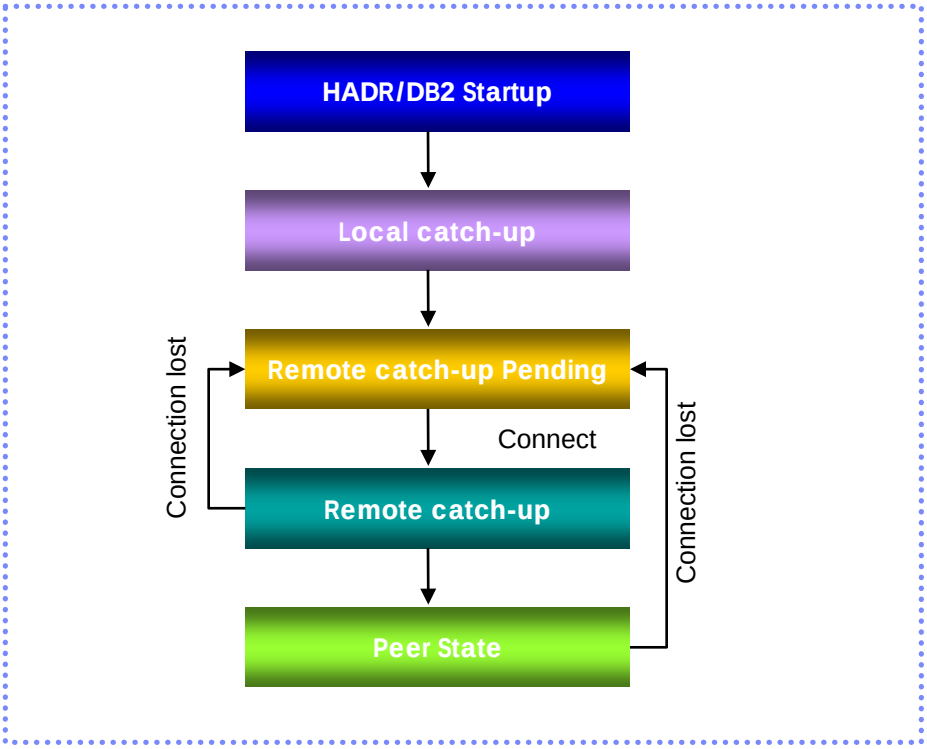


Figure 1508A HADR status의 변화

TOPIC 1509 Automatic Client Reroute

Point

Automatic Client Reroute 를 통해 사용자 application connection을 자동으로 대체서버로 전환해 수행할 수 있습니다.

1 Automatic Client Reroute

Primary Database의 장애로 인해, USER Application이 더 이상 서비스 할 수 없는 상황이 되면, 사용자 Application Connection을 자동으로 대체서버로 전환하여 Application을 계속 수행 할 수 있게 합니다.

- Client Reroute를 위해 별도로 Application을 수정할 필요는 없습니다.
- Client Run-Time Library에서 해당 REROUTE를 수행합니다.

대체서버 지정

Primary Database

```
UPDATE ALTERNATE SERVER FOR DATABASE SAMPLE USING
    HOSTNAME 192.1.1.20 PORT 50000
```

Standby Database

```
UPDATE ALTERNATE SERVER FOR DATABASE SAMPLE USING
    HOSTNAME 192.1.1.10 PORT 50000
```

Primary DB

db2 list db directory

```
데이터베이스 별명                = SAMPLE
데이터베이스 이름                = SAMPLE
데이터베이스 드라이브             = C:\DB2
데이터베이스 릴리스 레벨          = a.00
주석                        =
디렉토리 항목 유형               = 간접
카탈로그 데이터베이스 파티션 번호   = 0
대체 서버 호스트 이름            = 192.1.1.20
대체 서버 포트 번호              = 50000
```

Standby DB

db2 list db directory

```
데이터베이스 별명                = SAMPLE
데이터베이스 이름                = SAMPLE
데이터베이스 드라이브             = C:\DB2
데이터베이스 릴리스 레벨          = a.00
주석                        =
디렉토리 항목 유형               = 간접
카탈로그 데이터베이스 파티션 번호   = 0
대체 서버 호스트 이름            = 192.1.1.10
대체 서버 포트 번호              = 50000
```

TOPIC 1509 Automatic Client Reroute

Point

Automatic Client Reroute 설정 후 Application에서의 장애처리 예 입니다.

Tip

SQLCODE : 30081 / 30108
Communication 장애 또는 제 접속 이후에 발생하는 SQLCODE 입니다. Application에서 해당 SQLCODE에 따른 장애 처리를 하여, 사용자에게는 장애에 투명하게 업무를 수행 할 수 있게 합니다.

Tip

장애 처리가 없으면, 사용자는 해당 Transaction 처리에 대한 오류를 Return 받게 되며, Transaction 재 요청을 하게 되면 대체 서버에서 Transaction을 재처리하게 됩니다.

Tip

Alternate Server 정보
Client가 DB 에 접속하는 시점에, Alternate Server에 대한 정보가 Client Runtime 환경에 자동으로 저장됩니다.

2 Automatic Client Reroute

➲ Application에서의 장애 처리

User Application을 수행 하는 중,
1. Database와 Connection이 단절된 경우
2. 단절된 Database Connection이 재 연결된 경우

SQL을 재 수행 할 수 있습니다.
이로 인해, 사용자는 DB의 장애를 감지 못하고 지속적인 업무 수행이 가능합니다.

Application에서의 장애처리 예

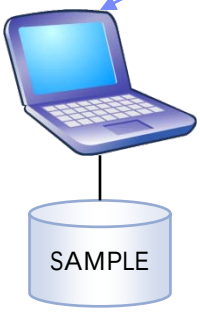
```
exec sql ....
if (sqlca->sqlcode == -30081)
{
   // Communication 장애에 따른 Error Code
   printf(...);
   // SQL 재 처리 Routine 또는 별도의 Routine
   ....;
}
else if (sqlca->sqlcode == -30108)
{
   // Communication이 다시 연결된 후 Error Code
   printf(...);
   // SQL 재 처리 Routine 또는 별도의 Routine
   ....;
}
```

접속 시, Client Library에 대체서버에 대한 정보 저장 됨.

hostname <hhh>
port <nnn>

장애 발생시, 자동으로 대체 Server에 재 연결

KR048800



KR048800	DB CFG Parameter	XR048800
192.1.1.10	HADR_LOCAL_HOST	192.1.1.20
55001	HADR_LOCAL_SVC	55002
192.1.1.20	HADR_REMOTE_HOST	192.1.1.10
55002	HADR_REMOTE_SVC	55001
192.1.1.20 50000	Alternate Server & Port	192.1.1.10 50000

XR048800

TOPIC 15-10 HADR 튜닝 Parameter

Point

HADR 성능을 최적화하기 위해 제공되는 Parameter를 이용하여 적절한 튜닝을 합니다.

Tip

HADR은 Public 네트웍이 아닌 전용 네트웍으로 상호 연결되는 것이 성능에 좋습니다.

1 통신 Parameter

변수	설명
tcp_nodealyack (HADR 전용 통신 네트웍)	권장 값 1, 디폴트 0

2 메모리 Parameter

Tip

db2_hadr_buf_size는 Sync, Nearsync모드에서 성능 개선에 많은 효과가 있습니다.

변수	설명
DB2_HADR_BUF_SIZE (db2 registry 변수)	Standby서버에서 로그 수신 버퍼 크기, 디폴트 Primary Log Buffer(logbufsz)의 2배
db2_hadr_sosndbuf (db2 registry 변수)	HADR연결에 대한 TCP/IP 송신 버퍼 값, 디폴트 OS통신 설정 값
db2_hadr_sorcvbuf (db2 registry 변수)	HADR연결에 대한 TCP/IP 수신 버퍼 값, 디폴트 OS통신 설정 값

3 프로세스 Parameter

Standby에서 Rollforward 하는 동시 프로세스 수를 결정하며, Default는 CPU수이며, AIX시스템의 경우 SMT Enable되어 있을 경우, CPU수 *2 만큼 프로세스를 생성합니다. 이때 프로세스간 경합이 발생되어 오히려 HADR성능이 저하되는 경우가 발생하고, 또한 Standby의 CPU 를 과도하게 사용하는 경우가 발생됩니다.

CPU 수가 12일 경우

```
$cat /db2/bpvars.cfg
Prec_num_agents=12


db2set db2bpvars = /db2/bpvars.cfg
```

4 클라이언트 Parameter

변수	설명
db2_max_client_connretries (db2 registry 변수)	자동 클라이언트 리라우트에 대한 최대 재연결 시도 수
db2_connretries_interval (db2 registry 변수)	자동 클라이언트 리라우트에 대한 연속 재연결 시도에 대한 간격 시간(초)
db2tcp_client_contimeout (db2 registry 변수)	TCP/IP 연결 시 클라이언트가 대기하는 시간(초)
db2tcp_client_rcvtimeout (db2 registry 변수)	TCP/IP 수신 조작 시 클라이언트가 대기하는 시간(초)

UNIT 16

아키텍처

DB2의 아키텍처, 프로세스 모델, 메모리 모델에 대해 소개합니다. 무공유 아키텍처를 지원하므로 다중 서버에 다중 데이터베이스 파티션을 구축하면, 성능과 확장성이 좋은 병렬 처리 환경의 데이터베이스를 운영할 수 있습니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 아키텍처 개요
- 단일 데이터베이스 파티션 아키텍처
- 다중 데이터베이스 파티션 아키텍처
- 데이터베이스 시스템
- 단일 데이터베이스 파티션의 프로세스 모델
- 다중 데이터베이스 파티션의 프로세스 모델
- 인스턴스 수준의 프로세스
- 데이터베이스 수준의 프로세스
- 응용프로그램 수준의 프로세스
- 메모리 모델
- 인스턴스 공유 메모리
- 데이터베이스 공유 메모리
- 응용프로그램 공유 메모리
- 응용프로그램 개별 메모리
- 스레드 모니터링
- 메모리 사용량 모니터링

TOPIC 1601 아키텍처 개요

Point

DB2는 다중 스레드 기반의 안정적이고 고성능을 추구하는 아키텍처에 기반하고 있습니다.

Tip

9.1까지 프로세스 기반의 모델이며, 9.5이후에는 스레드기반의 프로세스 모델로 변경되었습니다. 이는 자원의 효율적 사용과 동시에 성능을 보다 향상시킵니다.

Tip

64비트를 지원하므로 대용량의 버퍼 풀을 생성할 수 있고, 정렬 작업을 원활하게 수행할 수 있습니다.

Tip

*프리페처*는 디스크에서 데이터를 검색하고 응용프로그램에 데이터가 필요하기 전에 버퍼 풀로 이동합니다.

Tip

*페이지 클리너*는 버퍼 풀에서 디스크로 다시 데이터를 이동합니다. 페이지 클리너는 응용프로그램 에이전트와 관계가 없는 백그라운드 EDU입니다.

Tip

데이터베이스 서버에서 접속을 요청하는 응용프로그램을 지역 응용프로그램이라고 하고, 클라이언트에서 요청하는 응용프로그램을 원격 응용프로그램이라고 합니다.

1 DB2 9.5이전 DB2프로세스 모델은 아래와 같습니다.

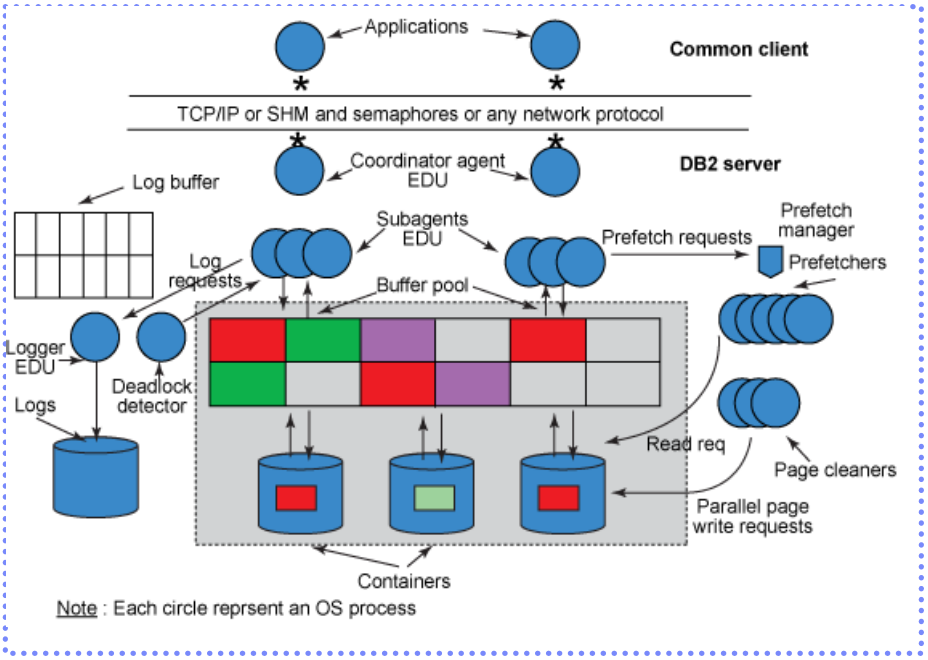


Figure 1601A ••• DB2 9.5 이전 프로세스 모델

2 DB2 9.5부터 다중 스레드 기반의 모델로 변경되었습니다.

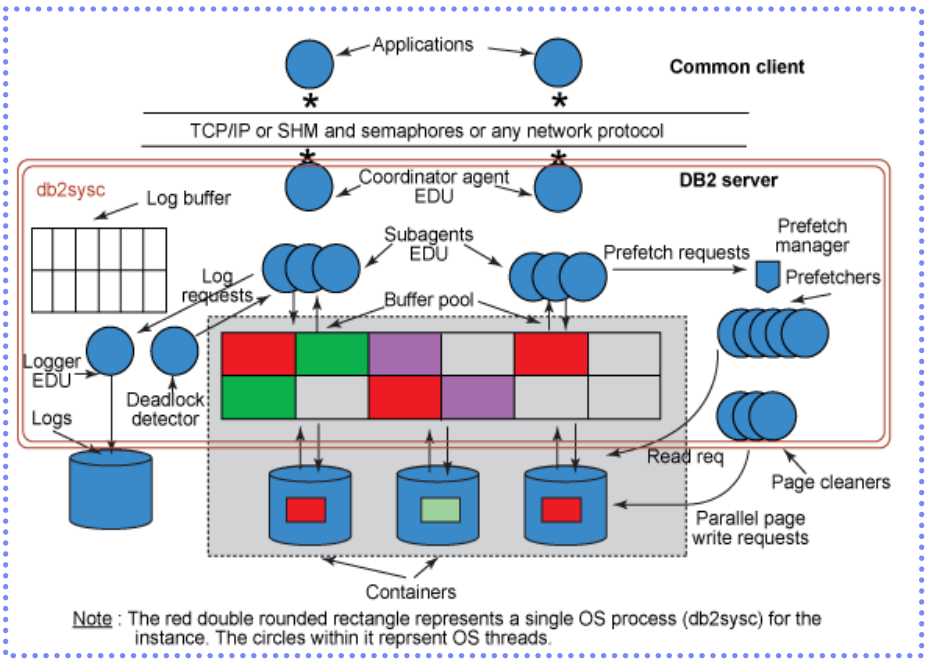


Figure 1601B ••• DB2 9.5 이후 새로운 프로세스 모델

TOPIC 16-02 단일 데이터베이스 파티션의 아키텍처

Point

단일 데이터베이스 파티션 환경에서 접속을 요청한 응용프로그램은 에이전트 프로세스를 통하여 엔진에게 SQL문의 처리를 요청할 수 있습니다. 파티션내 병렬 기능을 이용하면, SQL문을 병렬로 처리할 수 있습니다.

Tip

SMP 머신에서 파티션내 병렬 기능을 이용하면, 여러 개의 Subagent를 이용하여 쿼리를 병렬로 처리할 수 있습니다.

Tip

에이전트 프로세스의 생성과 제거로 인한 오버헤드를 줄이기 위해 에이전트 프로세스를 위한 POOL 을 운영할 수 있습니다.

Tip

데이터가 여러 디스크에 걸쳐 스트라이핑이 되어있다면, 여러 개의 프리페처를 지정하여 여러 디스크를 동시에 액세스할 수 있도록 합니다.

1 단일 데이터베이스 파티션으로 구성된 인스턴스의 아키텍처는 다음과 같습니다.

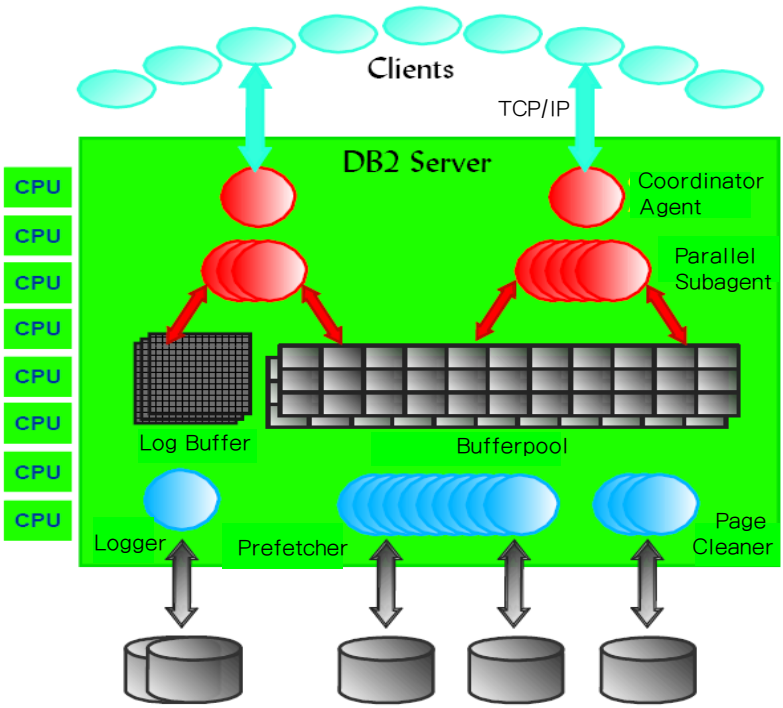


Figure 1602A ••• 단일 데이터베이스 파티션의 아키텍처

2 DB2 엔진은 고유한 기능을 담당하는 여러 개의 EDU(Engine Dispatchable Unit)로 구성되어 있으며, 각 EDU는 UNIX에서는 스레드로 구현됩니다.

3 응용프로그램이 데이터베이스에 접속을 요청하면, 전용 Coordinator Agent 프로세스가 생성되어 응용프로그램을 대신하여 데이터베이스에게 필요한 SQL문의 실행을 요청하는 역할을 하게 됩니다. 클라이언트와 서버 사이의 통신에는 TCP/IP, APPC, Netbios 등의 프로토콜이 지원됩니다.

4 에이전트 프로세스가 필요한 데이터를 가져오기 위하여 프리페치 큐를 통해 비동기 방식의 읽기 요청을 하면, 프리페처 프로세스는 big-block I/O를 통해 요청된 페이지들을 버퍼풀로 미리 가져오므로 에이전트 프로세스는 디스크 I/O 대기 시간을 줄일 수 있습니다.

5 페이지 클리너는 백그라운드 EDU로 특정 상황이 될 때마다 버퍼풀의 Dirty Page를 디스크로 미리 반영하여 버퍼풀의 가용 공간을 확보함으로써 버퍼풀을 사용하는 에이전트 프로세스가 대기하는 일이 없도록 합니다.

6 변경된 데이터 페이지와 로그 레코드는 성능을 위해 디스크로 즉시 반영되지 않습니다. 변경된 데이터를 디스크에 반영할 때는 반드시 로그 파일에 먼저 기록됩니다. 이러한 방식을 Write Ahead Logging이라고 합니다.

TOPIC 16 03 다중 데이터베이스 파티션의 아키텍처

Point

다중 데이터베이스 파티션 환경에서 접속을 요청한 응용프로그램은 코디네이터 에이전트와 파티션별 서브에이전트를 통하여 엔진에게 SQL문의 처리를 요청합니다. 파티션간 병렬 처리가 적용되므로 SQL문을 병렬로 처리할 수 있습니다.

Tip

다중 데이터베이스 파티션 환경에서는 파티션간 병렬 처리가 자동적으로 적용되며, 파티션내 병렬 처리 기능과 함께 사용할 수도 있습니다.

1 다중 데이터베이스 파티션으로 구성된 인스턴스의 아키텍처는 다음과 같습니다.

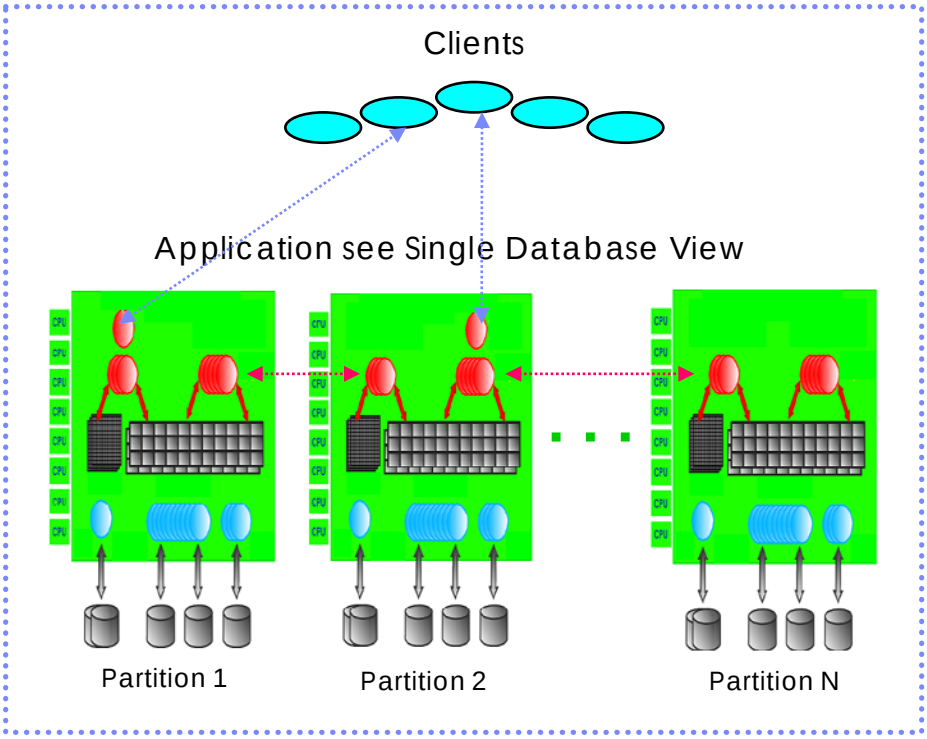


Figure 1603A ••• 다중 데이터베이스 파티션의 아키텍처

2 DB2 UDB의 Data Partitioning Feature는 다중 데이터베이스 파티션 기능을 이용하여 병렬 데이터베이스를 구축합니다. 다중 데이터베이스 파티션에 생성된 데이터베이스는 사용자에게는 한 개의 논리적인 데이터베이스로 인식되지만, 각 파티션에 물리적으로 데이터베이스를 생성합니다. 파티션별로 생성된 데이터베이스는 일반적인 단일 데이터베이스와 동일하게 독립적인 자원을 사용할 수 있으므로 각 파티션은 버퍼풀, 잠금 관리, 디스크 등을 독립적으로 운영하게 됩니다.

Tip

한 개의 서버에 기본적으로 4개의 논리적 파티션이 생성될 수 있으며, 필요시 4개 이상도 생성할 수 있습니다.

3 데이터베이스 파티션은 동일한 머신에 생성되는 논리적 파티션과 다른 머신에 생성되는 물리적 파티션으로 구분됩니다. 동일한 머신에 존재하는 데이터베이스 파티션들은 공유 메모리를 이용하여 통신하고, 다른 머신에 존재하는 데이터베이스 파티션끼리는 고속의 네트웍을 통해서 필요한 부분만 통신합니다.

Tip

SQL문을 처리하는 각 Subagent들의 실행은 모든 파티션을 통하여 병렬로 처리됩니다. DB2가 제공하는 여러 가지 유틸리티들도 모두 병렬로 처리됩니다.

4 응용프로그램이 접속을 요청하면, 접속한 데이터베이스 파티션에 Coordinator Agent 프로세스가 생성되고, 다른 파티션에는 Subagent 프로세스들이 생성됩니다. SQL문은 Global optimizer에 의한 최적화된 후에 각 Subagent들에게 전송됩니다. Subagent가 해당 파티션의 데이터베이스에 대해서만 SQL문의 요청을 처리하여 결과를 Coordinator Agent 프로세스에게 반환하면, Coordinator Agent는 응용프로그램에게 최종 결과를 반환합니다.

5 MPP 머신으로 구축한 DB2의 다중 데이터베이스 파티션 환경은 완전한 Shared Nothing Architecture를 제공하므로 확장성이 좋습니다.

TOPIC 1604 데이터베이스 시스템

Point

DB2 UDB 데이터베이스 시스템은 엔진, 인스턴스, 데이터베이스로 구성됩니다. DB2 엔진은 서버 머신에 설치된 실제적인 제품 모듈입니다. 인스턴스는 DB2 엔진을 사용하기 위한 논리적인 환경이며, 데이터베이스는 실제적인 데이터를 저장합니다.

1 DB2 UDB 데이터베이스 시스템은 엔진, 인스턴스, 데이터베이스로 구성됩니다. 엔진, 인스턴스, 데이터베이스, 테이블스페이스 컨테이너 등은 물리적으로 분리되어 있으므로 제품의 재설치, 인스턴스의 재생성시에도 데이터베이스와 관련된 물리적인 파일은 보존되고, 필요시에 다시 사용할 수 있습니다.

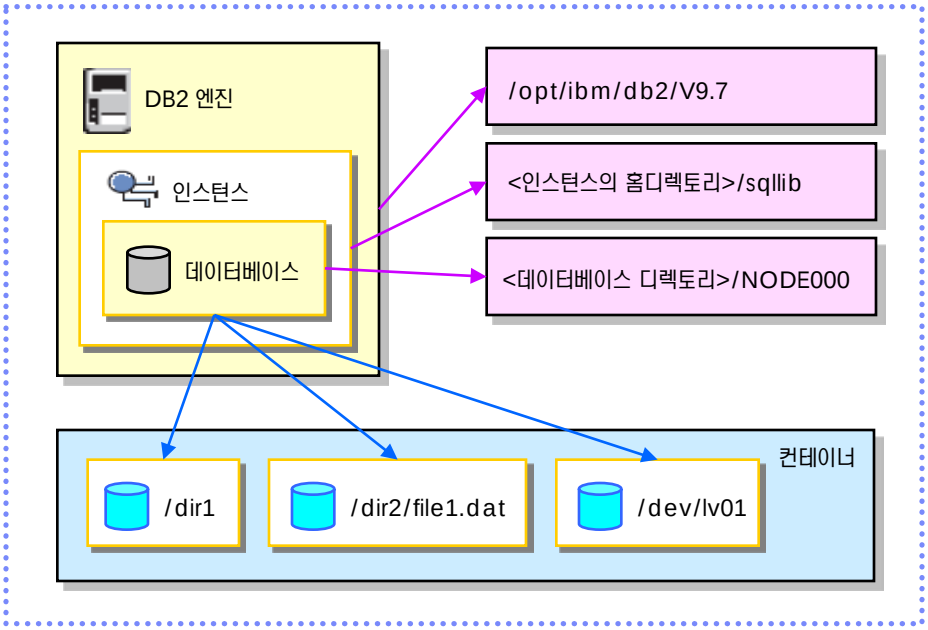


Figure 1604A ••• 데이터베이스 시스템과 대응되는 디렉토리

Tip

V9부터 설치 경로는 임의로 바꿀 수 있습니다. Default경로는 /opt/ibm/db2/V9.7입니다.

2 DB2 UDB 제품 모듈은 AIX 머신에서는 default 경로인 /opt/ibm/db2/V9.7 디렉토리에 설치되고, Windows 머신에서는 원하는 디렉토리에 설치됩니다. 한 개의 서버 머신에서 서로 다른 디렉토리에 각각의 다른 버전의 제품의 설치가 가능합니다.

Tip

인스턴스를 생성할 때, Windows 머신에서는 사용자 계정이 필요하지 않습니다. 또한, 설치시에 DB2 라는 이름으로 기본적인 인스턴스가 생성됩니다.

3 한 개의 서버 머신에서 여러 개의 인스턴스를 생성할 수 있습니다. AIX 머신에서는 인스턴스마다 사용자 계정이 필요합니다.

4 root 사용자가 db2icrt 명령어를 이용하여 인스턴스를 생성하면, 인스턴스 사용자 계정의 홈 디렉토리에 sqllib 라는 서브디렉토리가 생성됩니다. sqllib에 생성된 일부 디렉토리와 파일은 제품이 설치된 /opt/ibm/db2/v9.7 디렉토리의 일부 서브디렉토리와 심볼릭 링크(symbolic link)로 연결되거나 복사됩니다.

5 인스턴스 사용자가 db2start 명령어를 이용하여 인스턴스를 기동시키면, sqllib에 존재하는 파일들을 이용하여 인스턴스와 관련된 여러 개의 엔진 프로세스가 생성되고, 인스턴스 전역 메모리가 할당됩니다.

Tip

다중 파티션 환경인 경우에는 한 개의 논리적인 데이터베이스를 생성하면, 데이터베이스 파티션 개수만큼의 물리적인 데이터베이스가 생성됩니다.

6 인스턴스 사용자가 create database 명령어를 이용하여 데이터베이스를 생성하면, 특정 디렉토리에 데이터베이스 구성 파일을 비롯한 물리적인 디렉토리와 파일들이 생성됩니다.

7 데이터베이스에 접속한 후에 create tablespace 명령어를 이용하여 테이블스페이스와 컨테이너를 지정하면, 실제 데이터를 물리적으로 독립적인 공간에 저장할 수 있습니다.

TOPIC 1605 단일 데이터베이스 파티션의 프로세스 모델

Point

단일 데이터베이스 파티션 환경에서 DB2 UDB의 프로세스는 인스턴스 수준, 데이터베이스 수준, 응용프로그램 수준으로 구분됩니다. 수준별 프로세스는 인스턴스 기동, 데이터베이스 활성화, 응용프로그램 접속시에 생성됩니다.

1 단일 데이터베이스 파티션으로 구성된 인스턴스에서는 다음과 같은 프로세스가 생성됩니다.

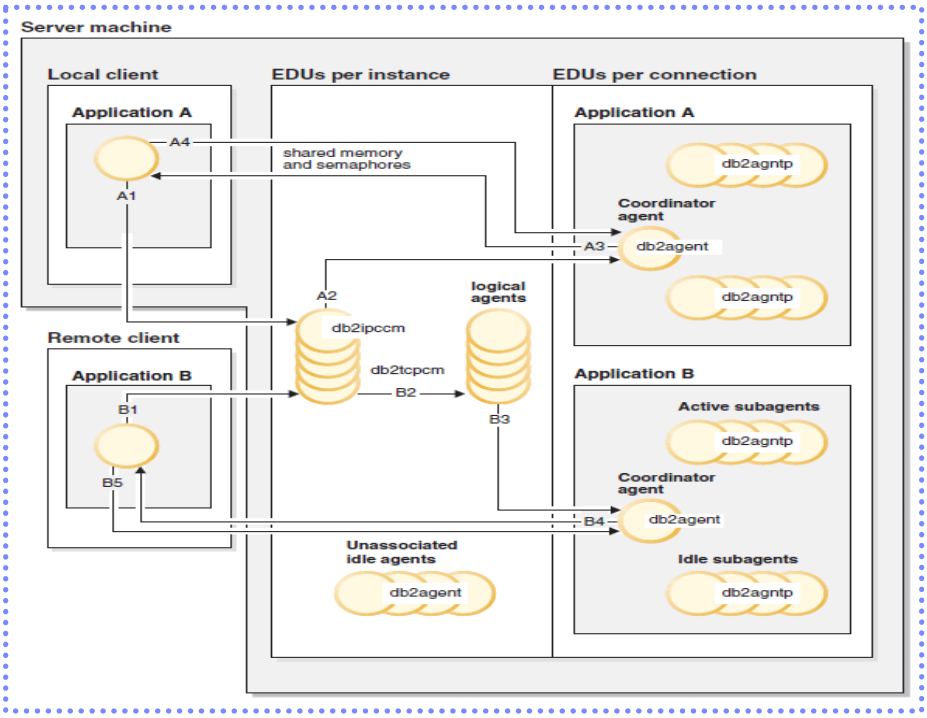


Figure 1605A ••• 단일 데이터베이스 파티션의 프로세스 모델

2 단일 데이터베이스 파티션 환경에서 DB2 프로세스는 인스턴스 레벨 에서만 보여진다.

수 준	설 명
인스턴스	인스턴스 수준의 프로세스는 db2start 명령을 실행하여 인스턴스를 기동할 때 생성되며 db2fmcd, db2wdog, db2sysc, db2ckpwd, db2acd, db2fmd 가 있습니다. 그리고 동시에 db2sysc, db2tcpcm, db2pfchr, db2pclnr 등의 EDUs가 실행 됩니다.
데이터베이스	데이터베이스 수준에서는 EDUs로 보여지며 activate db 명령어에 의해 데이터베이스가 활성화될 때 생성되고, deactivate db 명령어에 의해 데이터베이스가 비활성화되면 제거됩니다. db2evmgi, db2fw0, db2wlmd, db2pfchr, db2pclnr, db2dlock, db2lfr, db2loggw, db2loggr, db2taskd, db2stmm 등의 EDUs가 있습니다.
응용프로그램	응용프로그램 수준에서도 EDUs로 보여지며 지역 또는 원격 응용프로그램이 데이터베이스에 접속을 요청하는 경우에 생성됩니다. db2agent, db2agntp 등의 EDUs가 있습니다.

3 데이터베이스 엔진과 관련된 프로세스들은 방화벽을 통해 외부의 응용프로그램 프로세스들과 다른 address space를 사용하도록 구성되어 있으므로, 데이터베이스 제어 블록 및 중요한 데이터베이스 파일과 분리되도록 설계되어 있습니다.

Tip

최초의 connect 요청은 데이터베이스를 간접적으로 활성화시키고, 최종의 connect reset 요청은 데이터베이스를 비활성화시킵니다.

Tip

데이터베이스를 액세스하는 응용프로그램은 반드시 데이터베이스에 접속하므로, connection과 응용프로그램은 동일한 의미로 사용됩니다.

Tip

특정 명령어나 유틸리티를 실행하기 위하여 attach 명령어를 이용하여 인스턴스에 접속하는 경우에도 응용프로그램 수준의 프로세스가 생성됩니다.

Tip

db2agentg 프로세스는 DB2 Connect 제품으로 호스트에 접속하거나, 특정한 인스턴스를 게이트웨이로하여 다른 인스턴스의 데이터베이스에 접속을 요청할 때 생성됩니다.

TOPIC 16 06 다중 데이터베이스 파티션의 프로세스 모델

Point

다중 데이터베이스 파티션 환경에서 DB2 UDB의 프로세스는 인스턴스 수준, 카탈로그 파티션 수준, 데이터베이스 파티션 수준, 응용프로그램 수준으로 구분됩니다. 수준별 프로세스는 인스턴스 기동, 데이터베이스 활성화, 응용프로그램 접속시에 생성됩니다.

Tip

다중 데이터베이스 파티션을 구성하려면 DB2 UDB ESE 제품에 DPF 옵션이 필요합니다.

1 다중 데이터베이스 파티션으로 구성된 인스턴스에서는 다음과 같은 프로세스가 생성됩니다.

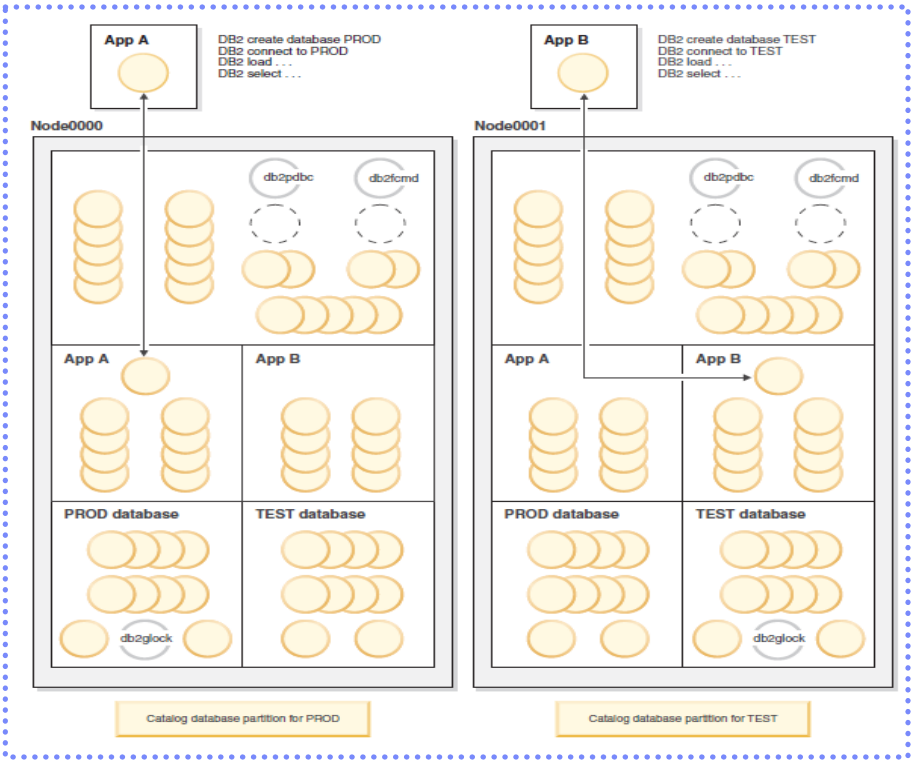


Figure 1606A ••• 다중 데이터베이스 파티션의 프로세스 모델

Tip

db2pdbc, db2panic 등은 다중 파티션 환경에서만 존재하는 프로세스입니다.

Tip

db2glock EDU는 카탈로그 파티션에만 존재합니다.

2 다중 데이터베이스 파티션 환경에서 DB2 프로세스는 인스턴스, 카탈로그 파티션, 데이터베이스 파티션, 응용프로그램의 4가지 수준으로 분리됩니다.

수 준	설 명
인스턴스	단일 데이터베이스 파티션 환경과 동일한 시점에 생성되고 제거됩니다. db2pdbc 와 db2fcmd 등의 EDUs가 추가됩니다.
카탈로그 파티션	단일 데이터베이스 파티션 환경과 동일한 시점에 생성되고 제거됩니다. db2glock EDUs가 추가됩니다.
데이터베이스 파티션	단일 데이터베이스 파티션 환경과 동일한 시점에 생성되고 제거됩니다.
응용프로그램	단일 데이터베이스 파티션 환경과 동일한 시점에 생성되고 제거됩니다.

TOPIC 1607 인스턴스 수준의 프로세스

Point

인스턴스 수준의 프로세스는 db2start 명령으로 인스턴스를 기동할 때 생성됩니다. db2fmcd, db2wdog, db2sysc, db2ckpwd, db2acd, db2fmd 등의 프로세스가 있습니다.

Tip

※DB2 V9.5 부터는 프로세스 방식이 아닌 쓰레드 방식으로 운영되므로 유닉스의 $ ps -ef 명령으로는 실행중인 EDU(Engine Dispatchable Unit) 의 확인이 불가능합니다.

※DB2 V9.5 이후 제품에서는 $ db2pd -edus 로 확인합니다.

Tip

특정 인스턴스를 위한 엔진 프로세스들이 문제가 없는지를 확인하려면, ps 명령어로 db2sysc 프로세스를 확인하면 됩니다.

Tip

응용프로그램의 재접속이 많아지면, db2agent 프로세스의 생성이 빈번해지고, db2gds가 db2agent를 생성하고 삭제하기 위한 kernel CPU의 사용량이 증가합니다.

1 인스턴스 수준의 프로세스에는 db2fmcd, db2wdog, db2sysc, db2ckpwd, db2acd, db2fmd 등이 있습니다. 인스턴스 사용자가 db2start를 실행한 후에 ps 명령어를 이용하여 확인할 수 있습니다.

```
$ login <인스턴스 사용자>
$ db2start
$ ps –ef | grep -i <인스턴스명>
```

2 인스턴스가 기동될 때, 최초로 생성되는 기본 프로세스는 다음과 같습니다.

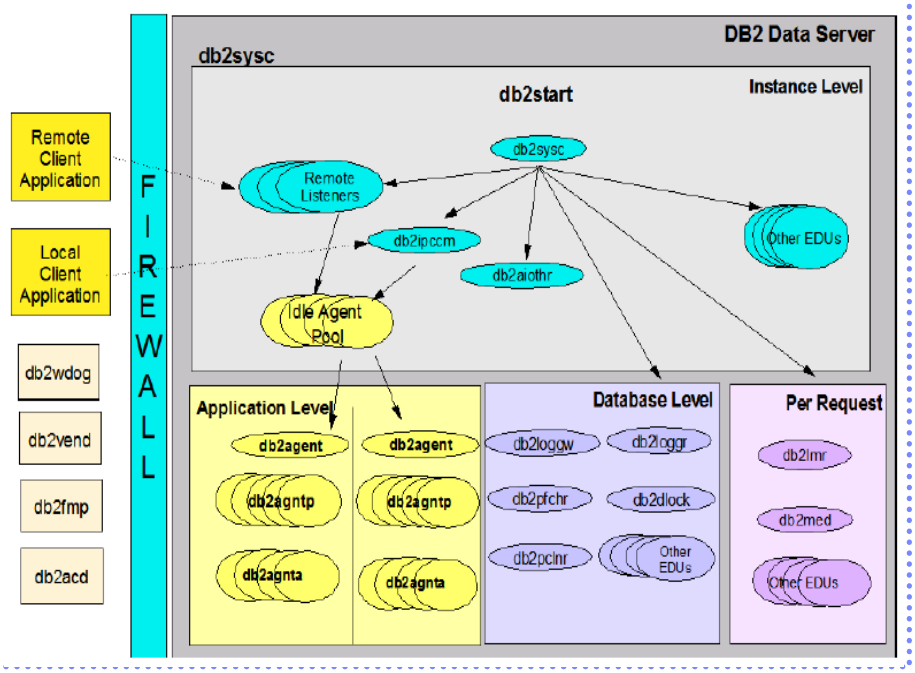


Figure 1607A DB2 프로세스 모델

프로세스 이름	설 명
db2sysc (Linux) db2syscs (Win)	DB2 9.5 이상의 시스템에서 db2start 명령과 동시에 발생되는 프로세스이다. 이 한 개의 프로세스로 모든 파티션의 thread를 처리 하도록 multi-thread로 되어 있으며, 여기에는 모든 Engine Dispatchable Units (EDUs) 이 thread로 구성되어 있다. 따라서 이 프로세스 없이는 데이터베이스가 실행될 수 없다.
db2acd	Health Monitor, 자동 유지보수 유틸리티 및 관리 태스크 스케줄을 관장하는 autonomic computing daemon 이다. db2hmon 에서 db2ace로 바뀐 것이다.
db2wdog	UNIX, LINUX 운영 체제에서 비정상 종료를 처리를 감시한다.
db2vend	EDUS에서 처리 할 수 없는 3rd party vendor 의 응용프로그램을 처리한다. Ex) userexit
db2fmp	stored procedures 또는 user defined functions(UDF) 와 같이 DB2의 외부에서 실행되는 코드를 처리한다. db2fmp 프로세스는 항상 별도의 프로세스이지만 실행하는 루틴의 유형에 따라 멀티스레드일 수 있다. DB2 8.7 이하에서의 db2udf 와 db2dari 프로세스가 db2fmp 프로세스로 대체되었다.

TOPIC 1607 인스턴스 수준의 프로세스

Point

인스턴스 수준의 프로세스는 db2start 명령으로 인스턴스를 기동할 때 생성됩니다. db2fmcd, db2wdog, db2sysc, db2ckpwd, db2acd, db2fmd 등의 프로세스가 있습니다.

Tip

특정한 인스턴스에서 지원할 프로토콜의 유형은 db2set 명령어를 이용하여 DB2COMM 이라는 레지스터리 변수에 한 가지 이상을 지정할 수 있습니다.

Tip

DB2COMM 레지스터리 변수와 함께 각 프로토콜별로 필요한 구성 변수를 적절히 설정하지 않으면, db2start 명령어로 인스턴스를 기동할 때, 해당하는 통신 리스너 프로세스를 생성할 수 없습니다.

Tip

DISCOVERY 기능은 클라이언트 머신에서 GUI 도구인 구성 지원 프로그램을 이용하여 요청할 수 있으면, DAS가 시작되어 있어야 합니다.

3 지역 또는 원격에서 데이터베이스에 대한 접속을 요청하면, 클라이언트와 서버 머신 사이에 사용되는 통신 프로토콜의 유형에 따라 다음과 같은 통신 리스너 스레드가 db2agent 라는 EDU를 생성하여 응용프로그램과 엔진을 연결시켜 줍니다.

EDU 명	생성시점	설명
db2ipccm	db2start 실행시	DB2 IPC Communication Manager 입니다. 데이터베이스 서버 머신에서 지역적으로 접속을 요청하는 응용프로그램을 지원하는 통신 리스너 EDU입니다. 한 개의 지역 응용프로그램으로 데이터베이스의 접속을 요청받으면, db2agent EDU를 생성하여 해당 응용프로그램 프로세스와 생성한 db2agent EDU를 연결합니다. 연결이 완료되면, db2agent EDU가 지역 응용프로그램 프로세스와 엔진을 연결하는 역할을 하게 되고, db2ipccm은 새로운 접속 요청을 대기하게 됩니다.
db2tcpcm	db2start 실행시	DB2 TCP Communication Manager 입니다. TCP/IP 프로토콜을 이용하여 접속을 요청하는 원격 응용프로그램을 지원하는 통신 리스너 EDU입니다. db2ipccm과 동일한 방법으로 작동합니다.
db2snacm	db2start 실행시	DB2 SNA/APPC Communication Manager 입니다. SNA/APPC 프로토콜을 이용하여 접속을 요청하는 원격 응용프로그램을 지원하는 통신 리스너 EDU입니다. db2ipccm과 동일한 방법으로 작동합니다.
db2tcpdm	db2start 실행시	DB2 TCP Discovery Manager 입니다. 클라이언트 머신에서 TCP/IP 프로토콜을 이용하여 원격 데이터베이스 서버의 인스턴스를 자동적으로 발견해내는 기능인 DISCOVERY 요청을 처리하는 통신 리스너입니다.

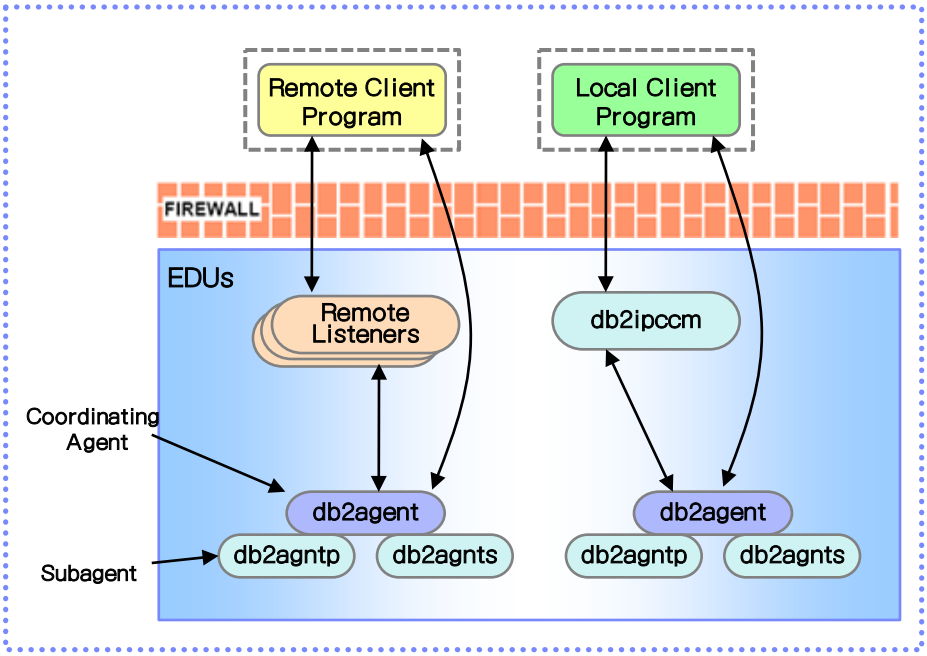


Figure 1607B ••• 통신 리스너와 db2agent EDU

TOPIC 1607 인스턴스 수준의 프로세스

Point

인스턴스 수준의 프로세스는 db2start 명령으로 인스턴스를 기동할 때 생성됩니다. db2fmcd, db2wdog, db2sysc, db2ckpwd, db2acd, db2fmd 등의 프로세스가 있습니다.

Tip

UserExit 프로그램으로 로그를 아카이브하려면, LOGARCHMETH1 데이터베이스 구성 변수에 아카이브용 디렉토리명을 지정해도 됩니다.

Tip

UserExit 프로그램을 사용하지 않고, 아카이브 로깅을 설정하는 경우에는 LOGARCHMETH1 데이터베이스 구성 변수에 LOGRETAIN 이라고 지정해도 됩니다.

Tip

UserExit 프로그램을 사용하지 않고, 아카이브 로깅을 설정하는 경우에는 LOGRETAIN 데이터베이스 구성 변수에 ON 이라고 지정해도 됩니다.

4 아카이브 로깅만 사용되거나, UserExit 프로그램을 이용하여 아카이브 로깅을 하는 경우에는 다음과 같은 프로세스가 관련됩니다.

프로세스명	생성시점	설명
db2cart	db2start 실행시	아카이브 로깅에서 UserExit이 설정되어 있는 경우에 사용됩니다. 로그 파일을 아카이브해야 하는 시점을 결정하고, Disk 또는 Tape 등의 저장 매체로 로그 파일의 아카이브를 실제로 수행하는 user exit 프로그램을 호출합니다. UserExit 프로그램을 이용한 아카이브 로깅을 사용하려면 데이터베이스별로 설정합니다. 데이터베이스 구성 변수인 LOGRETAIN을 ON으로 설정하고 USEREXIT도 ON으로 설정합니다.
db2fmtlg	db2start 실행시	DB2 Format Log 입니다. UserExit 프로그램을 사용하지 않지만, 로그 파일을 현재의 로그 디렉토리에 아카이브하는 경우에 로그 파일을 미리 포맷하여 할당합니다. DB2 엔진 프로세스는 특정한 로그 파일에 대한 기록을 완료하고, 다른 로그 파일로 연속적으로 기록하는 경우에 새로운 로그 파일이 생성될 때까지 기다리지 않고 처리를 계속할 수 있도록 합니다. UserExit 프로그램을 이용하지 않고 현재의 로그 디렉토리에 아카이브만 하게 하려면, 데이터베이스 구성 변수인 LOGRETAIN을 ON으로 설정하고, USEREXIT을 OFF로 설정합니다.

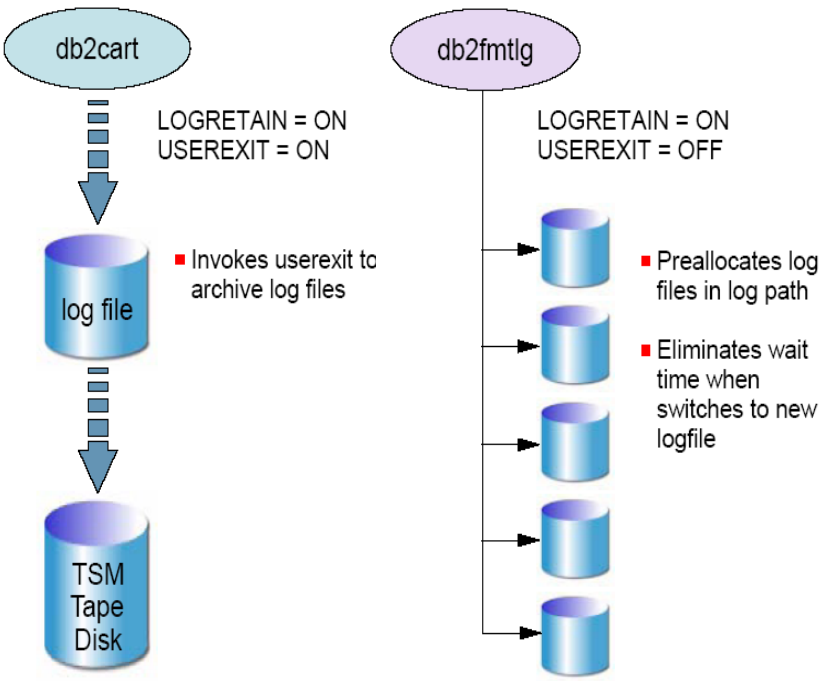


Figure 1607C ••• db2cart와 db2fmtlg 프로세스

TOPIC 1607 인스턴스 수준의 프로세스

Point

인스턴스 수준의 프로세스는 db2start 명령으로 인스턴스를 기동할 때 생성됩니다. db2fmcd, db2wdog, db2sysc, db2ckpwd, db2acd, db2fmd 등의 프로세스가 있습니다.

Tip

기존 컨테이너의 크기만 늘리는 경우에는 db2rebal 프로세스가 호출되지 않습니다.

Tip

새로운 컨테이너를 추가하는 경우에도 BEGIN NEW STRIPE SETS 옵션을 이용하면, db2rebal 프로세스가 호출되지 않지만, 데이터의 균등한 분배를 위하여 권장하지 않습니다.

5 특정한 DB2 명령어 또는 유틸리티를 실행할 때 생성되는 프로세스는 다음과 같습니다.

프로세스명	생성시점	설명
db2chkau	db2audit 시작시	DB2 Check Audit 입니다. db2audit 유틸리티가 DB2 audit 로그 파일의 각 항목을 audit 버퍼에서 $instance_home/sqllib/security/db2audit.log 파일로 로깅할 때 사용됩니다.
db2govd	db2gov 시작시	DB2 Governor Daemon 입니다. db2gov 유틸리티가 governor configuration으로 지정한 규칙에 따라 데이터베이스에서 수행되는 응용프로그램을 모니터링하고, 지정한 임계값을 초과한 응용프로그램에 대한 조치를 취하기 위해 사용됩니다. db2govd 유틸리티는 데이터베이스에 대한 응용프로그램이 아니므로, 데이터베이스에 접속하는 것이 아니라, 인스턴스에 접속합니다.
db2rebal	케이블스페이스에 컨테이너 추가시	DB2 Rebalancer 입니다. 기존의 테이블스페이스에 컨테이너를 추가할 때 호출되어, 기존 데이터를 추가된 컨테이너로 균등하게 분산시키는 작업을 수행합니다.

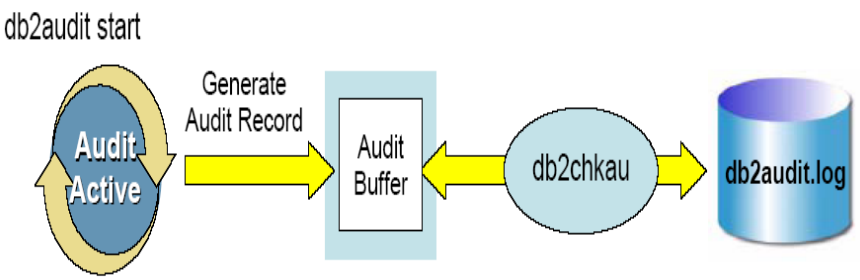


Figure 1607D ••• db2chkau 프로세스

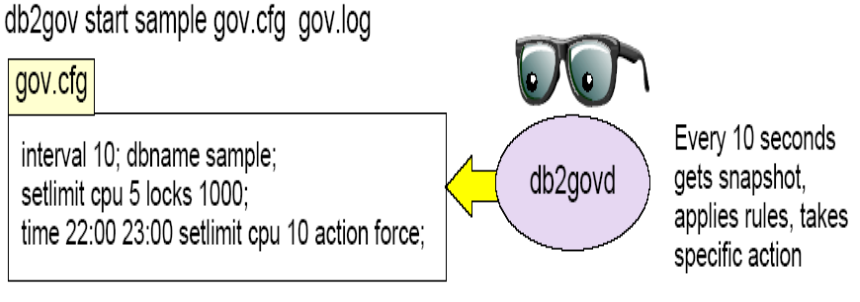


Figure 1607E ••• db2govd 프로세스

TOPIC 1607 인스턴스 수준의 프로세스

Point

인스턴스 수준의 프로세스는 db2start 명령으로 인스턴스를 기동할 때 생성됩니다. db2fmcd, db2wdog, db2sysc, db2ckpwd, db2acd, db2fmd 등의 프로세스가 있습니다.

6 DPF를 사용하여 다중 데이터베이스 파티션 환경을 구성한 경우에만 생성되는 프로세스들은 다음과 같습니다.

프로세스명	생성시점	설명
db2fcmd	db2start 실행시	DB2 Fast Communication Manager Daemon 입니다. 파티션간의 통신에 사용됩니다.
db2glock	db2start 실행시	DB2 Global Deadlock Detector 입니다. 개별 파티션에 있는 db2dlock 프로세스로부터 수집된 정보를 모아서 데이터베이스 파티션간의 교착 상태가 발생했는지를 점검합니다. 카탈로그 파티션에 존재합니다.
db2panic	db2start 실행시	DB2 Panic 입니다. 특정한 데이터베이스 파티션에서 agent의 한계에 도달하게 되면 긴급 요청을 처리합니다.
db2pdbc	db2start 실행시	DB2 Parallel Database Controller 입니다. 원격 데이터베이스 파티션간의 병렬 요청을 처리합니다. 개별 데이터베이스 파티션마다 한 개씩 존재합니다.

Tip

trans1.subsection1은 trans2.subsection2에 의해 잠금 대기 상태에 있고, trans1.subsection2에게 메시지를 반환할 수 없으므로, trans1은 종료되지 못하고 있습니다.

Tip

trans2.subsection1은 trans1.subsection2에 의해 잠금 대기 상태에 있고, trans2.subsection2에게 메시지를 반환할 수 없으므로, trans2도 종료되지 못하고 있습니다.

7 global deadlock 의 작동 원리를 이해하기 위하여 다음과 같은 상황을 가정하도록 합니다. 두 개의 트랜잭션 Trans1과 Trans2가 있습니다. 두 트랜잭션은 각각 한 개의 SQL문에 대한 subsection1과 subsection2 를 두 개의 파티션에서 병렬로 실행시키려고 합니다.
trans1은 파티션2에서 subsection2의 작업을 먼저 끝내고, 파티션1에서 subsection1 이 작업 결과 또는 메시지를 반환하기를 대기하고 있지만, subsection1은 파티션1에서 trans2의 subsection2가 lock을 걸고 있기 때문에 잠금 대기 상태에 있습니다.
trans2는 파티션1에서 subsection2의 작업을 먼저 끝내고, 파티션2에서 subsection1 이 작업 결과 또는 메시지를 반환하기를 대기하고 있지만, subsection1은 파티션2에서 trans1의 subsection2가 lock을 걸고 있기 때문에 잠금 대기 상태에 있습니다.
다중 파티션 환경에서 서로 잠금을 대기하는 교착 상황이 발생하게 되면, db2glock 프로세스가 점검하여 한 쪽의 응용프로그램을 강제로 종료시킵니다.

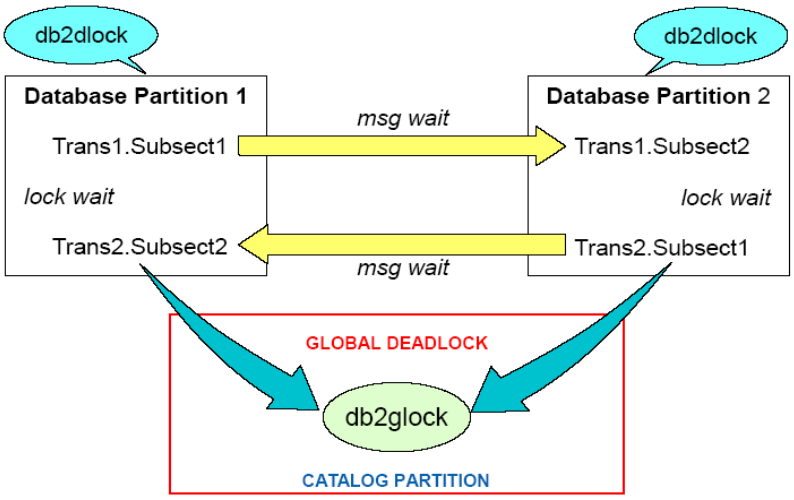


Figure 1607F ••• db2dlock과 db2glock 프로세스

TOPIC 1607 인스턴스 수준의 프로세스

Point

인스턴스 수준의 프로세스는 db2start 명령으로 인스턴스를 기동할 때 생성됩니다. db2fmcd, db2wdog, db2sysc, db2ckpwd, db2acd, db2fmd 등의 프로세스가 있습니다.

Tip

db2fmcd 와 db2fmd 프로세스는 UNIX 시스템 에서만 존재합니다.

8 Fault Monitor 와 연관된 프로세스입니다.

프로세스명	생성시점	설명
db2fmcd	시스템 boot 시	DB2 Fault Monitor Coordinator Daemon 입니다. 각 데이터베이스 서버 머신에 한 개씩 존재하며, db2fmcd 프로세스에 이상이 없는지를 모니터링 합니다. 사용자가 kill 명령을 이용하여 강제로 db2fmd 프로세스를 제거하더라도, db2fmd 프로세스는 db2fmcd 프로세스에 의해 다시 생성됩니다.
db2fmd	db2start 실행시	DB2 Fault Monitor Daemon 입니다. DB2 인스턴스를 모니터링하는 데몬 프로세스로 DB2 인스턴스가 db2stop 명령어에 의하여 정상적으로 중지되는 것을 제외한 경우에 다시 엔진을 기동시키는 역할을 합니다. HA 환경을 구성하는 경우에는 db2fmcd 와 db2fmd 프로세스가 자동적으로 시작되지 않도록 해야 합니다.

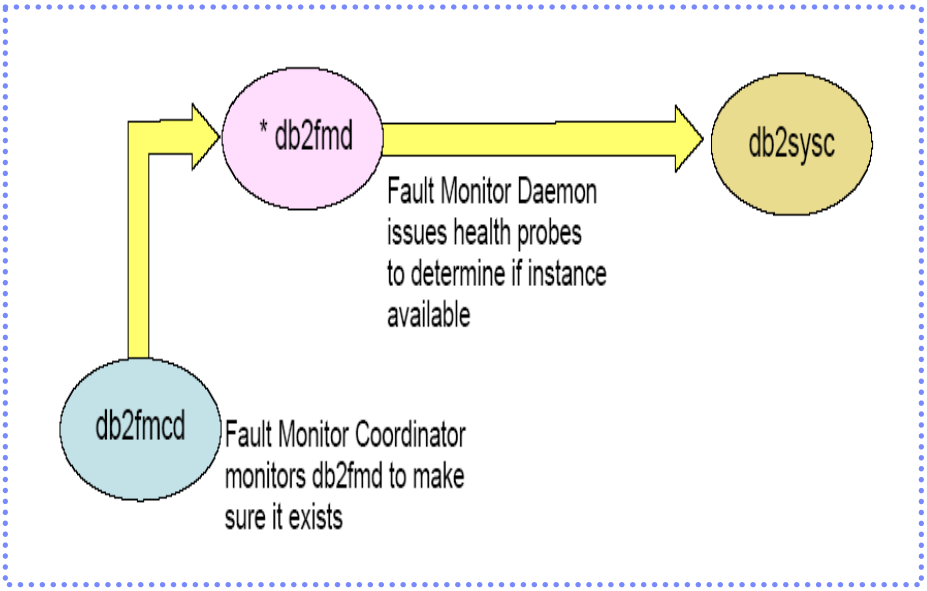


Figure 1607G db2fmcd과 db2fmd 프로세스

9 db2fmcd 프로세스는 기본적으로 시스템 부팅시에 자동적으로 시작되도록 /etc/inittab 파일에 등록되어 있습니다. 자동적으로 시작되는 것을 원하지 않는다면, 해당 명령행을 주석으로 처리하도록 합니다.

```
$ login root
$ cat /etc/inittab
...
#respawn:/opt/ibm/db2/V9.7/bin/db2fmcd
...
```

TOPIC 1607 인스턴스 수준의 프로세스

Point

인스턴스 수준의 프로세스는 db2start 명령으로 인스턴스를 기동할 때 생성됩니다. db2fmcd, db2wdog, db2sysc, db2ckpwd, db2acd, db2fmd 등의 프로세스가 있습니다.

Tip

원격 클라이언트의 응용프로그램에서 DB2 서버의 데이터베이스로 접속하려면, connect 문을 사용할 때 반드시 user와 using 옵션으로 사용자명과 암호를 제공해야 합니다.

Tip

Parallel Sysplex는 워크로드를 처리하기 위해 특정한 멀티시스템 하드웨어 구성 요소 및 소프트웨어 서비스로 서로 통신하고 협력하는 z/OS 또는 OS/390 시스템 세트입니다.

Tip

connection concentration 기능을 사용하려면, 인스턴스 구성 변수인 MAX_CONNECTIONS 의 값을 MAX_COORDAGENTS 보다 크게 설정합니다. 9.7에서는 기본값이 AUTOMATIC입니다.

10 OS와 관련된 작업을 수행하는 프로세스는 다음과 같습니다.

프로세스명	생성시점	설명
db2ckpw	db2start 실행시	DB2 Check Password 입니다. DB2 UDB는 OS의 인증 시스템을 이용하여 사용자에 대한 인증을 실행합니다. 원격 클라이언트 응용프로그램에서 제공된 사용자명과 암호가 OS에 등록된 사용자 정보와 일치하는지를 점검하여 접속의 허용 여부를 결정합니다. AIX에서 사용자 인증을 위해 /etc/security/passwd 파일을 사용합니다.
db2resyn	db2start 실행시	DB2 Resynchronization 입니다. 2단계 확약(2-phase commit) 방식을 사용하는 응용프로그램에 대한 동기화를 담당합니다. 단일 또는 다중 파티션 환경에서 모두 사용됩니다.
db2srvlst	db2start 실행시	DB2 Server List 입니다. OS/390과 같은 시스템에 대한 address list를 관리하는데 사용됩니다. Parallel Sysplex와 함께 사용됩니다.
db2syslog	db2start 실행시	DB2 System Logger 입니다. OS의 시스템 오류 로그 파일에 오류를 기록합니다.
db2disp	db2start 실행시	DB2 Dispatcher 입니다. connection concentration 기능을 사용하는 경우에 생성됩니다. 응용프로그램에 할당되어 있는 logical coordinating agent와 사용 가능한 coordinate agent를 연결시키는 역할을 합니다.

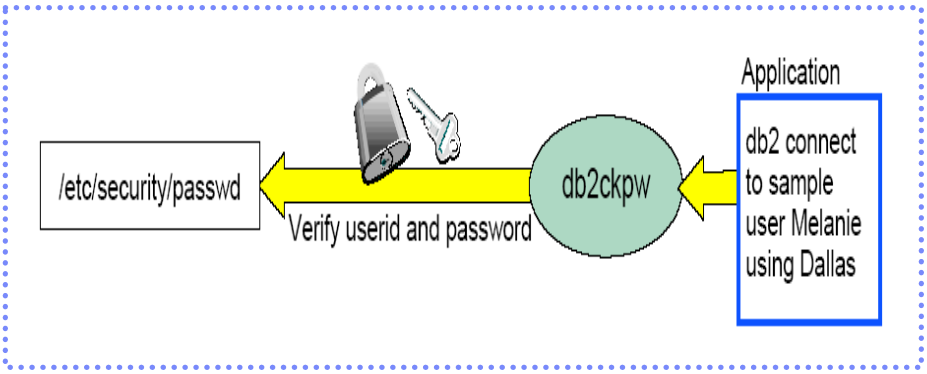


Figure 1607H ··· db2ckpw 프로세스

TOPIC 1608 데이터베이스 수준의 프로세스

Point

데이터베이스 수준의 프로세스는 ACTIVATE DB 명령어에 의해 데이터베이스가 활성화될 때 생성됩니다. db2pfchr를 비롯하여 db2pclnr, db2loggr, db2loggw, db2logts, db2dlock 등의 프로세스가 있습니다.

Tip

NUM_IOSERVER 데이터베이스 구성 변수의 기본값은 AUTOMATIC 입니다.

Tip

NUM_IOCLEANERS 데이터베이스 구성 변수의 기본값은 AUTOMATIC 입니다.

Tip

db2agent 프로세스가 버퍼풀의 페이지를 요구하는 시점 직전에 디스크로 변경 사항이 기록된 페이지를 good victim page 라고 하며, db2agent 프로세스는 버퍼풀 전체를 검색할 필요가 없게 됩니다. good victim page의 수가 기준값 이하로 떨어지면 페이지 클리너 프로세스를 호출합니다.

Tip

버퍼풀에서 그 값이 변경되었으나, 테이블스페이스의 컨테이너에 아직 반영되지 않은 값이 포함된 페이지를 Dirty Page 라고 합니다.

Tip

CHNGPGS_THRESH 데이터베이스 구성 변수의 기본값은 60%입니다. 갱신 작업이 많다면, CHNGPGS_THRESH 변수를 기준값 이하로 설정하여 버퍼풀에 가용 페이지를 충분히 확보하는 것이 유리합니다.

Tip

LSN은 로그 시퀀스 번호로 엔진이 트랜잭션을 식별하고 추적하게 합니다. MINBUFFLSN은 이미 커미트되었지만, 테이블스페이스 컨테이너로 반영되지 않은 가장 오래된 LSN을 표시합니다. 즉, 버퍼풀에서 가장 오래된 Dirty Page의 LSN을 의미합니다.

Tip

SOFTMAX 데이터베이스 구성 변수의 기본값은 100% 로 로그 파일 한 개의 크기만큼입니다.

1 db2pfchr 와 db2pclnr 프로세스는 버퍼풀과 테이블스페이스 컨테이너 사이의 비동기적인 I/O를 담당합니다.

프로세스명	생성시점	설명
db2pfchr	데이터베이스 활성화시	DB2 Bufferpool Prefetcher 입니다. 프리페처는 테이블과 인덱스의 데이터를 테이블스페이스 컨테이너인 디스크로부터 데이터베이스 버퍼풀로 비동기적인 방법으로 미리 읽어들이는 역할을 합니다. 각 응용프로그램을 대신하는 db2agent 프로세스가 db2pfchr 프로세스에게 프리페치 요청을 보내면, db2pfchr는 Big-block I/O 방식으로 데이터를 효율적으로 읽어들입니다. 데이터베이스 구성 변수인 NUM_IOSERVER 를 이용하여 데이터베이스별로 활동할 프리페처의 개수를 지정합니다.
db2pclnr	데이터베이스 활성화시	DB2 Bufferpool Page Cleaner 입니다. 데이터베이스 버퍼풀에 가용 공간을 확보하기 위해 버퍼풀로부터 비동기적인 방식으로 테이블스페이스 컨테이너에 변경된 데이터를 기록합니다. 특정한 페이지 클리너가 작업을 시작하면, 다른 페이지 클리너들도 동시에 작업을 수행합니다. 작업이 완료되면, 페이지 클리너 프로세스들은 sleep 상태로 대기하게 됩니다. 데이터베이스 구성 변수인 NUM_IOCLEANERS 를 이용하여 데이터베이스별로 활동할 페이지 클리너의 개수를 지정합니다.

2 db2pclnr 프로세스는 다음과 같은 경우에 활동하게 됩니다.

활동시점	설명
Dirty Page Threshold	버퍼풀의 Dirty Page 의 비율이 CHNGPGS_THRESH 데이터베이스 구성 변수의 값을 초과할 때 db2agent 프로세스가 호출됩니다.
LSN Gap Threshold	MINBUFFLSN과 현재LSN의 차이가 SOFTMAX 데이터베이스 구성 변수의 값을 초과할 때 log writer 프로세스가 호출합니다.

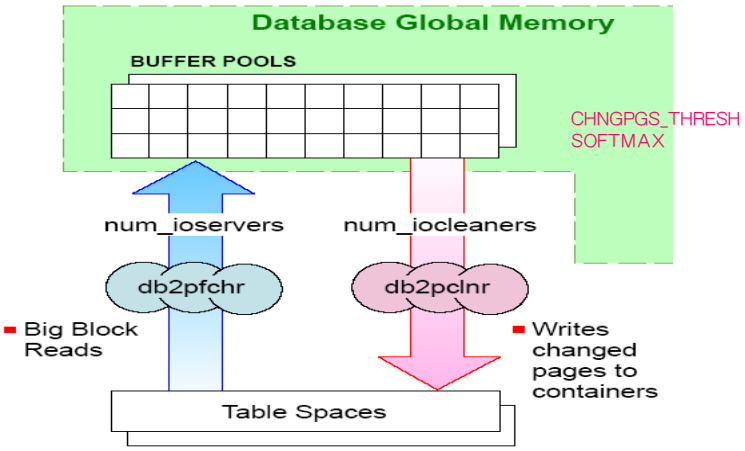


Figure 1608A ··· db2pfchr와 db2pclnr 프로세스

TOPIC 16 08 데이터베이스 수준의 프로세스

Point

데이터베이스 수준의 프로세스는 ACTIVATE DB 명령어에 의해 데이터베이스가 활성화될 때 생성됩니다. db2pfchr를 비롯하여 db2pclnr, db2loggr, db2loggw, db2logts, db2dlock 등의 프로세스가 있습니다.

Tip

버퍼풀 또는 로그 버퍼에 있는 데이터가 해당되는 테이블스페이스의 컨테이너에 기록될 때에는 반드시 로그 파일에 먼저 기록이 되어야 합니다. 이것을 WAL (Write Ahead Logging) 프로토콜이라고 합니다.

3 데이터베이스 로그 파일의 I/O 와 연관된 프로세스입니다.

프로세스명	생성시점	설명
db2loggr	db2start 실행시	DB2 Database Log Reader 입니다. rollback 문의 요청에 의한 트랜잭션 롤백, restart db 명령어 요청에 의한 응급 복구, rollforward db 명령어에 의한 아카이브 로그 적용시에 로그 파일로부터 데이터베이스 로그 레코드를 읽어들이는 역할을 합니다.
db2loggw	db2start 실행시	DB2 Database Log Write 입니다. 로그 버퍼에 공간이 없는 경우에 로그 버퍼에 저장된 로그 레코드를 로그 파일로 기록하는 역할을 합니다. 버퍼풀의 Dirty Page 가 디스크로 반영되거나 COMMIT 문이 요청되는 경우에도 작동됩니다.

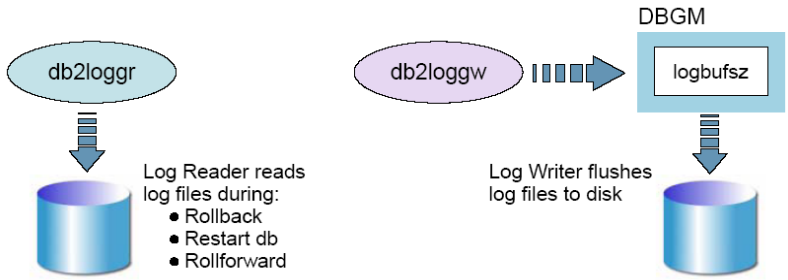


Figure 1608B ··· db2loggr 와 db2loggw 프로세스

Tip

db2logts는 레지스터리 변수인 DB2_COLLECT_TS_REC_INFO 를 설정할 때 사용됩니다.

4 테이블스페이스의 복구와 연관된 프로세스입니다.

프로세스명	생성시점	설명
db2logts	db2start 실행시	DB2 Log Tablespace입니다. 테이블스페이스가 변경될 당시에 어떤 로그가 활성화 상태였는지에 대한 history 정보를 수집하는데 사용됩니다. 수집된 정보는 DB2TSCHG.HIS 파일에 저장되며, 테이블스페이스에 대한 롤포워드 복구시에 해당 테이블스페이스의 복구 작업에 불필요한 로그 파일은 적용하지 않으므로 복구의 성능을 향상시키는데 사용됩니다.

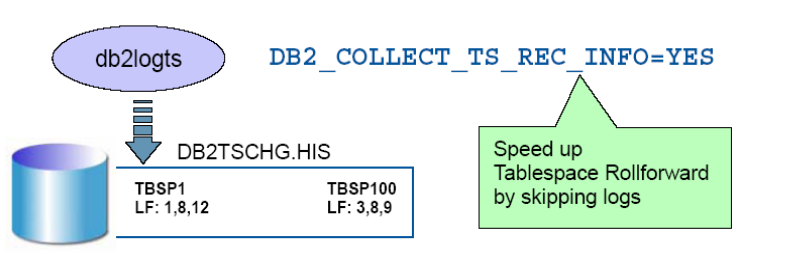


Figure 1608C ··· db2logts 프로세스

TOPIC 1608 데이터베이스 수준의 프로세스

Point

데이터베이스 수준의 프로세스는 ACTIVATE DB 명령어에 의해 데이터베이스가 활성화될 때 생성됩니다. db2pfchr를 비롯하여 db2pclnr, db2loggr, db2loggw, db2logts, db2dlock 등의 프로세스가 있습니다.

Tip

LOGARCHMETH1 데이터베이스 구성 변수는 V8.2 부터 지원됩니다.

Tip

USEREXIT 데이터베이스 구성 변수를 이용하여 사용자가 작성한 user exit program 을 이용하는 방법도 여전히 유효합니다.

5 아카이브 로그 파일의 USEREXIT 과 연관된 프로세스입니다.

프로세스명	생성시점	설명
db2logmgr	db2start 실행시	LOGARCHMETH1 데이터베이스 구성 변수에 아카이브 로그 디렉토리명이 지정된 경우에 사용됩니다. 기존에 사용자가 정의한 user exit program을 사용하지 않고, 엔진이 직접 로그 파일을 아카이브합니다.

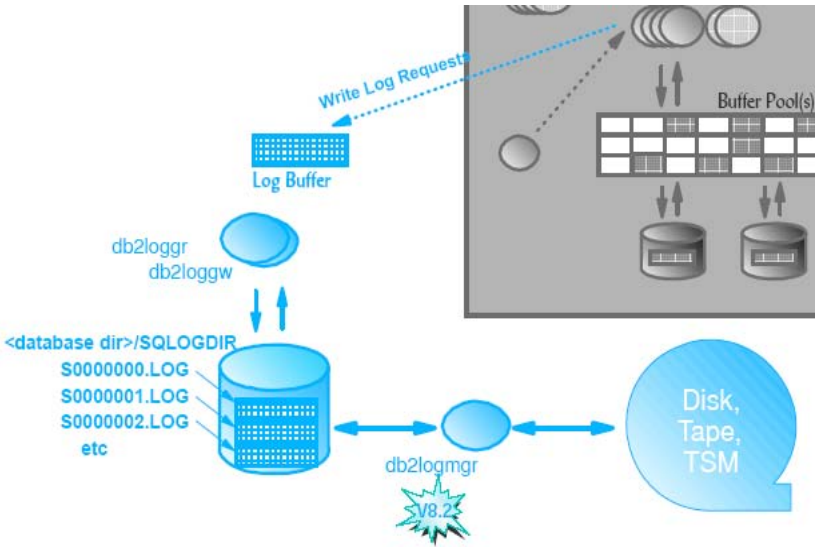


Figure 1608D db2logmgr 프로세스

Tip

확장 스토리지 영역은 데이터베이스 구성 변수인 ESTORE_SEG_SZ 와 NUM_ESTORE_SEGS 가 설정된 경우에만 사용할 수 있습니다. 64비트 머신에서는 필요하지 않습니다.

6 데이터베이스 수준의 프로세스에는 다음과 같은 프로세스들도 있습니다.

프로세스명	생성시점	설명
db2dlock	db2start 실행시	DB2 Local Deadlock Detector 입니다. 개별 파티션마다 존재하며, LOCKLIST를 스캔하여 교착 상태를 검출하는 역할을 합니다. 교착 상태에 있는 응용프로그램 중에서 희생자(victim)로 선택된 응용프로그램은 롤백됩니다. 희생자를 결정하는 우선 순위는 내부적인 기준에 의해 결정됩니다. db2dlock은 Z잠금을 보유한 응용프로그램 또는 load, redistribute, online reorg 등을 실행하는 세션은 희생자 프로세스가 되지 않도록 고려합니다.
db2estor	db2start 실행시	DB2 Extended Storage Manager 입니다. 데이터베이스 버퍼풀과 확장 스토리지 사이에서 데이터 페이지를 복사할 때 사용됩니다. NUM_ESTORE 데이터베이스 구성 변수가 설정된 경우에만 사용됩니다.
db2event	이벤트 모니터 시작시	DB2 Event Monitor 입니다. 활성화된 이벤트 모니터에 대해 각각 한 개씩 할당되어 이벤트 모니터 정보를 수집합니다.

TOPIC 16 09 응용프로그램 수준의 프로세스

Point

응용프로그램 수준의 프로세스는 지역 또는 원격 응용프로그램이 데이터베이스에 접속을 요청할 때 생성됩니다. db2agent를 비롯하여 db2agntp, db2agnta, db2agentg 등의 프로세스가 있습니다.

Tip

특정한 데이터베이스에 접속할 수 있는 응용프로그램의 최대 개수는 데이터베이스 구성 변수인 MAXAPPLS로 조절합니다.

Tip

특정한 인스턴스에서 생성될 수 있는 Coordinator Agent 프로세스의 최대 개수는 인스턴스 구성 변수인 MAX_COORDAGENTS 로 조절합니다.

Tip

에이전트 프로세스의 총 개수는 인스턴스 구성 변수인 MAXAGENTS를 이용하여 조절합니다.

1 통신 리스너 프로세스는 지역 또는 원격 응용프로그램이 데이터베이스에 대한 접속을 요청하면 다음과 같은 에이전트 프로세스를 생성합니다.

프로세스명	생성시점	설명
db2agent	데이터베이스 접속시	DB2 Coordinator Agent 입니다. 접속을 요청한 응용프로그램을 대신하여 엔진에게 응용프로그램의 요청을 전달하고, 엔진이 반환한 결과를 응용프로그램에게 전달합니다. 응용프로그램별로 한 개의 db2agent 프로세스로 관리합니다. Coordinator Agent 또는 간단하게 Agent 프로세스라고 합니다.
db2agntp	데이터베이스 접속시	DB2 Subagent 입니다. 병렬 처리가 가능한 환경인 경우에 Coordinator Agent는 응용프로그램으로부터 전달받은 데이터베이스 요청들을 Subagent 에게 분배하여, 응용프로그램을 대신하여 엔진에게 요청을 전달하도록 합니다. Subagent 들이 엔진으로부터 받은 결과를 다시 Coordinator Agent 에게 반환하면, Coordinator Agent 가 종합하여 응용프로그램에게 최종적인 결과를 반환합니다.
db2agentg	데이터베이스 접속시	DB2 Gateway Agent 입니다. DB2 Connect 제품을 이용하여 호스트 데이터베이스에 접속하거나, 다른 인스턴스를 경유하여 데이터베이스에 접속을 요청하는 경우에는 db2agent 대신에 db2agentg 프로세스가 생성됩니다.

Tip

병렬 처리의 유형은 파티션내 (Intra-Partition) 병렬 처리와 파티션간 (Inter-Partition) 병렬 처리로 구분됩니다.

Tip

파티션내 (Intra -Partition) 병렬 처리는 단일한 데이터베이스 파티션에서 항 개의 SQL문이 병렬로 처리될 수 있게 합니다. 인스턴스 구성 변수인 INTRA_PARALLEL 옵션을 적용하였을 경우에만 지원됩니다.

Tip

파티션간 (Inter -Partition) 병렬 처리는 다중 데이터베이스 파티션 환경에서 가능합니다. 파티션내 병렬 처리와 함께 지원되도록 할 수 있습니다.

2 단일 데이터베이스 파티션에서 INTRA_PARALLEL 인스턴스 구성 변수를 이용하여 파티션내 (Intra-Partition) 병렬 처리를 지정한 경우에 db2agent와 db2agntp 프로세스의 관계는 다음과 같습니다.

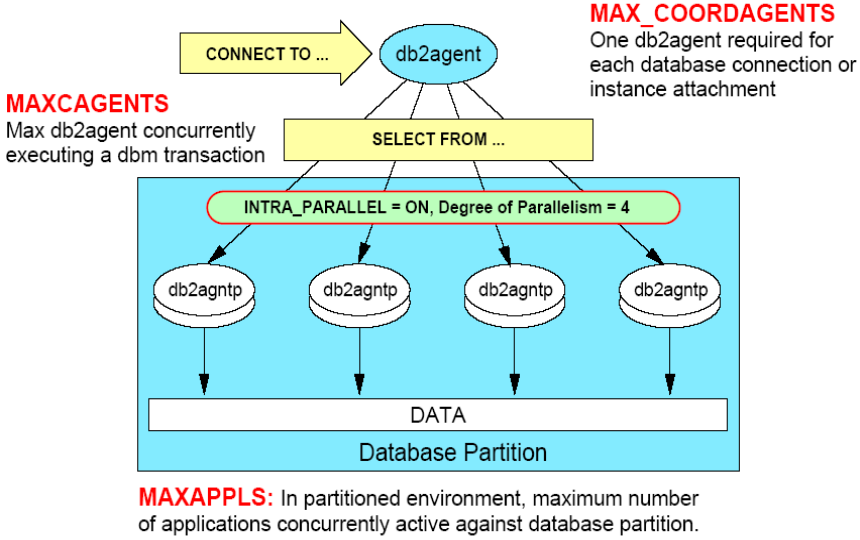


Figure 1609A ··· db2agent 와 db2agntp 프로세스

TOPIC 1609 응용프로그램 수준의 프로세스

Point

응용프로그램 수준의 프로세스는 지역 또는 원격 응용프로그램이 데이터베이스에 접속을 요청할 때 생성됩니다. db2agent를 비롯하여 db2agntp, db2agnta, db2agentg 등의 프로세스가 있습니다.

Tip

Agent Pooling을 사용하면, 응용프로그램이 접속을 요청할 때마다 에이전트 프로세스를 생성하는 오버헤드를 줄일 수 있습니다. 인스턴스 구성 변수인 NUM_POOLAGENTS와 NUM_INITAGENTS로 조절합니다.

3 Subagent로 생성되어 사용이 완료된 프로세스는 성능을 위해 즉시 제거하지 않고 Agent Pool 에 저장하였다가 재사용할 수 있습니다.

프로세스명	생성시점	설명
db2agnta	특정 응용프로그램에 대한 작업 종료시	Agent Pool 에서 응용프로그램에게 할당되기를 기다리는 Agent를 Idle Agent라고 하고, 다른 Coordinator Agent에 의해 요청되어 다시 Subagent로 사용됩니다.

4 응용프로그램이 접속을 요청했을 때, Agent 프로세스를 할당하는 우선 순위는 다음과 같은 방법으로 결정됩니다.

1. 요청한 응용프로그램과 연관이 있으면서 Idle 상태에 있는 Subagent를 할당합니다.
2. 다른 응용프로그램과 연관이 있으면서 Idle 상태에 있는 Subagent 를 할당합니다.
3. MAXAGENTS 구성 변수의 값에 도달하지 않았다면 새로운 Subagent를 생성합니다.
4. 다른 응용프로그램과 연관이 있으면서 Idle한 Subagent를 가로채기 합니다.

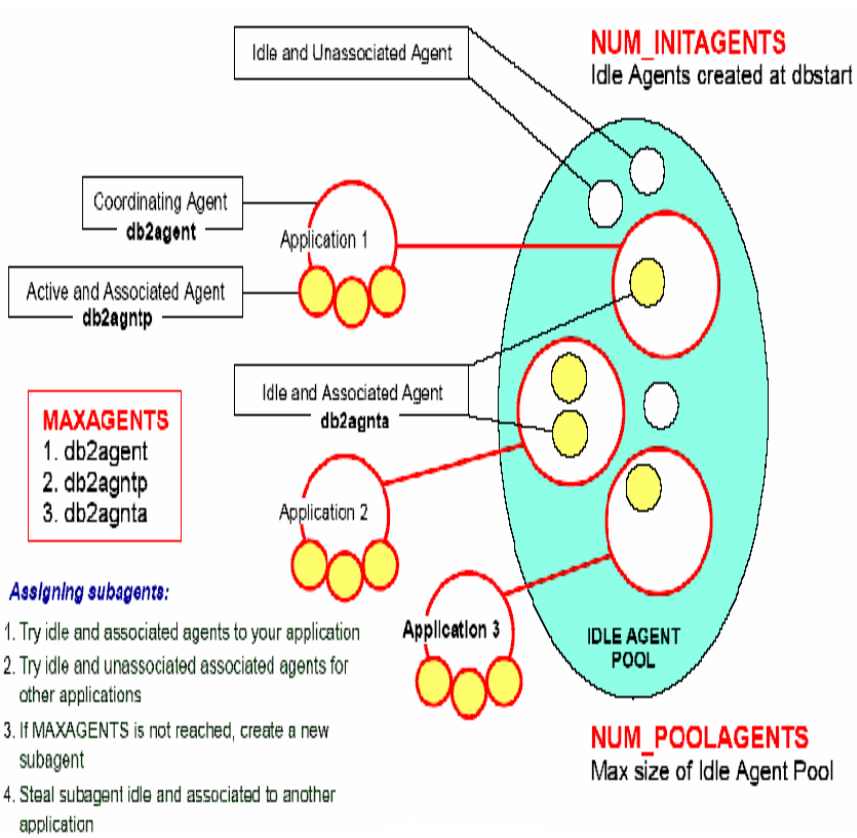


Figure 1609B ••• Agent Pool과 db2agnta 프로세스

TOPIC 1609 응용프로그램 수준의 프로세스

Point

응용프로그램 수준의 프로세스는 지역 또는 원격 응용프로그램이 데이터베이스에 접속을 요청할 때 생성됩니다. db2agent를 비롯하여 db2agntp, db2agnta, db2agentg 등의 프로세스가 있습니다.

5 병렬 복구 처리를 위한 프로세스입니다.

프로세스명	생성시점	설명
db2agnsc	데이터베이스 접속시	DB2 Parallel Recovery Agent 입니다. 로그 파일에서 읽어온 로그 레코드의 내용에 따라 rollforward restart db, rollforward 명령을 병렬로 처리하여 복구 유틸리티의 성능을 높이기 위해 사용됩니다. 병렬 처리에 사용되는 Agent 프로세스의 개수는 SMP 머신의 CPU 개수를 고려하여 엔진이 결정합니다.

6 아래 그림에서 Coordinator Agent는 읽어온 로그 레코드를 병렬 프리페처를 이용하여 버퍼풀로 저장합니다. Parallel Recovery Agent 들은 버퍼풀의 데이터를 이용하여 병렬로 복구 작업을 진행합니다.

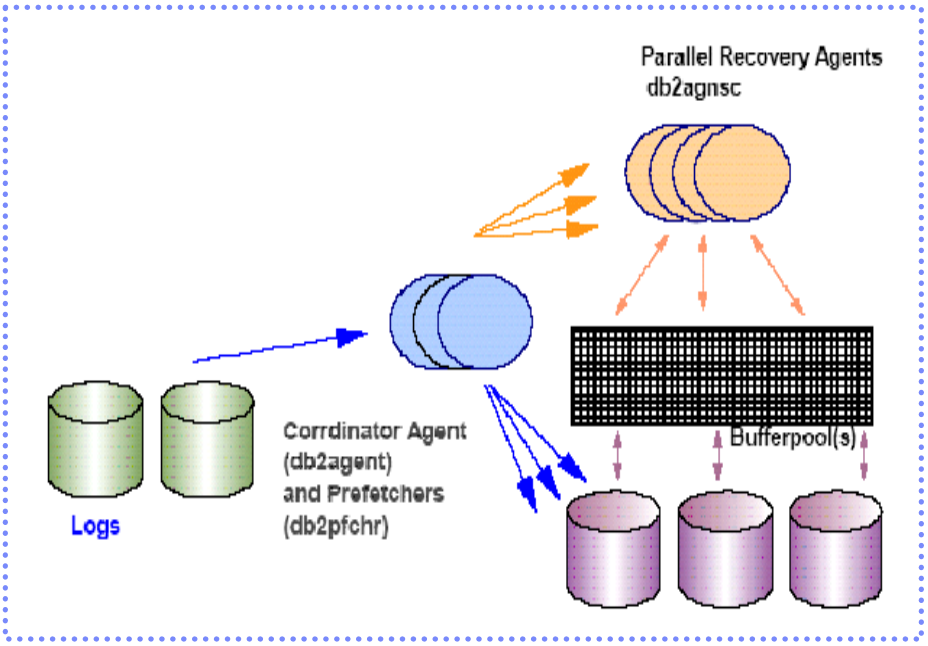


Figure 1609C ••• db2agnsc 프로세스

TOPIC 16-10 메모리 모델

Point

DB2 UDB의 메모리는 인스턴스 공유 메모리, 데이터베이스 공유 메모리, 응용프로그램 공유 메모리, 응용프로그램 개별 메모리로 구성됩니다. 각 수준의 메모리는 인스턴스 기동, 데이터베이스 활성화, 응용프로그램 접속시에 할당됩니다.

Tip

특정한 인스턴스에서 동시에 활성화 될 수 있는 데이터베이스의 최대 개수는 인스턴스 구성 변수인 NUMDB를 이용하여 조절합니다.

Tip

특정한 데이터베이스에 동시에 접속할 수 있는 응용프로그램의 최대 개수는 MAXAPPLS 데이터베이스 구성 변수를 이용하여 조절합니다.

Tip

특정한 인스턴스에서 존재할 수 있는 에이전트 프로세스의 최대 개수는 인스턴스 구성 변수인 MAXAGENTS를 이용하여 조절합니다.

1 DB2 메모리는 다음과 같이 4가지로 구성됩니다.

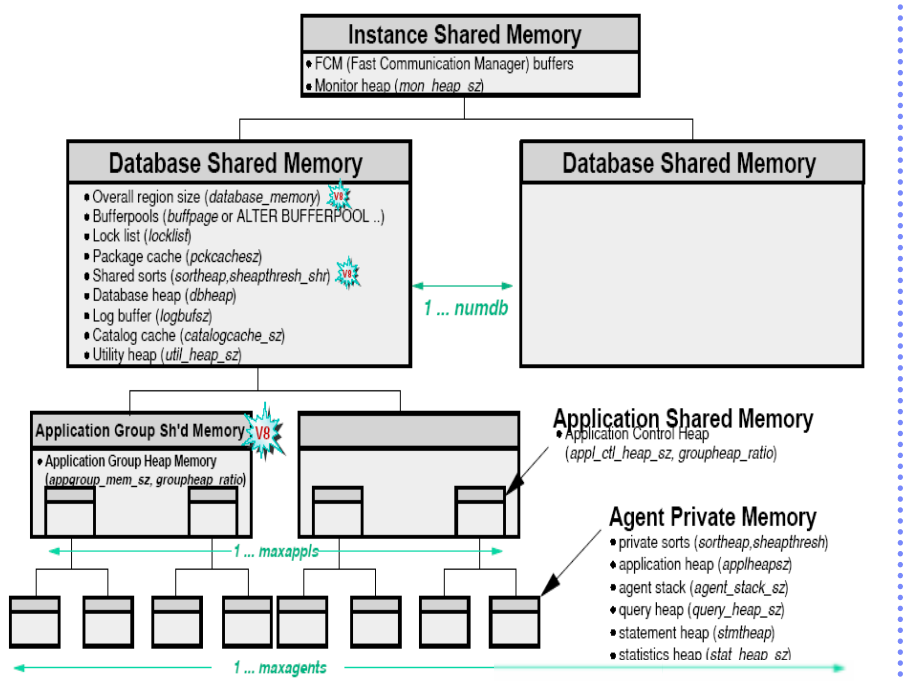


Figure 1610A ••• 메모리 모델

2 각 메모리별 특성은 다음과 같습니다.

메 모 리	설 명
인스턴스 공유 메모리	db2start 명령을 실행하여 인스턴스를 기동할 때 할당되고, db2stop 명령어로 인스턴스를 중지할 때 해제됩니다. Monitor heap, FCM, Audit buffer 등이 포함됩니다.
데이터베이스 공유 메모리	activate db 명령어에 의해 데이터베이스가 활성화될 때 할당되고, deactivate db 명령어에 의해 데이터베이스가 비활성화되면 해제됩니다. Buffer pool 을 비롯하여 Database Heap, Catalog Cache, Log Buffer, Lock List, Package Cache, Shared Sort heap, Utility Heap 등이 포함됩니다.
응용프로그램 공유 메모리	병렬 처리가 가능한 환경에서 응용프로그램의 첫 번째 에이전트 프로세스가 데이터베이스에 연결을 요청하는 경우에 할당되며, 해당 응용프로그램과 연관된 모든 에이전트 프로세스 사이의 정보를 교환을 위해 사용되는 영역입니다. Application control heap 이 포함됩니다.
응용프로그램 개별 메모리	지역 또는 원격 응용프로그램이 데이터베이스에 접속을 요청하는 경우에 할당되고, 접속을 종료하면 해제됩니다. Sort Heap, Statement Heap, Application Heap, Statistics Heap, Query Heap 등이 포함됩니다.

TOPIC 16.11 인스턴스 공유 메모리

Point

인스턴스 공유 메모리는 db2start 명령을 실행하여 인스턴스를 기동할 때 할당됩니다. Monitor heap, FCM, Audit buffer 등이 포함됩니다.

1 인스턴스 공유 메모리는 db2start 명령어로 엔진 기동시에 할당되며, 인스턴스에 존재하는 모든 EDU들은 이 영역을 액세스할 수 있습니다.

```
$ login <인스턴스 사용자>
$ db2start
```

2 할당된 인스턴스 공유 메모리의 양은 db2mtrk 명령어로 확인할 수 있습니다.

```
$ db2mtrk -i -v
메모리 추적 설정: 16:30:07에서 2009/08/17
인스턴스용 메모리
  Other Memory의 크기는 13697024바이트입니다.
  FCMBP Heap의 크기는 786432바이트입니다.
  Database Monitor Heap의 크기는 65536바이트입니다.
  총계: 14548992바이트
```

Tip

-i 옵션은 인스턴스 수준의 메모리를 대상으로 지정하고, -v 옵션은 상세한 메모리 사용 정보를 표시합니다.

3 인스턴스 공유 메모리의 양은 get snapshot for dbm 명령어로 확인할 수 있습니다.

```
$ db2 get snapshot for dbm
데이터베이스 관리 프로그램의 메모리 사용:
 노드 번호                    = 0
  메모리 풀 유형                = 기타 메모리
    현재 크기(바이트)           = 13828096
    상위 워터 마크(바이트)      = 13893632
    구성된 크기(바이트)         = 32768000
 노드 번호                    = 0
  메모리 풀 유형                = FCMBP 힙
    현재 크기(바이트)           = 786432
    상위 워터 마크(바이트)      = 786432
    구성된 크기(바이트)         = 917504
 노드 번호                    = 0
  메모리 풀 유형                = 데이터베이스 모니터 힙
    현재 크기(바이트)           = 327680
    상위 워터 마크(바이트)      = 327680
    구성된 크기(바이트)         = 327680
```

Tip

한 개의 물리적인 서버 머신에서 정의된 논리적인 데이터베이스 파티션들은 FCM을 공유할 수 있습니다. AIX에서는 DB2_FCM_FORCE_BP를 설정합니다.

4 인스턴스 공유 메모리는 다음과 같은 메모리 영역으로 구성됩니다.

메모리명	할당시점	설명
Monitor heap	db2start 실행시	스냅샷 모니터, 이벤트 모니터 등을 이용하여 모니터링 정보를 수집할 때 사용되는 메모리 영역으로 MON_HEAP_SZ 인스턴스 구성 변수를 이용하여 조절합니다.
FCM	db2start 실행시	다중 데이터베이스 파티션 환경에서 파티션간의 통신에 사용되는 메모리 영역으로 FCM_NUM_BUFFERS 인스턴스 구성 변수를 이용하여 조절합니다.
Audit buffer	db2audit 실행시	db2audit 유틸리티가 수집한 데이터를 저장하는 audit 용 버퍼 영역입니다. db2chkau 프로세스는 audit 용 버퍼에 저장된 데이터를 db2audit.log 파일에 비동적으로 기록합니다. AUDIT_BUF_SZ 인스턴스 구성 변수를 이용하여 조절합니다.

Tip

db2chkau 프로세스는 audit 용 버퍼가 없으면 db2audit.log 파일에 동기적으로 기록하게 되므로 성능이 저하됩니다.

TOPIC 16-12 데이터베이스 공유 메모리

Point

데이터베이스 공유 메모리는 데이터베이스가 활성화시에 할당됩니다. Buffer pool, Database Heap, Catalog Cache, Log Buffer, Lock List, Package Cache, Shared Sort heap, Utility Heap 등이 포함됩니다.

Tip

-d 옵션은 데이터베이스 수준의 메모리를 대상으로 지정하고, -v옵션은 상세한 메모리 사용 정보를 표시합니다.

1 데이터베이스 공유 메모리는 activate db 명령어로 데이터베이스 활성화시에 할당되며, 해당 데이터베이스에 연관된 모든 EDU들은 이 영역을 액세스할 수 있습니다.

```
$ login <인스턴스 사용자>
$ db2 activate db <데이터베이스명>
```

2 할당된 데이터베이스 공유 메모리의 양은 db2mtrk 명령어로 확인할 수 있습니다.

```
$ db2mtrk -d -v
메모리 트랙 위치: 19:48:41의 2005/07/18
데이터베이스용 메모리: SAMPLE
  Backup/Restore/Util Heap의 크기는 16384바이트임
  Package Cache의 크기는 131072바이트임
  Catalog Cache Heap의 크기는 65536바이트임
  Buffer Pool Heap의 크기는 4341760바이트임
  Buffer Pool Heap의 크기는 655360바이트임
  Buffer Pool Heap의 크기는 393216바이트임
  Buffer Pool Heap의 크기는 262144바이트임
  Buffer Pool Heap의 크기는 196608바이트임
  Lock Manager Heap의 크기는 491520바이트임
  Database Heap의 크기는 3555328바이트임
  Other Memory의 크기는 0바이트임
  총계: 10108928바이트
```

3 데이터베이스 공유 메모리의 양은 get snapshot for db 명령어로 확인할 수 있습니다.

```
$ db2 get snapshot for db on <데이터베이스명>
데이터베이스의 메모리 사용:
  메모리 풀 유형              = 백업/리스토어/유틸리티 힙
    현재 크기(바이트)          = 16384
    상위 워터 마크(바이트)      = 16384
    구성된 크기(바이트)         = 20496384
  메모리 풀 유형              = 패키지 캐시 힙
    현재 크기(바이트)          = 131072
    상위 워터 마크(바이트)      = 131072
    구성된 크기(바이트)         = 2147483648
  메모리 풀 유형              = 카탈로그 캐시 힙
    현재 크기(바이트)          = 65536
    상위 워터 마크(바이트)      = 65536
    구성된 크기(바이트)         = 2147483648
  ...
```

TOPIC 16-12 데이터베이스 공유 메모리

Point

데이터베이스 공유 메모리는 데이터베이스가 활성화시에 할당됩니다. Buffer pool, Database Heap, Catalog Cache, Log Buffer, Lock List, Package Cache, Shared Sort heap, Utility Heap 등이 포함됩니다.

4 데이터베이스 공유 메모리는 다음과 같은 메모리 영역으로 구성됩니다.

메모리명	할당시점	설명
Buffer pool	데이터베이스 활성화시	데이터베이스 공유 메모리에서 가장 큰 메모리 영역으로 다중 버퍼풀을 생성하여 테이블스페이스별로 할당할 수 있습니다. 데이터베이스가 활성화될 때 할당되며, CREATE / ALTER / DROP BUFFERPOOL문에 의해 동적으로 버퍼풀을 추가하거나, 크기를 조절할 수 있습니다.
Database Heap	데이터베이스 활성화시	테이블, 인덱스, 테이블스페이스, 버퍼풀 등의 데이터베이스 오브젝트에 대한 Control Block 정보를 저장하고, Log Buffer와 Catalog Cache를 포함하는 영역으로 데이터베이스 구성 변수인 DBHEAP 을 이용하여 동적으로 조절할 수 있습니다.
Catalog Cache	데이터베이스 활성화시	SQL문을 컴파일하는 동안 SYSIBM.SYSTABLES, SYSIBM.SYSDBAUTH, SYSIBM.SYSROUTINES 등의 시스템 카탈로그의 정보를 저장하는 메모리 영역으로 Database Heap 에서 할당되며 데이터베이스 구성 변수인 CATALOGCACHE_SZ 를 이용하여 동적으로 조절할 수 있습니다.
Log Buffer	데이터베이스 활성화시	로그 레코드를 디스크의 로그 파일에 기록하기 전에 버퍼링하는 메모리 영역으로 Database Heap 에서 할당되며 데이터베이스 구성 변수인 LOGBUFSZ 를 이용하여 동적으로 조절할 수 있습니다.
Lock List	데이터베이스 활성화시	데이터베이스에 접속된 모든 응용프로그램이 보유하고 있는 테이블과 행 잠금 정보를 저장하는 메모리 영역입니다. 데이터베이스 구성 변수인 LOCKLIST 를 이용하여 동적으로 조절할 수 있습니다.
Package Cache	데이터베이스 활성화시	static 및 dynamic SQL문을 저장하는 캐시 영역으로 Static SQL문이 포함된 팩키지를 reload 하거나 dynamic SQL문을 재컴파일하는 내부적인 오버헤드를 줄여줍니다. PCKCACHESZ 데이터베이스 구성 변수를 이용하여 동적으로 조절할 수 있습니다.
Shared sort heap	데이터베이스 활성화시	INTRA_PARALLEL 인스턴스 구성 변수를 설정하거나 Connection Concentrator 기능을 사용한다면, SQL문에 대한 병렬 처리가 가능합니다. Shared Sort Memory는 Subagent들이 정렬 작업을 병렬로 처리하고, 그 결과를 공유하기 위해 사용하는 메모리 영역입니다. SHEAPTHRES_SHR 데이터베이스 구성 변수로 조절합니다.
Utility Heap	Utility 실행시	Load, Backup, Restore, Runstats 등의 유틸리티가 실행될 때 사용하는 메모리 영역으로 동적으로 조절이 가능합니다. 유틸리티가 수행될 때 할당되고, 유틸리티가 종료되면 반환됩니다. 데이터베이스 구성 변수인 UTIL_HEAP_SZ 를 이용하여 조절합니다.

Tip

32비트 데이터베이스에서는 한 개의 잠금이 40 또는 80 바이트를 차지하고, 64비트 데이터베이스에서는 64 또는 128 바이트를 차지합니다.

Tip

SHEAPTHRES_SHR 데이터베이스 구성 변수를 0으로 설정하면, 데이터베이스 구성 변수인 SHEAPTHRES의 값을 사용합니다.

TOPIC 16-13 응용프로그램 공유 메모리

Point

병렬 처리가 가능한 환경에서 응용프로그램의 데이터베이스에 연결을 요청하는 경우에 할당되며, 해당 응용프로그램과 연관된 모든 에이전트 프로세스 사이의 정보를 교환을 위해 사용되는 영역입니다. Application control heap 이 포함됩니다.

Tip

DFT_DEGREE 데이터베이스 구성 변수의 기본값은 1 이고, 인스턴스 구성 변수인 INTRA_PARALLEL 기본값은 NO 이므로 단일 데이터베이스 파티션 환경에서 기본적으로 SQL문은 병렬로 처리되지 않습니다.

Tip

DFT_DEGREE 데이터베이스 구성 변수의 값에서 지정하는 병렬 등급은 1 ~ 32767입니다. -1을 지정하면, 옵티마이저가 프로세서의 개수 및 쿼리 유형에 기초하여 파티션내 병렬 처리 등급을 결정하게 됩니다.

Tip

처음에는 기본값으로 시작하고, 복잡한 응용프로그램을 실행하거나 데이터베이스 파티션 개수가 많은 경우 또는 선언된 임시 테이블을 사용할 경우에는 값을 증가시킵니다.

1. 응용프로그램 공유 메모리는 응용프로그램의 첫 번째 에이전트 프로세스가 데이터베이스에 연결을 요청하는 경우에 할당되며, 응용프로그램이 완료되면, 반환됩니다. 해당 응용프로그램과 연관된 모든 EDU들이 액세스할 수 있습니다.

2. 이 영역은 다중 데이터베이스 파티션 환경이나 INTRA_PARALLEL 구성 변수를 이용한 파티션내 병렬 처리가 가능한 환경에서 할당됩니다. 이 메모리 영역의 용도는 특정한 응용프로그램의 요청을 처리하는 Subagent와 Coordinator Agent 간의 정보 교환입니다. 단일 데이터베이스 파티션 환경이거나 INTRA_PARALLEL 구성 변수를 이용하였지만 쿼리에 대한 병렬 처리 등급인 1인 경우에는 이 영역을 사용은 최소화됩니다.

```
$ login <인스턴스 사용자>
$ db2 update dbm cfg using INTRA_PARALLEL YES
$ db2 update db cfg for <데이터베이스명> using DFT_DEGREE <병렬 등급>
$ db2stop force
$ db2start
$ db2 get dbm cfg | grep INTRA_PARALLEL
 파티션내 병렬 처리 사용            (INTRA_PARALLEL) = YES
$ db2 get db cfg for <데이터베이스명> |  grep DFT_DEGREE
 병렬 처리 등급                     (DFT_DEGREE) = <병렬 등급>
```

3. SQL문의 파티션 내 병렬 처리 수준은 CURRENT DEGREE 특수 레지스터 또는 DEGREE 바인드 옵션을 사용하여 명령문 컴파일시 지정됩니다. 기본값은 DFT_DEGREE 데이터베이스 구성 변수에 지정된 값입니다. 실행 중인 응용프로그램의 파티션 내 병렬 처리의 최대 런타임 수준은 SET RUNTIME DEGREE 명령을 사용하여 지정됩니다.

4. 에이전트 공유 메모리는 다음과 같은 메모리 영역으로 구성됩니다.

메모리명	할당시점	설명
Application control heap	데이터베이스 연결시	다중 데이터베이스 파티션의 파티션간 병렬 처리 또는 INTRA_PARALLEL 인스턴스 구성 변수를 이용한 파티션내 병렬 처리가 가능한 환경에서 응용프로그램의 첫 번째 에이전트 프로세스가 데이터베이스에 연결을 요청하는 경우에 할당되며, 해당 응용프로그램과 연관된 모든 에이전트 프로세스 사이의 정보를 교환하기 위해 사용되는 영역입니다. 이 영역은 선언된 임시 테이블에 대한 디스크립터 정보를 저장하는 데도 사용됩니다. 명시적으로 삭제되지 않은 선언된 모든 임시 테이블에 대한 디스크립터 정보가 보존되므로 선언된 임시 테이블이 삭제될 때까지 메모리를 반환할 수 없습니다. 파티션마다 각 연결에 한 개씩 할당되며, APP_CTL_HEAP_SZ 데이터베이스 구성 변수로 조절할 수 있습니다.

TOPIC 16.14 응용프로그램 개별 메모리

Point

지역 또는 원격 응용프로그램이 데이터베이스에 접속을 요청하는 경우에 할당되고, 접속을 종료하면 해제됩니다. Sort Heap, Statement Heap, Application Heap, Statistics Heap, Query Heap 등이 포함됩니다.

Tip

-p 옵션은 에이전트 수준의 메모리를 대상으로 지정하고, -v옵션은 상세한 메모리 사용 정보를 표시합니다.

1 에이전트 개별 메모리는 connect 명령어로 데이터베이스에 접속할 때 생성되는 에이전트 프로세스에게 개별적으로 할당되는 메모리 영역입니다.

```
$ login <인스턴스 사용자>
$ db2 connect to <데이터베이스명>
```

2 할당된 데이터베이스 공유 메모리의 양은 db2mtrk 명령어로 확인할 수 있습니다.

```
$ db2mtrk -p -v
메모리 추적 설정: 16:43:40에서 2009/08/17
에이전트 3708용 메모리

   Other Memory의 크기는 720896바이트입니다.
   총계: 720896바이트
```

3 특정한 에이전트 개별 메모리의 양은 확인하려면, list applications 명령어로 에이전트에 대응되는 핸들 번호를 확인합니다.

```
$ db2 list applications
권한 ID  응용프로그램 이름  Appl. 핸들  응용프로그램 ID                          DB 이름  에이전트 수
-------- ---------------- ---------- ---------------------------------------- -------- ----------
ADF      db2bp.exe        146        *LOCAL.DB2.090817073144                  SAMPLE   1
```

4 get snapshot for application 명령어에서 agentid 옵션으로 특정한 에이전트 개별 메모리의 양을 확인합니다.

```
$ db2 get snapshot for application agentid <응용프로그램 핸들 번호>
응용프로그램의 메모리 사용:

  메모리 풀 유형                = 응용프로그램 힙
    현재 크기(바이트)           = 65536
    상위 워터 마크(바이트)      = 65536
    구성된 크기(바이트)         = 1048576

에이전트 프로세스/스레드 ID     = 3708
  에이전트 잠금 시간종료(초)    = -1
  에이전트의 메모리 사용:

    메모리 풀 유형              = 기타 메모리
      현재 크기(바이트)         = 720896
      상위 워터 마크(바이트)    = 917504
      구성된 크기(바이트)       = 3195011072
```

TOPIC 16-14 응용프로그램 개별 메모리

Point

지역 또는 원격 응용프로그램이 데이터베이스에 접속을 요청하는 경우에 할당되고, 접속을 종료하면 해제됩니다. Sort Heap, Statement Heap, Application Heap, Statistics Heap, Query Heap 등이 포함됩니다.

Tip

Sort heap 영역이 부족하게 되면, 임시 테이블스페이스에 임시 파일 또는 테이블을 생성하게 되므로 성능이 저하됩니다.

5 에이전트 개별 메모리는 다음과 같은 메모리 영역으로 구성됩니다.

메모리명	할당시점	설명
Sort Heap	정렬 작업,해쉬 조인	정렬 작업 또는 해쉬 조인(Hash Join)이 필요할 때 할당되고, 작업이 종료되면 메모리를 반환합니다. SORTHEAP 데이터베이스 구성 변수를 이용하여 동적으로 조절이 가능합니다.
Statement Heap	프리컴파일, 바인드	SQL 컴파일러의 작업 영역으로 사용되는 메모리 영역으로 프리컴파일 또는 바인드 작업시에 각 SQL문에 대해 할당되고, 작업이 종료되면 반환됩니다. STMTHEAP 데이터베이스 구성 변수를 이용하여 동적으로 조절이 가능합니다.
Application Heap	데이터베이스 접속시	db2agent 프로세스의 개별 작업 영역으로 사용되는 메모리로 에이전트와 서브에이전트 프로세스가 SQL문에 대한 팩키지 중에서 실행 가능한 섹션을 복사하여 저장하는데 사용됩니다. SQL문에 대한 실행 섹션은 요청된 SQL문을 수행하기 위해 필요한 정보를 제공하며, 이 메모리 영역에 캐시된 섹션은 재사용되므로 동일한 SQL문을 수행하는데 필요한 오버헤드를 줄일 수 있습니다. APPLHEAPSZ 데이터베이스 구성 변수를 이용하여 조절이 가능합니다.
Statistics Heap	Runstats 실행시	통계 자료를 갱신하기 위하여 runstats 명령어가 실행될 때만 Utility Heap으로 부터 할당받는 메모리 영역입니다. runstats 명령어에서 with distribution 옵션으로 분산 통계 정보를 수집하는 경우에는 더 많은 메모리를 사용하게 됩니다. STAT_HEAP_SZ 데이터베이스 구성 변수를 이용하여 조절이 가능합니다.
Query Heap	데이터베이스 접속시	db2agent 프로세스가 쿼리문, SQLCA, SQLDA, 펙키지의 이름과 생성자, 섹션 번호, 일관성 토큰 등을 저장하는데 사용됩니다. QUERY_HEAP_SZ 인스턴스 구성 변수를 이용하여 조절합니다.

TOPIC 16 15 Thread 모니터링

Point

9.5이전에는 ps명령어 또는 db2_local_ps 명령어로 Active한 EDU를 리스트할 수 있었습니다. 그러나 9.5부터는 db2sysc밖에는 표시되지 않습니다.

Tip

9.5이후 버전에 해당됩니다.

1 아키텍처가 변경됨에 따라 DBA도 관리방법이 변경됩니다.

```
$ ps -fu db2ins10
    UID     PID    PPID   C    STIME    TTY  TIME CMD
db2ins10 1237176 2109662   0   Feb 28      -  0:12 db2acd 0
db2ins10 1921136 2109662   0   Feb 28      -  0:14 db2sysc 0
db2ins10 2101494 1941686   0 14:22:34  pts/1  0:00 -ksh
db2ins10 2420958 2101494   0 15:25:33  pts/1  0:00 ps -fu db2ins10
```

2 db2sysc가 1921135 입니다. AIX에서 모든 Thread를 확인하려면 아래와 같이 입력합니다.

```
 $ db2pd -edu

Database Partition 0 -- Active -- Up 1 days 01:05:54
List of all EDUs for database partition 0

db2sysc PID: 1921136
db2wdog PID: 2109662
db2acd  PID: 1237176


EDU ID    TID       Kernel TID    EDU Name            USR         SYS
============================================================
    ========================
1801      1801        2216003      db2agent (idle) 0    0.706935    1.071737
1543      1543        5628153      db2resync 0          0.002641    0.004271
1286      1286        1851457      db2ipccm 0           0.082388    0.044037
1029      1029        2023571      db2licc 0            0.000211    0.001055
772       772         4390991      db2thcln 0           0.000244    0.000105
515       515         2621455      db2aiothr 0          2.740874    6.287562
2         2         1273899      db2alarm 0           0.274076    0.408226
258       258         2658531      db2sysc 0            2.085981    1.379128
```

TOPIC 16 16 Memory 사용량 모니터링

Point

다중 스레드 모델에서 메모리 사용량에 대한 모니터링 방법을 소개합니다.

Tip

9.5이후 버전에 해당됩니다.

1 4가지 방법이 제공됩니다.

```
1) db2pd -dbptnmem
2) db2 get snapshot for applications on sample
3) select * from table(admin_get_dbp_mem_usage())
4) db2mtrk -a and db2mtrk -p
```

2 db2pd 사용

```
$ db2pd -dbptnmem

Database Partition 0 -- Active -- Up 1 days 01:11:27

Database Partition Memory Controller Statistics

Controller Automatic: Y
Memory Limit:        13994636 KB
Current usage:       76608 KB
HWM usage:           332736 KB
Cached memory:       16064 KB

Individual Memory Consumers:

Name          Mem Used (KB) HWM Used (KB) Cached (KB)
========================================================
DBMS-db2ins10          46784       46784      10048
FMP_RESOURCES          22528       22528          0
PRIVATE                 7296        7296       6016
```

3 db2 get snapshot 사용

```
$ db2 get snapshot for applications on sample

Memory usage for application:

 Memory Pool Type                  = Application Heap
   Current size (bytes)            = 65536
   High water mark (bytes)         = 65536
   Configured size (bytes)         = 1048576

Agent process/thread ID            = 6463
 Agent Lock timeout (seconds)      = -1
 Memory usage for agent:

  Memory Pool Type                 = Other Memory
    Current size (bytes)           = 196608
    High water mark (bytes)        = 196608
    Configured size (bytes)        = 16710107136
```

TOPIC 16 16 Memory 사용량 모니터링

Point

다중 스레드 모델에서 메모리 사용량에 대한 모니터링 방법을 소개합니다.

Tip

9.5이후 버전에 해당됩니다.

4 SQL 사용

```
$ db2 "select * from table(admin_get_dbp_mem_usage())"
    DBPARTITIONNUM MAX_PARTITION_MEM
    CURRENT_PARTITION_MEM PEAK_PARTITION_MEM ---------- --
    ---------- ---------- ---------- 0 14330507264
    340590592 340852736 1 record(s) selected.
```

5 db2mtrk 사용 (db2mtrk –a 또는 db2mtrk –p)

```
 $ db2mtrk -a
Tracking Memory on: 2008/02/29 at 15:51:00

Application Memory for database: SAMPLE
  appshrh
  128.0K

 Memory for application 546
  apph       other
  64.0K      192.0K

 Memory for application 545
  apph       other
  64.0K      192.0K

 Memory for application 544
  apph       other
  64.0K      320.0K

 Memory for application 543
  apph       other
  64.0K      576.0K

 Memory for application 547
  apph       other
  64.0K      192.0K
```

UNIT 17

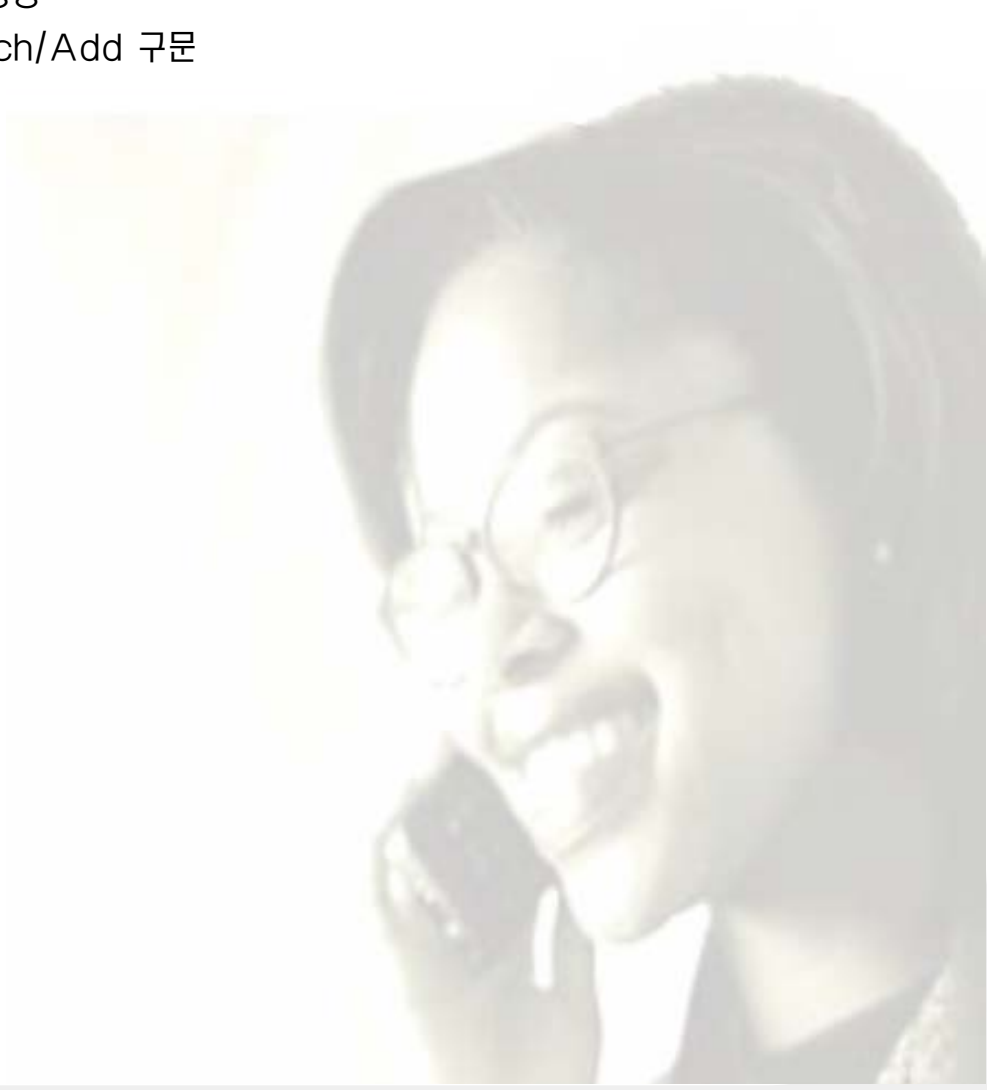
테이블 파티셔닝

대용량 데이터의 처리 시 테이블 파티셔닝은 관리 및 성능상 많은 장점을 줄 수 있습니다. Range 파티션 테이블의 생성 방법과 유지 보수 방법을 소개합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 테이블 파티셔닝 개요
- 파티션 추가
- 파티션 제거
- 파티션 테이블 생성
- Detach/Attach/Add 구문



TOPIC 17 01 테이블 파티셔닝 개요

Point

테이블 파티셔닝은 대용량 테이블의 효과적인 저장 및 처리를 위해 사용되며 데이터의 조회 시 파티션 Elimination을 통하여 성능을 향상 시킬 수 있습니다.

Tip

파티션 테이블의 유연한 유지 보수를 위해서 파티션 별로 별도의 테이블 공간에 저장하는 것이 좋습니다.

Tip

파티셔닝 컬럼은 여러 컬럼으로 구성 될 수 있습니다.

1 테이블 파티셔닝은 다음과 같습니다.

테이블 파티셔닝 은 새로운 범위의 데이터가 계속 추가되어 테이블이 대용량화 되었을 때 Roll-in, Roll-out 기술을 사용하여, 데이터 조회 시 옵티마이저에 의해 불필요한 범위의 스캔을 하지 않음으로 쿼리 성능을 향상시킵니다.

➲ 테이블 파티셔닝

- ➢ 테이블을 범위 단위(일반적으로 주, 월, 분기 또는 년)로 나눕니다.
- ➢ 각 Range는 서로 다른 테이블스페이스에 둘 수 있습니다.
- ➢ Range는 독립적으로 스캔이 가능합니다.
- ➢ ATTACH/DETACH 명령으로 테이블의 roll-in/roll-out이 가능합니다.

일반 테이블 조회

```
SELECT ID, HDATE FROM IBMDB2.SAWON
    WHERE HDATE >= '2009-01-01' AND HDATE < '2010-01-01' ;
```

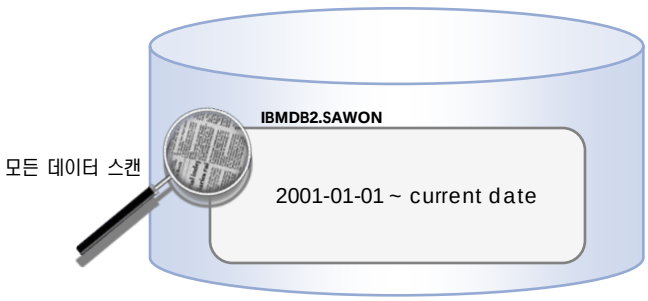


Figure 1701A ••• 일반 테이블 조회 시 모든 데이터 범위 스캔

파티션된 테이블 조회

```
SELECT ID, HDATE FROM IBMDB2.SAWON
    WHERE HDATE >= '2009-01-01' AND HDATE < '2010-01-01' ;
```

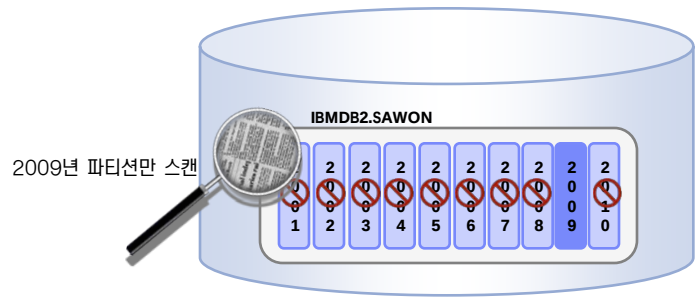


Figure 1701B ••• 파티션된 테이블 조회 시 해당 년도 파티션만 스캔

TOPIC 17 02 파티션 추가

Point

파티션 테이블에 특정 파티션 데이터 추가를 Roll-in이라고 합니다. 관련 명령어는 Attach 입니다.

Tip

Attach 명령을 통해 파티션을 Roll-in 하였다면 SET INTEGRITY 명령을 발행해야 Roll-in 한 데이터를 사용 할 수 있습니다.

Tip

SET LOCK TIMEOUT WAIT 명령을 사용하면 lock 으로 인한 실패로부터 SET INTEGRITY 를 보호할 수 있습니다.

Tip

기회비용을 줄이기 위해 새로운 range partition을 추가 할 때에는 텅빈 파티션으로 Attach 명령을 실행합니다.

1 Roll-in

파티션 된 테이블은 물리적으로 분리된 저장공간에 저장되어 독립된 오브젝트의 데이터를 논리적인 하나의 오브젝트로 만들어진 테이블입니다. 파티션 된 테이블에서 각 파티션을 구분 짓는 range는 기존 데이터와 중첩되지 않는다면 ATTACH 명령으로 새로운 파티션을 Roll-in 할 수 있습니다.

- Rolling in 작업의 장점
  - 테이블의 가용성이 높다. ATTACH 명령의 사용으로 짧은 시간 안에 테이블 사용 가능 합니다.
  - SET INTEGRITY 명령 실행 즉시 모든 데이터를 사용할 수 있습니다.
  - MQTs, constraints, generated columns 를 사용할 수 있습니다.

2 파티션 ATTACH 절차

- ALTER TABLE DETACH PARTITON
  - source와 target 테이블의 Row는 SYSIBM.SYSTABLES 에 X-lock.
  - detach된 파티션의 Row는 SYSIBM.SYSDATAPARTITIONS 에 X-lock.
  - source 테이블에는 Z-lock 이 생성됩니다.
- COMMIT
  - 작업 중 발생된 모든 Lock 이 release 됩니다.
- ONLINE SET INEGRITY
  - ATTACH된 파티션에 Read/Write 가 가능하게 됩니다.

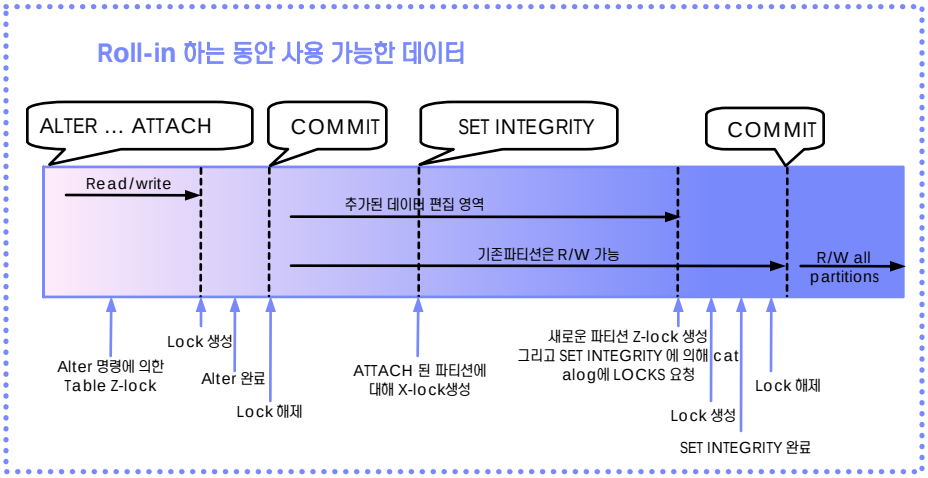


Figure 1702A ••• Roll-in 내부 절차

TOPIC 17 03 파티션 제거

Point

파티션 테이블에 특정 파티션 데이터의 제거를 Roll-out이라고 합니다. 관련 명령어는 Detach 입니다.

Tip

분리된 파티션 테이블은 데이터 변경 이후 다시 Attach될 수 있습니다.

1 Roll-out

파티션 된 테이블에서 기존의 정의된 range 를 DETACH 명령으로 분리 할 수 있습니다. Roll-out된 파티션은 기존 파티션에서 분리되어 사용할 수 없으므로 굳이 DELETE작업을 수행 할 필요가 없습니다.

➲ Rolling out 의 고려사항

- ➢ DETACH 명령이 실행되는 동안 테이블은 offline이 됩니다.
- ➢ 인덱스는 비동기적으로 정리가 된다. 그러므로 따로 인덱스 작업을 따로 수행할 필요가 없습니다.
- ➢ 파티션된 인덱스의 Roll-out 시, 인덱스 클린업 작업이 필요 없습니다.

① 시점 별 가용한 테이블 의 데이터

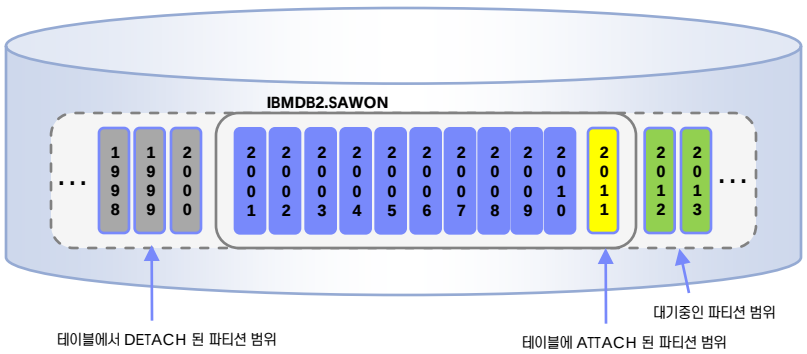


Figure 1703A • • • Roll-out된 테이블

Tip

DETACH, COMMIT 실행 후 SET INTEGRITY 명령은 따로 실행할 필요가 없습니다. 그러나 MQT를 사용할 경우는 SET INTEGRITY ... FULL ACCESS 명령을 실행하여야 이용 가능합니다.

2 파티션 DETACH 절차

➲ ALTER TABLE DETACH PARTITON

- ➢ source와 target 테이블의 Row는 SYSIBM.SYSTABLES에 X-lock.
- ➢ detach된 파티션의 Row는 SYSIBM.SYSDATAPARTITIONS 에 X-lock.
- ➢ source 테이블에는 Z-lock 이 생성됩니다.

➲ COMMIT

- ➢ 작업 중 발생된 모든 Lock 이 release 됩니다.

② Roll-out 하는 동안 사용 가능한 데이터

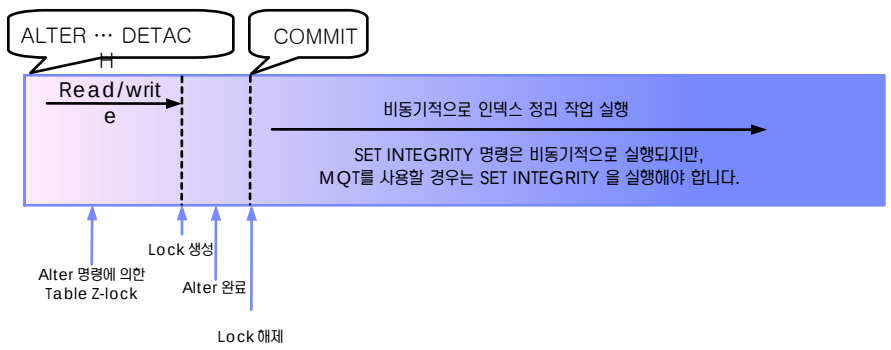


Figure 1703B • • • Roll-out 내부 절차

TOPIC 17 04 파티션 테이블 생성

Point

파티션 테이블의 작성방법은 짧은 구문과 긴 구문으로 작성될 수 있습니다.

Tip

파티션 된 테이블의 각 파티션은 서로 다른 테이블스페이스에 위치시킬 수 있습니다.

Tip

DB2 9.7에서는 서로 다른 파티션에 대해 파티션 인덱스를 생성 할 수 있습니다.

Tip

INCLUSIVE 옵션은 포함된 값이며 EXCLUSIVE 옵션은 제외된 값입니다. STARTING, ENDING에 아무런 표시가 없는 값은 INCLUSIVE 를 의미합니다.

1 파티션 된 테이블 작성

테이블의 데이터는 CREATE TABLE문의 PARTITION BY절에 의해 여러 개의 서로 다른 물리적인 저장 공간에 저장 할 수 있습니다. Partition range는 PARTITION BY절의 STARTING FROM 및 ENDING AT 값으로 지정 할 수 있습니다.

```
CREATE TABLE <table name>
      ...
      IN <table space name1>
      INDEX IN <table space name2>
      LONG IN <table space name3>
   PARTITIONED BY ... PARTITION <partition name> | boundary specification |
                IN <table space name4>
                INDEX IN <table space name5>
                LONG IN <table space name6>
```

2 파티션 테이블 생성을 위한 구문은 아래와 같습니다.

짧은 구문

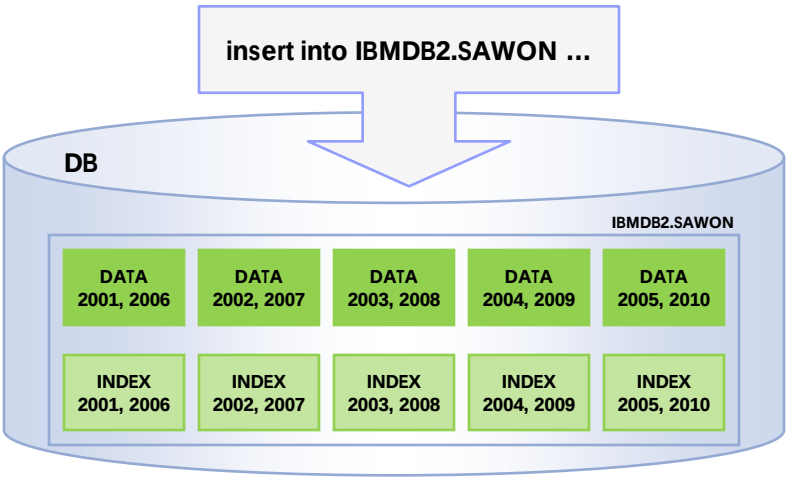
```
CREATE TABLE T1(C1 INT) IN tbsp1, tbsp2, tbsp3
 PARTITION BY RANGE(C1)
 STARTING FROM (1) ENDING (100) EVERY (33)
```

긴 구문

```
CREATE TABLE T1(C1 INT)  PARTITION BY RANGE(C1)
 STARTING FROM (1) ENDING (34) IN tbsp1,
                   ENDING(67)   IN tbsp2,
                   ENDING(100) IN tbsp3
```

➲ 아래는 각 년도 별로 파티션을 만들어 데이터들을 원하는 저장 장소로 모으는 예제입니다.

	1년과 6년	2년과 7년	3년과 8년	4년과 8년	5년과 0년
데이터저장	TSD_IBM01	TSD_IBM02	TSD_IBM03	TSD_IBM04	TSD_IBM05
인덱스저장	TSI_IBM01	TSI_IBM02	TSI_IBM03	TSI_IBM04	TSI_IBM05



TOPIC 17 05 Detach/Attach/Add 구문

Point

파티션 테이블에서 Detach후 Archive를 하거나 새로운 파티션을 추가하는 경우 구문입니다.

Tip

이미 만들어진 새로운 영역의 테이블을 파티션 된 테이블에 삽입 하려면 ATTACH PARTITION 명령을 사용하고 기존에 테이블로 만들어지지 않은 영역은 ADD PARTITION 명령을 사용합니다.

1 Archive 시킬 파티션은 먼저 DETACH합니다.

```
connect to TESTDB;

ALTER TABLE IBMDB2.SAWON
  DETACH PARTITION P_SAWON_2000
  INTO IBMDB2.SAWON_OLD_DETACH;

connect reset;
terminate;
```

2 기존 파티션 테이블에 새로운 파티션의 추가 – 데이터를 가진 기존 테이블인 경우

```
connect to TESTDB;

ALTER TABLE IBMDB2.SAWON
    ATTACH PARTITION P_SAWON_2011
        STARTING FROM ('2011-01-01') INCLUSIVE
        ENDING AT ('2011-12-31') INCLUSIVE
    FROM IBMDB2.SAWON_2011_ATTACH ;

connect reset;
terminate;
```

3 기존 파티션 테이블에 새로운 파티션의 추가 – 빈 파티션만 추가

```
connect to TESTDB;

ALTER TABLE IBMDB2.SAWON
      ADD PARTITION P_SAWON_2012
      STARTING ('2012-01-01') ENDING('2012-12-31') IN TSD_IBM02 INDEX IN
TSI_IBM02 ;

connect reset;
terminate;
```

UNIT 18

데이터 압축

DB2 압축은 테이블의 데이터타입과 관계 없이 데이터, 인덱스, 임시테이블, LOB, XML에 대한 탁월한 압축율 및 성능을 제공합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 데이터 압축
- 인덱스 압축
- LOB, 임시 테이블 압축

TOPIC 1801 데이터 압축

Point

데이터의 압축은 스토리지 비용의 절감뿐만 아니라 성능 향상 및 관리비용도 절감할 수 있게 합니다.

Tip

DB2 9.1이후 데이터의 일부분 압축이 지원됩니다.

Tip

비압축 테이블은 ALTER 문을 이용하여 Compress를 지정할 수 있습니다. 단, Reorg를 수행하여야 Data Dictionary가 빌드되어 이후 압축이 진행됩니다.

Tip

Compress을 했을 경우 Memory와 I/O 사용이 적어져 전체 성능은 향상됩니다.

1 DB2는 Lempel-Ziv(LZ) 기반의 알고리즘을 사용하여 테이블 별 Dictionary를 기반으로 한 Row 레벨 압축 방식을 지원합니다.

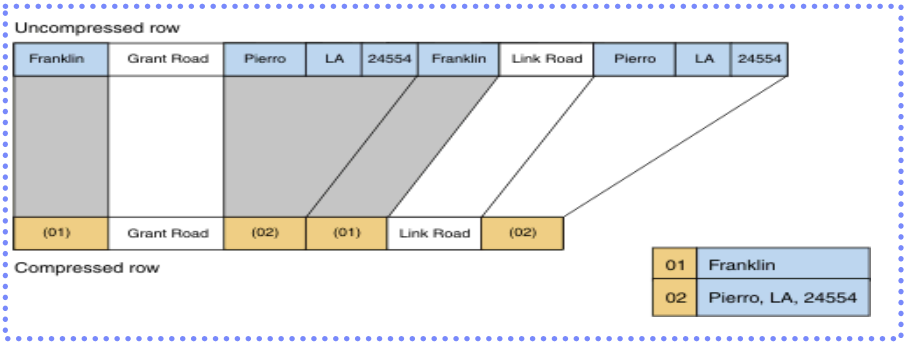


Figure 1801A ••• 데이터 압축 알고리즘

2 압축 활성화

CREATE TABLE 명령어의 "COMPRESS YES" option 을 사용하거나, 또는 기존 테이블에 "ALTER TABLE COMPRESS [YES | NO]" 명령어를 수행합니다.

```
>>-CREATE TABLE--table-name-------------------------------------->

>--+-| element-list |----------------------------+--●----------->
   +-OF--type-name1--+--------------------------+-+
   |                 '-| typed-table-options |-' |
   +-| materialized-query-definition |-----------+
   +-| staging-table-definition |----------------+
   '-LIKE--+-table-name1-+--+------------------+-'
           +-view-name---+  '-| copy-options |-'
           '-nickname----'
                                        .-COMPRESS NO--.
>--●--+---------------------------+--●--+--------------+---------->
      '-| partitioning-clause |-'     '-COMPRESS YES-'
```

3 데이터의 압축 결과 확인 방법은 아래와 같습니다.

ADMIN_GET_TAB_COMPRESS_INFO 함수로 compress로 인하여 절약되는 공간을 추정 및 확인이 가능합니다.

```
SELECT tabname, pages_saved_percent, compress_attr
FROM TABLE (SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO('SIMAP2',
    'STAFF', 'ESTIMATE')) AS T

TAB_NAME    PAGES_SAVED_PERCENT    COMPRESS_ATTR
----------- ---------------------- -------------------------
STAFF       57                      Y
```

Column	설 명
TAB_NAME	테이블 이름
PAGES_SAVED_PERCENT	COMPRESS로 인하여 절약되는 공간
COMPRESS_ATTR	현재 COMPRESS 설정 여부

TOPIC 18 02 인덱스 압축

Point

DB2에서는 인덱스 압축을 위해 세가지의 복합적인 알고리즘을 사용하여 보다 효율적인 압축 성능을 제공합니다.

Tip

Index compress 사용하는 알고리즘은 RID List Compression, Prefix Compression, Variable Slot Directory 세 가지가 있습니다.

Tip

테이블 생성 시 압축 옵션을 YES로 하였다면 인덱스는 기본으로 함께 압축됩니다.

Tip

인덱스의 압축기능은 9.7부터 지원됩니다.

1 사용자가 선택하지 않아도 인덱스 압축은 자동으로 여러 알고리즘을 사용하여 수행됩니다.

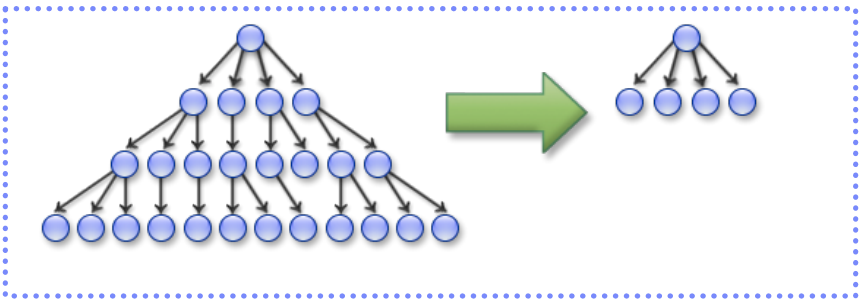
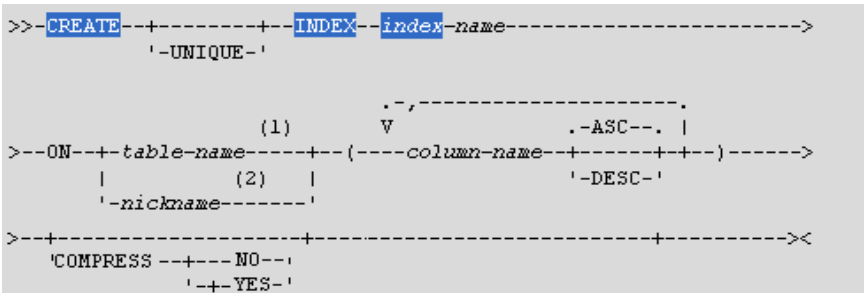


Figure 1802A 인덱스 압축

2 압축 활성화

CREATE INDEX 명령어의 "COMPRESS YES" option 을 사용합니다.

"ALTER INDEX COMPRESS [YES | NO]" 명령어를 수행했을 경우, REORG 실행 시점 이후에 압축이 적용됩니다.



3 인덱스의 압축 결과 확인 방법은 아래와 같습니다.

ADMIN_GET_INDEX_COMPRESS_INFO 함수로 compress로 인하여 절약되는 공간을 추정 및 확인이 가능합니다.

```
SELECT index_name, pages_saved_percent, compress_attr,
    index_compressed
FROM TABLE SYSPROC.ADMIN_GET_INDEX_COMPRESS_INFO('T',
    'myschema', 'T1', '', '')) AS T
```

```
INDEX_NAME  PAGES_SAVED_PERCENT    COMPRESS_ATTR    INDEX_COMPRESSED
----------  ---------------------- ---------------- ----------------
INDEX1      57                     N                N
```

Column	설 명
INDEX_NAME	인덱스 이름
PAGES_SAVED_PERCENT	COMPRESS로 인하여 절약되는 공간
COMPRESS_ATTR	현재 COMPRESS 설정 여부
INDEX_COMPRESSED	현재 INDEX에 적용이 되어 있는지 여부

TOPIC 1803 LOB, 임시 테이블 압축

Point

License를 등록하면 별도의 설정 없이 임시 테이블 압축이 적용됩니다. LOB, XML데이터는 TABLE data와 같은테이블 공간에 위치시켜 압축 가능하게 합니다.

Tip

Storage Optimization 라이센스를 등록하면 자동으로 임시 테이블 압축이 가능해 집니다.

1. 임시 테이블 압축을 통하여 임시 테이블 공간을 절약합니다.User temporary table, system temporary table 모두 포함합니다.

2. db2pd를 사용하여 temporary tablespace를 확인 할 수 있습니다.

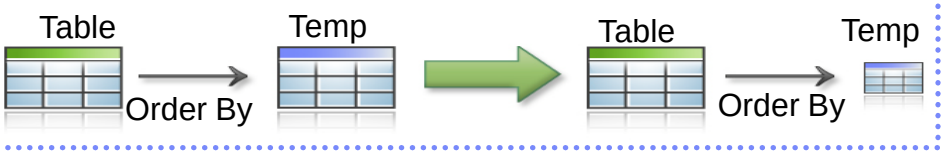


Figure 1803A • • • 임시 테이블 압축 후 공간 절약

3. LOB 데이터를 DATA tablespace와 동일한 tablespace에 넣어 data와 함께 압축이 가능합니다.

4. 테이블 생성 또는 변경할 때 다음과 같이 INLINE을 옵션을 사용합니다.

```
CREATE TABLE ... PICTURE BLOB(10MB) INLINE LENGTH 10000
ALTER TABLE ... ALTER COLUMN PICTURE SET INLINE LENGTH 10000
```

5. INLINE 제한 길이입니다.

Page size	Low size limit	Inline length limit
4k	4005	4001
8k	8101	8097
16k	16293	16289
32k	32677	32673

6. Inline LOB 과 기존 버전에서의 LOB 비교입니다.

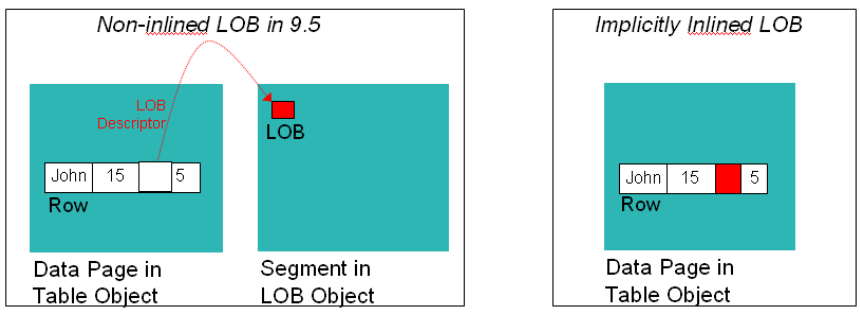


Figure 1803B • • • INLINE LOB

UNIT 19

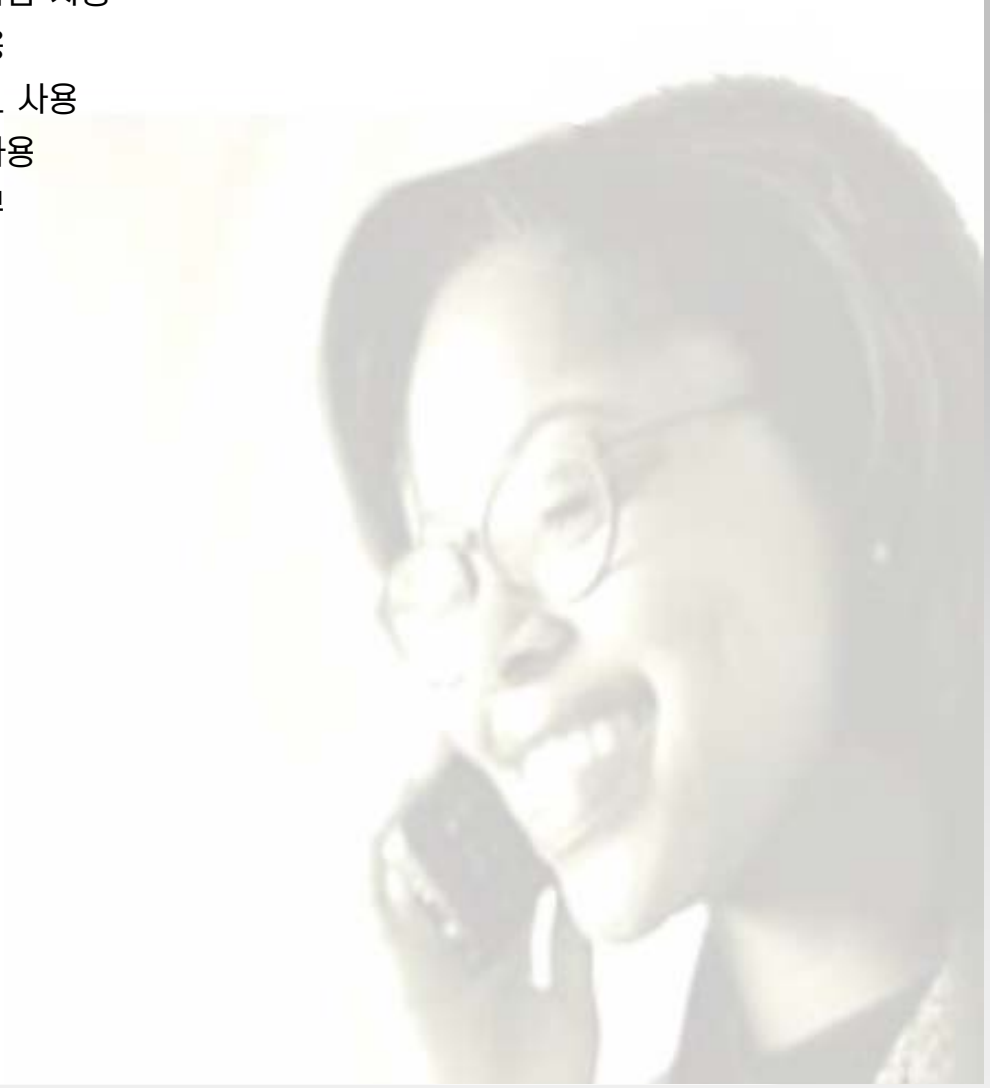
오라클 호환성 지원

DB2 9.7은 오라클의 데이터 타입, SQL, PL/SQL, SQLPLUS등의 애플리케이션을 변경없이 그대로 전환가능한 기능을 제공하고 있습니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 아키텍처 비교
- 오라클에 대한 호환성 지원
- CLPPlus 유틸리티
- 오라클 데이터 타입 사용
- 오라클 함수 사용
- 오라클 PL/SQL 사용
- 오라클 패키지 사용
- 오라클 관리자 뷰



TOPIC 1901 아키텍처 비교

Point

DB2 는 오라클과 다른 아키텍처를 가지고 있습니다. 하지만 용어와 기능에서 차이가 있더라도 DB서비스를 수행함에 있어서 매우 유사합니다.

Tip

일반적으로 오라클의 SGA영역과 DB2의 공유 메모리에서 Buffer pool, Log Buffer, Package Cache등의 순으로 사용 메모리 크기가 작아집니다.

1 아래 그림은 DB2와 오라클의 아키텍처 구성도입니다.

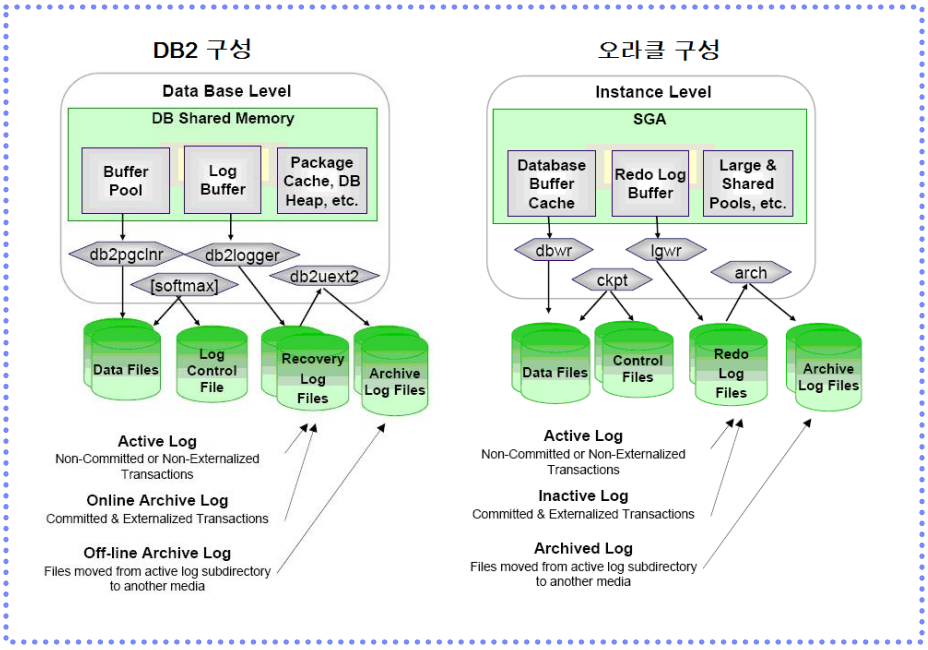


Figure 1901A ••• DB2와 오라클의 아키텍처 비교

2 DBMS의 역할을 수행하기 위해 다양한 프로세서로 구성됩니다.

Oracle	DB2
PMON	db2wdog
SMON	no equivalent
Server	db2agent
DBWx	db2pclnr
CKPT	no equivalent
LGWR	db2loggr
ARCx	no equivalent
no equivalent	db2agntp
no equivalent	db2pfchr

Figure 1901B ••• DB2와 오라클의 프로세스 비교

TOPIC 19 01 아키텍처 비교

Point
관련 오브젝트에 대한 용어에도 약간의 차이가 있습니다.

3 DB2와 오라클의 사용 용어에 차이가 있습니다.

Oracle	DB2
Instance	동일
Database	동일
Initialization File	Database Manager Configuration File
Tablespace	동일
Data Block	Pages
Extents	동일
Data Files	DMS Containers
PL/SQL	SQL/PL, PL/SQL (호환성 지원)
Data Buffer	Buffer Pool
SGA	Database Manager and Shared Memory
Data Dictionary	Catalog
Library Cache	Package Cache
Large Pool	Utility Heap
Data Dictionary Cache	Catalog cache
System tablespace	Syscatspace tablespace

4 오라클과 DB2에서 유사한 기능을 수행하는 명령문입니다.

명령문	Oracle	DB2
Start instance	Startup (pfile =...)	db2start
Stop instance	Shutdown (abort, immediate)	Db2stop (force)
Collect Statistics	Analyze table...	Runstats on table
Load...	Sqlldr	Load from ...
Exp/imp	Export/import	동일
리스너	Lsnrctl start	없음
관리자 모듈	Emctl start dbconsole	db2admin start

TOPIC 19 01 아키텍처 비교

Point
명령문 및 환경변수에도 두 데이터베이스는 다릅니다. 그러나 역할은 대칭됩니다.

5 오라클과 DB2에서 유사한 기능을 수행하는 명령문입니다.

명령문	Oracle	DB2
Backup (오프라인 백업)	DB shutdown 이후 관련 files 복사	db2 backup db <name>
Backup (온라인 백업:DB 별)	RMAN> backup database plus archivelog;	db2 backup db <name> online
Backup (온라인 백업: 테이블 스페이스 별)	alter tablespace <name> begin backup; ... copy ... alter tablespace <name> end backup;	backup database <name> tablespace (userspace1) online to 'path' include logs
Restore (DB 전체)	restore database; Recover database;	restore db <name>
Restore(로그)	Restore archivelog all;	db2 rollforward db <name> to end of logs
백업 히스토리	RMAN> list backup summary;	db2 list backup all for <sample>
아카이브 모드 설정	mount모드>alter database archivelog;	db2 update db cfg for <name> using LOGRETAIN on

6 파라미터 변수 중 오라클과 유사한 DB2 파라미터 수치를 비교합니다.

Oracle	DB2
db_block_size	pagesize
db_write_processes	Num_iocleaners
Local_listener	Svcename , db2comm
Log_buffer	logbufsz
Parallel_max_server	Dft_degree
user_dump_dest	diagpath

TOPIC 19 02 오라클에 대한 호환성 지원

Point

DB2 9.7에서는 오라클 환경 그대로 DB2 환경으로 서비스 이전이 가능합니다. DB2 9.7에서는 다양한 오라클 호환성 기능을 추가하여 오라클 운영자가 마치 오라클을 사용하는 것처럼 DB2 환경에 쉽게 적응할 수 있습니다.

Tip

동시성 제어인 경우에는 DB2 9.7 설치시 오라클 mode를 디폴트로 가지고 있다.

1 오라클과 공통으로 사용되는 호환성 기능입니다.

Oracle	DB2
동시성 제어	기본 지원
SQL 언어	기본 지원
PL/SQL, PL/SQL 패키지, 내장 패키지	기본 지원
JDBC 클라이언트 (확장 포함)	기본 지원
SQL*Plus 스크립트	기본 지원

Tip

설정값을 조회하려면 db2set –all 이나 db2 get cfg | grep compa 로 설정 값을 확인할 수 있습니다.

2 DB2 9.7에서 오라클 호환성 지원을 위한 데이터 유형

데이터 유형	설명
number	Decfloat 및 decimal 을 기반
Varchar2	Null과 후행공백을 구분하는 데이터 포함
Oracle DATE	달력 날짜와 함께 time 구성요소를 포함한다
Timestamp(n)	초단위의 범위를 0(날짜+시간)부터 12(피코초) 사이에 선택할 수 있다.
Boolean	절차적 논리, 변수 및 루틴 매개변수에서 사용할 수 있다.
Varray	프로시저의 array 지원 확장
Index by	배열 지원
Row type	복합 변수 유형 지원
Ref Cursor type	매개변수를 사용하여 커서를 변수에 할당하거나 전달 할 수 있다.

Tip

오라클 관리자 뷰 기능을 사용하시려면 운영자 DB 생성 전에 미리 기능을 설정하셔야 합니다.

TOPIC 1902 오라클에 대한 호환성 지원

Point

DB2 9.7에서는 오라클 환경 그대로 DB2 환경으로 서비스 이전이 가능합니다. DB2 9.7에서는 다양한 오라클 호환성 기능을 추가하여 오라클 운영자가 마치 오라클을 사용하는 것처럼 DB2 환경에 쉽게 적응할 수 있습니다.

Tip

오라클 호환성 기능을 reset하려면 db2_compability_vector= 값을 공란으로 설정하시고 재기동하면 됩니다.

Tip

설정값을 조회하려면 db2set –all 이나 db2 get cfg | grep compa 로 설정 값을 확인할 수 있습니다.

Tip

오라클 관리자 뷰 기능을 사용하시려면 운영자 DB 생성 전에 미리 기능을 설정하셔야 합니다.

3 DB2 9.7에서 오라클 호환성 기능을 전부 설정하는 경우

```
$ db2set DB2_COMPATIBILITY_VECTOR=ORA
  --FFF 값도 동일한 기능 수행
$ db2stop
$ db2start
```

4 DB2 9.7에서 오라클 호환성 지원중 일부 기능만 설정하는 경우 (varchar2 지원 설정 예시)

```
$ db2set DB2_COMPATIBILITY_VECTOR=7
$ db2stop
$ db2start
```

Bit position	Compatibility Feature	Comment
1 (0x01)	ROWNUM	Synonym for ROW_NUMBER() OVER()
2 (0x02)	DUAL	SYSIBM.DUAL
3 (0x04)	(+) Outer join operator	
4 (0x08)	Hierarchical queries	Using CONNECT BY clause
5 (0x10)	NUMBER	
6 (0x20)	DATE	Enables DATE as TIMESTAMP(0)
7 (0x40)	VARCHAR2	
8 (0x80)	TRUNCATE TABLE	
11 (0x400)	Data dictionary-compatible views	Views are created when the database is created
12 (0x800)	PL/SQL Compilation	Enables the compilation and execution of PL/SQL statements and language elements.

Figure 1902A ··· 오라클 호환성 지원 별 설정 값 구별

TOPIC 19 03 CLPPlus 유틸리티

Point

CLPPlus는 오라클의 SQL*Plus를 대신하는 DB2 9.7의 유틸리티입니다.

Tip

ID 입력은 DB 운영자 계정을 입력하시면 됩니다.

1 CLPPlus 명령어 수행 방법

```
$ clpplus  id/password
```

2 CLPPlus에서는 오라클의 SQL*plus 명령어를 그대로 수행합니다.

Tip

CLPPlus는 자바환경변수가 세팅이 되어야 합니다. 실행이 안되는 경우에는 자바 path 여부를 확인 바랍니다.

SQL*plus, CLPplus(DB2)	설명
REM This is a remark example	주석 처리 합니다.
SET termout off **SET linesize 250**	SQL*plus의 set … 명령어를 그대로 사용 가능합니다.
SPOOL myfile_out.txt	Output 파일을 만들어서 작업중인 내용을 저장합니다.
ACCEPT [substitution_variable_name]	키보드로 변수 값을 입력 받습니다.
DEFINE [variable_name]	변수를 지정합니다.
COLUMN [colname] format 9,999 **heading 'My heading'**	컬럼 출력 포맷을 변경합니다.
EXEC [procname]	저장 프로시저를 실행합니다.
CONNECT, DISCONNECT	DB2의 접속과 연결중단을 합니다.
EXIT, QUIT	CLPPlus 세션을 종료합니다.
HELP, ?	도움기능을 호출합니다.
HOST [OS_command]	OS 명령어를 실행합니다.
DESCRIBE [tablename]	오브젝트의 구조 정보를 확인합니다.
EDIT, INPUT, APPEND, CLEAR, etc.	CLPPlus에 할당된 버퍼에 대하여 작업합니다.

Figure 1903A CLPPlus와 SQL*Plus에서 사용 가능한 명령어 리스트

Tip

오라클의 문장 그대로 수행할 때 종결자가 /(백슬래시)로 되어 있는 경우

db2set SQLCOMPAT PLSQL 설정하세요

3 CLPPlus에서 입력 화일명에 있는 명령어 수행결과를 화일로 저장할 수 있습니다.

```
$ db2 -x -td@  -f 입력화일명  -z 출력화일명
```

TOPIC 1904 오라클 데이터 타입 사용

Point

DB2 9.7에서는 오라클 호환 모드를 지원하며 호환 모드 설정 시 오라클의 데이터 타입 number,varchar2,date 사용이 가능합니다. 사용가능한 오라클의 데이터 타입을 함께 살펴보겠습니다.

Tip

db2set –all에서 오라클의 호환성 파라미터 값 (db2_compatibility_vector) 설정을 확인할 수 있습니다.

1 사용 가능한 오라클의 데이터 타입입니다.

오라클	DB2 9.7
number	number decfloat(16) number decimal(p) number(p,2) decimal (p,s) 0<p<32; 0<s<=p
varchar2	varchar Max length 32672
date	Timestamp(0)

Figure 1904A • • • DB2와 오라클의 데이터 타입 비교

2 오라클과 데이터 타입 사용 예시입니다. 샘플 테이블을 생성 합니다.

```
CREATE TABLE emp_test (
empno        NUMBER,
ename        VARCHAR2(10),
hiredate     DATE,
sal          NUMBER(7,2))
```

Tip

오라클 호환 데이터 타입 사용 시 DB2 9.7 에서는 date 타입을 timestamp로 사용합니다. 오라클 호환모드 비설정시와는 다른 값을 보여줍니다.

3 생성한 테이블 결과입니다.

```
SELECT COLNAME,TYPENAME,LENGTH,SCALE
FROM SYSCAT.COLUMNS  where TABNAME ='EMP_TEST'
ORDER BY COLNO


 COLNAME    TYPENAME    LENGTH    SCALE
 ---------- ---------- --------- --------
EMPNO       DECFLOAT    8         0
ENAME       VARCHAR     10        0
HIREDATE    TIMESTAMP   7         0
SAL         DECIMAL     7         2
```

TOPIC 19 04 오라클 데이터 타입 사용

Point

DB2 9.7에서는 오라클 호환 모드를 지원하며 호환 모드 설정 시 오라클의 데이터 타입 number,varchar2,date 사용이 가능합니다. 사용가능한 오라클의 데이터 타입을 함께 살펴보겠습니다.

Tip

boolean 타입은 프로시저 안에서만 제한적으로 사용할 수 있습니다.

4 오라클과 유사한 Syntax을 사용하는 DB2 9.7 데이터 타입을 살펴봅니다.

```
CREATE TYPE emp_type AS ROW (
  ename    VARCHAR(50),
  age      INTEGER);
DECLARE r_emp  emp_type;
```

```
-- links to a column's data type
DECLARE v1 ANCHOR staff.name;

-- links to a table row definition
DECLARE empRow1 ANCHOR ROW OF employee;
```

```
CREATE TYPE genres AS
  VARCHAR(30) ARRAY[4];
```

```
CREATE TYPE empType AS
  INTEGER ARRAY[VARCHAR(50)];
DECLARE r_emp  empType;
```

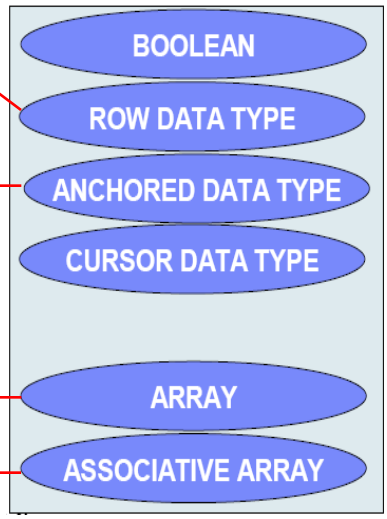


Figure 1904A • • • 오라클과 유사한 데이터 타입 비교

5 오라클과 유사한 Cursor data type 에 대하여 살펴봅니다.

```
CREATE TYPE myRowType AS ROW (edlevel SMALLINT, name VARCHAR(128))@
CREATE TYPE myCursorType AS myRowType CURSOR@
…
CREATE PROCEDURE P_CALLER( IN pempNo VARCHAR(8) ,
                           OUT edlevel SMALLINT,
                           OUT lastname VARCHAR(128))
LANGUAGE SQL BEGIN
   DECLARE c1 myCursorType;
   DECLARE c2 SYSCURSOR;
   CALL P (pempNo,c1);
   …
END@
```

Figure 1904B • • • cusor data type 사용 예시

Tip

느슨한 형 변환은 오라클 호환성 모드 설정과 상관없이 DB2 9.7에서 디폴드로 설정되어 있습니다.

6 DB2 9.7에서는 다른 데이터 타입 비교시 느슨한 묵시적 형 변환을 적용합니다.

대입 : set salary := '52000'
비교: where salary > '52000'
이어 붙히기(concat): 'salary:'|| 52000

TOPIC 19 05 오라클 함수 사용

Point

이전 DB2 버전에서 개발자 정의 함수로 구현한 오라클 함수도 DB2 9.7에서는 오라클 함수 그대로 사용이 가능합니다. 오라클 개발자가 마치 오라클을 사용하는 것처럼 DB2 환경에 쉽게 적응할 수 있습니다.

1 DB2 9.7에서 추가 지원되는 Built-in 함수 목록입니다.

함수 구분	오라클, DB2 함수 명
변환 및 캐시팅 함수	TO_DATE , TO_CHAR, TO_CLOB, TO_NUMBER, TO_TIMESTAMP
날짜 계산	EXTRACT, ADD_MONTHS, MONTHS_BETWEEN, NEXT_DAY, 요일부분을 추가하는 + 부호
문자열 조작	LPAD, RPAD, INSTR, INITCAP, SUBSTR에 대한 확장
기타	NVL,DECODE, LEAST, GREATEST, BITAND

2 오라클의 SQL문 지원 항목을 살펴봅니다.

SQL 구분	설명
CONNECT BY 재귀	LEVEL 및 CONNECT_BY_PATH같은 다양한 재귀 함수를 지원
조인(+) 구문	OUTER JOIN 구문을 사용하는 오라클 구문 지원
DUAL 테이블	단일 행과 단일 열로 구성된 더미 테이블
ROWNUM	리턴되는 행의 수를 제한하고 결과 세트의 행을 열거하는 데 사용
ROWID	오라클에서는 물리적 주소를 기반으로 빠르게 검색하는데 사용하나 DB2 9.7에서는 형식만 지원하고 해당 기능은 미지원
MINUS 연산자	차집합. 기본의 DB2문인 EXCEPT도 사용 가능
SELECT INTO FOR UPDATE	나중에 커서를 사용하지 않고 DB2에서 행을 추출할때 사용
CREATE GLOBAL TEMPORARY TABLE	전역 임시 테이블 생성
TRUNCATE	테이블 내용을 빠르게 삭제

TOPIC 19 05 오라클 함수 사용

Point

이전 DB2 버전에서 개발자 정의 함수로 구현한 오라클 함수도 DB2 9.7에서는 오라클 함수 그대로 사용이 가능합니다. 오라클 개발자가 마치 오라클을 사용하는 것처럼 DB2 환경에 쉽게 적응할 수 있습니다.

Tip

오라클 호환성 기능을 reset하려면 db2_compability_vector= 값을 공란으로 설정하시고 재기동하면 됩니다.

3 오라클의 sysdate를 사용하는 예시 (DB2:current date)

```
select sysdate as ora_compa, CURRENT DATE as db2_origin  from
    dual


 ORA_COMPA              DB2_ORIGIN
 ---------------------  ---------------------
 2009. 8. 10 오후 8:42:29  2009. 8. 10 오후 8:42:29
```

4 오라클의 to_char를 사용하는 경우(DB2: char)

```
select TO_CHAR(sysdate,'YYYY-MM-DD.hh.mm.ss')  as ora_compa ,
    char(current date)  as db2_origin from dual


 ORA_COMPA          DB2_ORIGIN
 -----------------  -------------------
 2009-08-10.10.08.19  2009-08-10-22.08.19
```

5 current date 함수 실행 시 DB2 9.7과 DB2 9.1에서 다른 결과를 보여주는 경우입니다.

```
-- 9.7에서 oracle 호환모드 적용

values (CHAR(current date))

 1
 -------------------
 2009-08-10-22.05.16


-- 9.1에서의 SQL문

values (CHAR(current date))

 1
 ----------
 2009-06-09
```

TOPIC 19 05 오라클 함수 사용

Point

이전 DB2 버전에서 개발자 정의 함수로 구현한 오라클 함수도 DB2 9.7에서는 오라클 함수 그대로 사용이 가능합니다. 오라클 개발자가 마치 오라클을 사용하는 것처럼 DB2 환경에 쉽게 적응할 수 있습니다.

Tip

- 오라클 호환성 설정 여부와 관계없이 오라클 함수의 기능은 DB2에서 사용하는 함수로도 사용이 가능합니다.

6 오라클의 nvl 함수를 사용하는 예시 (DB2: coalesce)

```
SELECT  DEPTNO , MGRNO,NVL(MGRNO, 'ABSENT') as ora_compa,
    COALESCE(MGRNO, 'ABSENT') as db2_origin
   FROM DEPARTMENT


DEPTNO     MGRNO     ORA_COMPA     DB2_ORIGIN
---------  --------  ------------  -------------
A00        000010    000010        000010
...
D01        (null)    ABSENT        ABSENT
```

7 오라클의 decode 함수를 사용하는 예시 (DB2: case문)

```
select deptno, decode (deptno,'B01','AAA','ZZZ' ) as ora_compa,
      case when deptno = 'B01' then 'AAA' else 'ZZZ' end as db2_origin
from department

DEPTNO     ORA_COMPA     DB2_ORIGIN
---------  ------------  -------------
A00        ZZZ           ZZZ
B01        AAA           AAA
C01        ZZZ           ZZZ
```

8 오라클의 rownum 함수를 사용하는 예시 (DB2: fetch first n rows only)

```
select a.empno, a.ora_compa, b.db2_origin
from
(select empno,firstnme as ora_compa  from emp where rownum <=2) a ,
(select empno,firstnme as db2_origin from emp fetch first 2 row only) b
where a.empno=b.empno


EMPNO     ORA_COMPA     DB2_ORIGIN
--------  ------------  -------------
000010    CHRISTINE     CHRISTINE
000020    MICHAEL       MICHAEL
```

TOPIC 19 05 오라클 함수 사용

Point

이전 DB2 버전에서 개발자 정의 함수로 구현한 오라클 함수도 DB2 9.7에서는 오라클 함수 그대로 사용이 가능합니다. 오라클 개발자가 마치 오라클을 사용하는 것처럼 DB2 환경에 쉽게 적응할 수 있습니다.

Tip

오라클 호환성 설정 여부와 관계없이 lag,lead 같은 일부 함수는 오라클 함수 명으로 추가되었습니다.

9 오라클의 last_day,add_months,dayofyear 함수를 사용하는 예시

```
select last_day(current date ) as last_day ,
 add_months (current date , 1) as add_months ,
 dayofyear(current date)
from dual


LAST_DAY              ADD_MONTHS            3
--------------------  --------------------  ----
2009. 8. 31 오후 11:28:48  2009. 9. 10 오후 11:28:48  222
```

오라클의 lower 함수를 사용하는 예시 (DB2: lcase)

```
select deptno,  lower(deptno) as ora_compa  , lcase(deptno) as
    db2_origin from dept

DEPTNO    ORA_COMPA    DB2_ORIGIN
--------  -----------  -------------
A00       a00          a00
B01       b01          b01
C01       c01          c01
D01       d01          d01
```

10 조회데이타를 기준으로 이후 데이터를 가져오는 Lead 함수,

조회데이타를 기준으로 이전 데이터를 가져오는 lag함수를 사용하는 예시

```
SELECT empno,  FIRSTNME,bonus,
     LEAD(bonus,1) OVER (ORDER BY bonus) AS next_sal,
     LAG(bonus,1) OVER (ORDER BY bonus) AS prev_sal
  FROM emp
  WHERE workdept='A00'


EMPNO    FIRSTNME    BONUS    NEXT_SAL    PREV_SAL
-------  ----------  -------  ----------  ----------
000120   SEAN        600      600         (null)
200120   GREG        600      900         600
000110   VINCENZO    900      1000        600
000010   CHRISTINE   1000     1000        900
200010   DIAN        1000     (null)      1000
```

TOPIC 19 05 오라클 함수 사용

Point

이전 DB2 버전에서 개발자 정의 함수로 구현한 오라클 함수도 DB2 9.7에서는 오라클 함수 그대로 사용이 가능합니다. 오라클 개발자가 마치 오라클을 사용하는 것처럼 DB2 환경에 쉽게 적응할 수 있습니다.

Tip

connect by 사용시 depth level이 64이상 초과시에는 with 구문으로 작성 할 것을 권장합니다.

11 오라클의 connect by 함수를 사용하는 예시

```
select level,deptno, deptname,admrdept
FROM  dept
WHERE DEPTNAME NOT LIKE 'BRANCH%'
START WITH DEPTNO=  'E01'
CONNECT BY PRIOR deptno= admrdept


 LEVEL    DEPTNO    DEPTNAME         ADMRDEPT
 -------- --------- ---------------- -----------
 1        E01       SUPPORT SERVICES  A00
 2        E11       OPERATIONS        E01
 2        E21       SOFTWARE SUPPORT  E01
```

12 오라클의 lpad 함수를 사용하는 예시

```
select deptno,  lower(deptno) as ora_compa  , lcase(deptno) as
    db2_origin from dept

 DEPTNO    ORA_COMPA    DB2_ORIGIN
 --------- ------------ -------------
 A00       a00          a00
 B01       b01          b01
 C01       c01          c01
 D01       d01          d01
```

TOPIC 1906 오라클 PL/SQL 사용

Point
오라클의 PL/SQL 도 DB2 9.7에서는 지원합니다.

1 DB2 9.7에서 오라클 PL/SQL 코드는 DB2 엔진의 전용 컴파일러를 사용합니다.

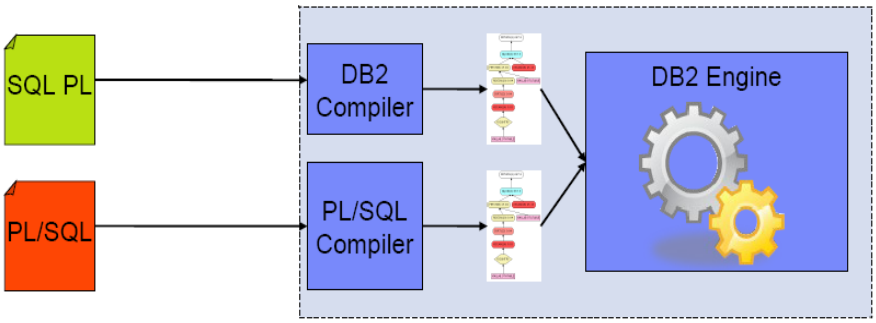


Figure 1906A • • • DB2 서버에서의 PL/SQL 컴파일러

2 DB2 오브젝트에서 or replace 옵션을 사용할 수 있습니다.

```
CREATE [OR REPLACE] FUNCTION
CREATE [OR REPLACE] PROCEDURE
CREATE [OR REPLACE] PACKAGE
CREATE [OR REPLACE] TRIGGER
CREATE [OR REPLACE] VIEW
```

Tip
Or replace 옵션은 해당 오브젝트가 invalid 되는 경우를 방지해 줍니다.

3 세션에서 'PL/SQL의 '/' 종결문자를 사용할 수 있습니다.

```
SET SQLCOMPAT PLSQL;
CREATE OR REPLACE FUNCTION get_browser_version
   ( p_name IN varchar, p_version IN varchar)
   RETURN browsers%TYPE
IS
BEGIN
  IF p_version IS NULL THEN
     RETURN p_name;
  END IF;
  RETURN p_name || '/' || p_version;
END;
/

CREATE PROCEDURE save_browser_version
   (p_ver IN number, p_name IN varchar, p_version IN varchar)
AS
BEGIN
   INSERT INTO browsers
   VALUES (p_ver, get_browser_version(p_name, p_version));
END;
/
CALL save_browser_version(100, 'Firefox', '5.0');
SET SQLCOMPAT DB2;
```

TOPIC 19 06 오라클 PL/SQL 사용

Point

오라클의 PL/SQL 도 DB2 9.7에서는 지원합니다.

Tip

sqlstate 와 sqlcode,get diagnostics 값도 사용이 가능합니다.

4 오라클의 PL/SQL의 데이터 타입 선언과 DB2 9.7에서 새로 추가된 anchor문 비교 예시

```
create or replace procedure emp_output as
declare
 v_empno  emp.empno%TYPE;       -- v_empno  anchor emp.empno;
 v_ename  emp.FIRSTNME%TYPE;   -- v_ename  anchor emp.FIRSTNME;
 v_deptno  emp.workdept%TYPE;   -- v_deptno  anchor emp.workdept;
 v_sal   emp.salary%TYPE;           -- v_sal   anchor emp.salary;
 v_answer   varchar(20);

begin
 select empno, FIRSTNME, workdept, salary,
   case WHEN salary < 2000 THEN 'BAD'
        WHEN (salary > 2000  and salary < 3000) THEN 'GOOD'
       ELSE  'VERYGOOD'
   end
 Into
   v_empno, v_ename, v_deptno, v_sal, v_answer
 from emp
 where empno ='000010';

dbms_output.put_line(v_empno||' '|| v_ename||' '||v_deptno||'
    '||to_char(v_sal)||' '|| v_answer);
end;
/
```

5 커서 처리 조건 처리 예시

```
CREATE OR REPLACE PROCEDURE list_emp
    IS
       v_empno        NUMBER(4);
       v_ename        VARCHAR2(10);
       CURSOR emp_cur IS
          SELECT empno, ename FROM emp ORDER BY empno;
    BEGIN
       OPEN emp_cur;
       DBMS_OUTPUT.PUT_LINE('EMPNO    ENAME');
       DBMS_OUTPUT.PUT_LINE('-----    -------');
       LOOP
          FETCH emp_cur INTO v_empno, v_ename;
          EXIT WHEN emp_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE(v_empno || '    ' || v_ename);
       END LOOP;
       CLOSE emp_cur;
    END
```

3 오라클의 커서 조건 문입니다.

오라클	설명
SQL%NOTFOUND	커서에 데이터가 존재하지 않는 경우
SQL%FOUND	커서의 데이터가 존재하는 경우
SQL%ROWCOUNT	Insert/update/delete 문으로 반영된 데이터 건수

TOPIC 19 07 오라클 패키지 사용

Point

오라클의 패키지도 DB2 9.7에서 사용이 가능합니다. 단지 DB2에서 오라클의 패키지는 모듈(module) 이라고 명명하였습니다.

Tip

Utl.file , dbms_lob같은 일부 오라클 패키지(DB2 모듈)는 비 DPF 환경에서만 지원됩니다.

1 현재 DB2 9.7에서 사용 가능한 오라클 내장 패키지 입니다.

패키지명	설명
DBMS_OUTPUT	명령행에서 설정 또는 해제할 수 있는 기본적인 보고 기능을 제공
UTL_FILE	DB2 서버에서 파일 작업을 수행하는 데 필요한 기능을 제공
DBMS_SQL	기존 EXECUTE 및 EXECUTE IMMEDIATE 명령문과 함께 동적 SQL을 수행할 수 있는 SQL API를 제공하는 패키지
UTL_MAIL	SQL의 이메일 알림을 보낼 수 있는 모듈
UTL_SMTP	UTL_MAIL과 비슷한 하위 수준 API로 SMTP통합 기능을 제공
DBMS_ALERT	다른 세션끼리 서로 세마포어를 설정하는데 사용할 수 있는 패키지
DBMS_PIPE	세션간에 데이터를 보내는데 필요한 기능을 제공하는 모듈
DBMS_JOB	DB2의 작업 스케줄러와 통합되는 호환가능한 API를 제공
DBMS_LOB	DB2의 내장 LOG함수처리를 위함
DBMS_UTILITY	애플리케이션에서 사용하는 프로시저의 모음

Tip

SYSCAT.ROUTINES 에서도 오라클 패키지 리스트을 확인할 수 있습니다.

2 현재 DB2 9.7에서 사용 가능한 오라클 패키지를 쿼리문으로 확인합니다

```
select MODULEID , MODULENAME from SYSCAT.MODULES


MODULEID    MODULENAME
----------  -------------
1           DBMS_OUTPUT
2           DBMS_ALERT
3           DBMS_PIPE
4           DBMS_JOB
5           DBMS_LOB
6           DBMS_SQL
7           DBMS_UTILITY
8           UTL_DIR
9           UTL_FILE
10          UTL_ENCODE
11          UTL_TCP
12          UTL_SMTP
13          UTL_MAIL
14          DBMS_STANDARD
```

TOPIC 19.07 오라클 패키지 사용

Point

오라클의 패키지도 DB2 9.7에서 사용이 가능합니다. 단지 DB2에서 오라클의 패키지는 모듈(module) 이라고 명명하였습니다.

Tip

- 오라클 패키지 형식으로 컴파일 하면 자동적으로 syscat.modules의 dialect 값이 'PL/SQL'로 등록됩니다.

3 오라클 패키지 형식으로 컴파일 했는지, DB2 모듈로 컴파일 되었는지 확인이 가능합니다.

```
SQL> SELECT MODULENAME, DIALECT , MODULETYPE ,  REMARKS
 FROM SYSCAT.MODULES WHERE  MODULESCHEMA = 'SCOTT'


 MODULENAME     DIALECT     MODULETYPE     REMARKS
 -------------  ----------  -------------  --------------------
 EMP_ADMIN      PL/SQL      P              PL/SQL Package Body
 MOD_TEST1      DB2 SQL PL  M              (null)
```

4 오라클 패키지(DB2 모듈)에 어떤 오브젝트가 포함되어 있는지 확인이 가능합니다.

```
SELECT OBJECTMODULENAME , OBJECTNAME
 FROM SYSCAT.MODULEOBJECTS


 OBJECTMODULENAME     OBJECTNAME
 -------------------  ----------------------------------------
 DBMS_ALERT           MAXWAIT
 DBMS_PIPE            MAXWAIT
 DBMS_JOB             ANY_INSTANCE
 DBMS_LOB             DEFAULT_CSID
 DBMS_LOB             DEFAULT_LANG_CTX
 DBMS_LOB             FILE_READONLY
...
```

TOPIC 19-07 오라클 패키지 사용

Point

오라클의 패키지도 DB2 9.7에서 사용이 가능합니다. 단지 DB2에서 오라클의 패키지는 모듈(module) 이라고 명명하였습니다.

5 DBMS_OUTPUT 패키지(DB2 모듈) 사용 예시

```
CREATE OR REPLACE PROCEDURE list_emp
    IS
       v_empno        NUMBER(4);
       v_ename        VARCHAR2(10);
       CURSOR emp_cur IS
          SELECT empno, ename FROM emp ORDER BY empno;
    BEGIN
       OPEN emp_cur;
       DBMS_OUTPUT.PUT_LINE('EMPNO    ENAME');
       DBMS_OUTPUT.PUT_LINE('-----    -------');
       LOOP
          FETCH emp_cur INTO v_empno, v_ename;
          EXIT WHEN emp_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE(v_empno || '    ' || v_ename);
       END LOOP;
       CLOSE emp_cur;
    END
```

6 DBMS_OUTPUT 패키지 수행 결과

```
db2inst1@PROD:~> db2 "set serveroutput on"

→ output 설정 on으로 아래 명령어가 수행됨

CALL DBMS_OUTPUT.ENABLE( 50000 )

db2inst1@PROD:~> db2 "call scott.list_emp“

 Return Status = 0
EMPNO    ENAME
    -----    -------
    7369     SMITH
    7499     ALLEN
    7521     WARD
    7566     JONES
    7654     MARTIN
    7698     BLAKE
    7782     CLARK
…
)
```

Tip

set serveroutput on은 현재 세션에서만 효력이 발생합니다.

TOPIC 19 08 오라클 관리자 뷰

Point

DB2 9.7에서는 오라클 운영자에게 친숙한 DBA_*, ALL_*,USER_* 같은 관리자 뷰를 그대로 사용 할 수 있습니다.

Tip

- S해당 기능을 활용하기 위해서는 운영자 DB를 생성하기 전에 db2_compatibility_vector 설정값은 11이나 호환성 전체 적용을 하셔야 합니다.

Tip

- 오라클 관리자 오브젝트는 syspublic의 alias 나 sysadmin의 view에서 확인 할 수 있습니다.

Tip

- 구체적으로 위치가 지정되지 않은 오브젝트의 path 순서는 values(current path)로 확인할 수 있습니다.

  DB2 9.7에서는

  "SYSIBM","SYSFUN",

  "SYSPROC", "SYSIBMADM","운영자 스키마 "순으로 오브젝트를 검색합니다.

1 오라클 관리자 뷰에 대한 주석을 보려면 sysibmadm.dictionary를 참조하세요

```
$ db2 "select table_name, substr(comments,1,50)  from
    sysibmadm.dictionary"

--------------------  ----------------------------------------------
DBA_ARGUMENTS         Arguments in all objects in the database
ALL_ARGUMENTS         Arguments in objects accessible to the user
USER_ARGUMENTS        Arguments in objects owned by the user
DBA_CATALOG           All database tables, views and synonyms
ALL_CATALOG           All accessible tables, views and synonyms
USER_CATALOG          All user's own tables, views and synonyms
DBA_COL_COMMENTS      Comments on columns of all tables and views
ALL_COL_COMMENTS      Comments on columns of accessible tables and views
USER_COL_COMMENTS     Comments on columns of user's tables and views
DBA_CONS_COLUMNS      Information about all columns in constraint defini
ALL_CONS_COLUMNS      Information about columns in constraint definition
USER_CONS_COLUMNS     Information about columns in constraint definition
DBA_CONSTRAINTS       Constraint definitions on all tables
```

2 오라클에서 자주 사용하는 테이블 리스트 현황 보기 예시 쿼리

```
$db2 "SELECT * FROM TAB"       -- SYSIBMADM.TAB
$db2 "SELECT * FROM TABS"      -- SYSIBMADM.USER_TABLES

TSCHEMA        TNAME                TABTYPE
-------------  -----------          ----------
ADMINISTRATOR  CL_SCHED             TABLE
ADMINISTRATOR  DEPARTMENT           TABLE
ADMINISTRATOR  DEPT                 SYNONYM
ADMINISTRATOR  EMPLOYEE             TABLE
ADMINISTRATOR  EMP                  SYNONYM
ADMINISTRATOR  EMP_PHOTO            TABLE
ADMINISTRATOR  EMP_RESUME           TABLE
ADMINISTRATOR  PROJECT              TABLE
ADMINISTRATOR  PROJ                 SYNONYM
ADMINISTRATOR  PROJACT              TABLE
ADMINISTRATOR  EMPPROJACT           TABLE
```

UNIT 20

워크로드 매니저

데이터의 증가나 업무요건의 증가에 따라 사용자 서비스 응답시간을 만족시키기란 쉽지 않습니다. 이를 위해 WLM은 업무별 사용자 별 필요한 서비스 우선순위를 미리 정의하여 시스템 자원을 적절히 분배함으로써, 보다 만족할 만한 수준의 서비스 응답속도를 제공하고자 합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- WLM 개요
- WLM 정의
- WLM 관리

TOPIC 2001 WLM 개요

Point

DB2의 WLM(Work Load Manager)기능을 이용하여 워크로드 별로 시스템 자원(CPU,Memory등) 사용에 제한을 두거나 작업 우선 순위를 부여할 수 있습니다.

Tip

WLM은 DB2의 Performance Optimization Feature에 포함됩니다.

Tip

DB2 Query Patroller는 WLM으로 대체됩니다.

1 WLM은 다양한 워크로드별로 서비스 클래스를 정의하여 자원 할당에 우선순위를 부여할 수 있게 합니다.

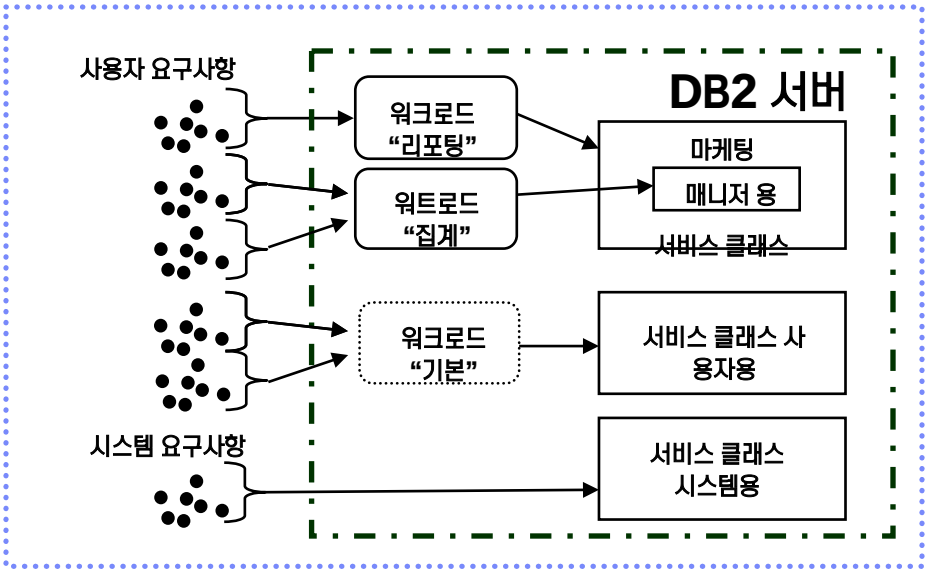


Figure 2001A ••• WLM 구성 예

2 WLM은 시스템 자원 분배 시 우선 순위를 고려하여 서비스 클래스를 분리합니다.

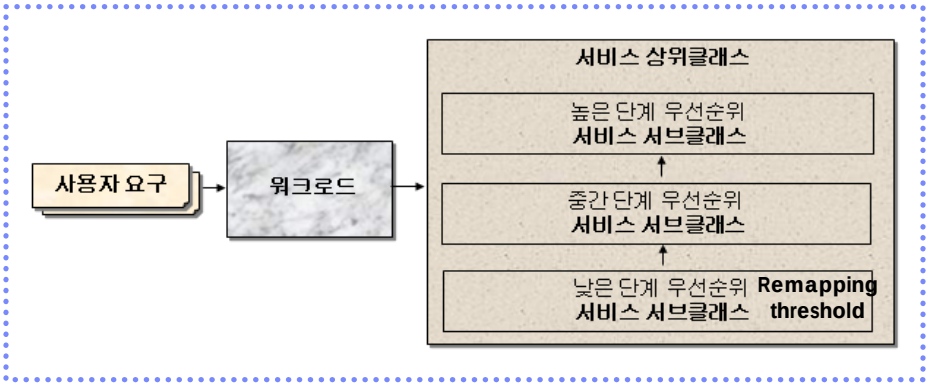


Figure 2001B ••• 서비스 우선 순위 구분

구성요소	설명
서비스 클래스 (Service Class)	시스템 자원 우선 순위를 조정하기 위한 업무 단위, 지정하지 않으면 default 서비스 클래스 사용.
서비스 서브클래스	서비스 클래스의 하위 개념으로 WLM에서 activity 데이터를 수집
워크로드 (Workload)	데이터베이스 연결시 연결 속성(appname, system_user 등) 에 따른 업무 제어, 특정 서비스 클래스에 할당됨
Threshold	업무가 수행되는 동안 사용될 시스템 자원의 허용 범위 지정. 허용된 범위를 초과할 경우 지정된 문장이 실행됩니다.
Work Class, Work Activity	연결속성이 아닌 실행 유형(read, write, call등)에 따라 임계값 설정 및 특정 서비스 클래스로 매핑합니다.

TOPIC 20 02 WLM 정의

Point

WLM을 정의하는 절차에 대한 소개입니다.

1 WLM의 구축 절차 는 아래와 같습니다.

```
Step1 : 서비스 클래스 생성
Step2 : 서비스 서브클래스 생성
Step3 : Workload 생성
Step4 : Threshold 생성
Step5 : Work Class set 생성
Step6 : Work Action set 생성
```

2 Step1: 서비스 클래스를 생성합니다.

```
Create service class ACT_CNTL_SC  --class name
```

3 Step2: 서비스 서브클래스를 생성합니다.

```
CREATE SERVICE CLASS ACT_CNTL_SUBC  -- 서브 클래스명
UNDER ACT_CNTL_SC – 클래스 명
COLLECT ACTIVITY DATA WITH DETAILS;   -- 데이터 모음 옵션
```

Tip

응용 프로그래 명은 반드시 대소문자를 구분하여 사용하며 사용자 명은 반드시 대문자를 사용할 것을 권장합니다.

4 Step3: Workload를 생성합니다.

1) 사용자명을 통한 DB 사용 제한을 위한 WORKLOAD 정의합니다.

2) 응용 프로그램 명을 통한 DB 사용 제한을 위한 WORKLOAD를 정의합니다.

```
CREATE WORKLOAD SELECT_USER_WL
        SYSTEM_USER ('SELUSR')
        SERVICE CLASS ACT_CNTL_SC;

CREATE WORKLOAD TOAD_AP_WL
        APPLNAME ('Toad.exe')
        SERVICE CLASS ACT_CNTL_SC;
```

TOPIC 20 02 WLM 정의

Point

WLM을 정의하는 절차에 대한 소개입니다.

5 Step4: Threshold를 생성합니다.

-- 수행 시 지정된 시간 이상 수행되는 SQL문을 수집 후 해당 연결을 강제 종료시킵니다.

-- Threshold-domain을 DB 전체로 정의하며, 임계 값 초과시에도 계속 수행하고 해당 SQL문을 수집합니다.

```
CREATE THRESHOLD LONG_EXEC_AP_TH
      FOR DATABASE ACTIVITIES
      ENFORCEMENT DATABASE
       WHEN ACTIVITYTOTALTIME > 1 HOURS
       COLLECT ACTIVITY DATA WITH DETAILS
       CONTINUE;
```

9 Step5: Work Class set를 생성합니다.

-- LOAD 작업에 대한 WORK CLASS SET을 정의합니다.

```
CREATE WORK CLASS SET LOAD_WCS
      (WORK CLASS LOAD_WC WORK TYPE LOAD);
```

10 Step6: Work Action set를 생성합니다.

-- LOAD 작업이 시도될 경우 수행될 ACTION을 정의합니다.

```
CREATE WORK ACTION SET LOAD_WAS
FOR DATABASE
USING WORK CLASS SET LOAD_WCS
 (WORK ACTION LOAD_WA ON WORK CLASS
   LOAD_WC PREVENT EXECUTION);
```

TOPIC 20 03 WLM 관리

Point

정의된 WLM에 대한 권한 부여 방법 및 비활성화 그리고 모니터링 방법입니다.

Tip

정의된 워크로드를 사용하기 위해서는 해당 workload에 대한 사용권한이 있어야 합니다. 권한이 없는 경우 기본 워크로드에 할당됩니다.

1 Workload에 대한 권한을 부여합니다.

```
GRANT USAGE ON WORKLOAD TOAD_AP_WL TO PUBLIC;
GRANT USAGE ON WORKLOAD WAS_AP_WL TO USER WASUSR;
GRANT USAGE ON SELECT_USER_WL TO USER SELUSR;
```

2 Workload에 대한 권한을 제거 합니다.

- 권한 제거 명령 수행 후 모든 연결에서 SQL문을 수행할 경우 다음 에러 발생합니다.

SQL4707N Workload "SYSDEFAULTUSERWORKLOAD" cannot service the request because it is not allowed to access the database or is disabled. SQLSTATE=5U020

```
REVOKE USAGE ON WORKLOAD TOAD_AP_WL TO PUBLIC;
REVOKE USAGE ON WORKLOAD SELECT_USER_WL FROM SELUSR;
```

3 Workload에 대한 비활성화

- 해당 WORKLOAD가 비활성화된 후에 SQL문 수행 시 아래 에러 발생합니다.

SQL4707N Workload "SYSDEFAULTUSERWORKLOAD" cannot service the request because it is not allowed to access the database or is disabled. SQLSTATE=5U020

```
ALTER WORKLOAD TOAD_AP_WL DISABLE;
ALTER WORKLOAD SELECT_USER_WL DISABLE;
```

4 SERVICE CLASS 또는 SERVICE SUBCLASS 비활성화

- SERVICE CLASS 또는 SERVICE SUBCLASS를 비활성화할 경우 관련된 모든 WORKLOAD가 사용 불가능하게 되므로 주의해야 합니다.

SQL4714N The request cannot be executed because service class "ACT_CNTL_SC.SYSDEFAULTSUBCLASS" is disabled. SQLSTATE=5U028

```
ALTER SERVICE CLASS ACT_CNTL_SC DISABLE;
```

Tip

WLM 관련 이벤트 모니터 생성 스크립트는 "./sqllib/misc/wlmevmon.dll'에 있습니다.

5 WLM 현재 상태와 누적 모니터링을 하기 위해서는 db2pd 명령에서 wlm 옵션을 사용하거나 또는 wlm_get_service_superclass_stats같은 WLM 상태조회 테이블 함수를 사용할 수 있습니다.

```
SELECT SUBSTR(WORK_ACTION_SET_NAME, 1, 12),
       SUBSTR(WORK_CLASS_NAME, 1, 12), LAST_RESET,
       SUBSTR(CHAR(ACT_TOTAL), 1, 10)
FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS('', -2))
ORDER BY WORK_ACTION_SET_NAME, WORK_CLASS_NAME ;
```

UNIT 21

XML 데이터 관리

DB2는 관계형 데이터 뿐만 아니라 XML 데이터를 Native로 저장 및 처리하는 Hybrid 데이터베이스입니다. XML 데이터 타입을 지원하여 XML데이터의 저장 및 XML쿼리를 가능하게 합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- XML 구조 및 데이터베이스
- XML 테이블 및 인덱스 생성
- XML 쿼리문
- XML 쿼리문-XPath
- XML 쿼리문-Xquery
- XML 쿼리문-SQL/XML

TOPIC 2101 XML 구조 및 데이터베이스

Point

DB2는 XML데이터를 Native로 저장하는 Pure XML을 지원함으로써, XML데이터의 관리 및 성능상 매우 유리한 기능을 제공하고 있습니다.

Tip

Pure XML의 장점

1) XML 함수의 활용으로 프로그램 코드의 복잡성을 줄입니다.
2) 스키마 변경작업등 XML 문서 관리 작업이 쉽습니다.
3) XML 애플리케이션에서 많은 성능 향상을 얻을 수 있습니다.

1 9.1이전 XML 데이터가 데이터베이스에 저장하는 방식입니다.

-A type: XML 문서가 데이터베이스의 clob 또는 varchar 컬럼에 저장됩니다.

-B type: XML문서가 테이블에 단편화 부분으로 나뉘어서 저장됩니다.

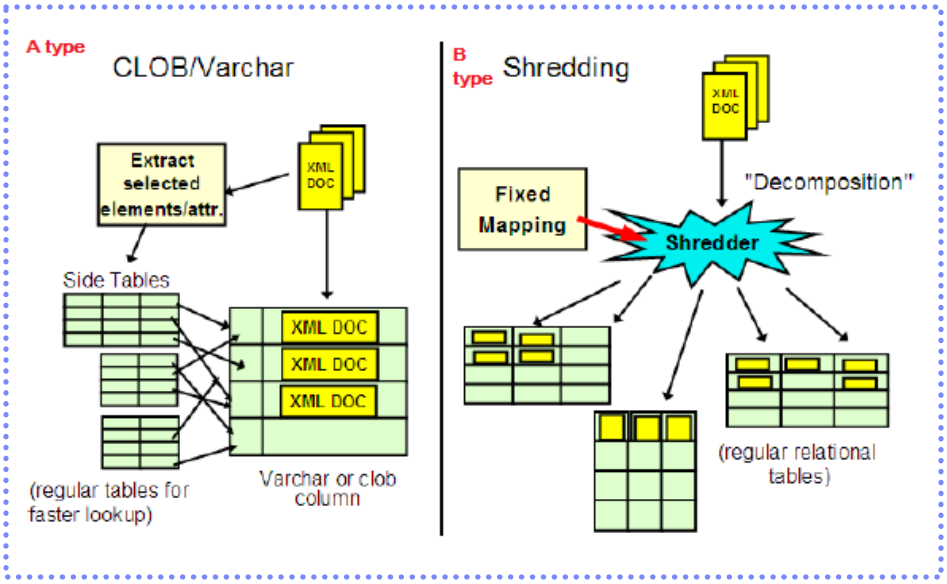


Figure 2101A ··· 9.1 이전 XML 문서 저장 방식

Tip

Sample DB를 이용하여 XML 데이터를 생성할 때는 db2sampl –sql –xml 옵션을 사용합니다.

2 9.1 이후, 하나의 테이블은 관계형 데이터와 계층적 데이터로 나누어서 저장됩니다.

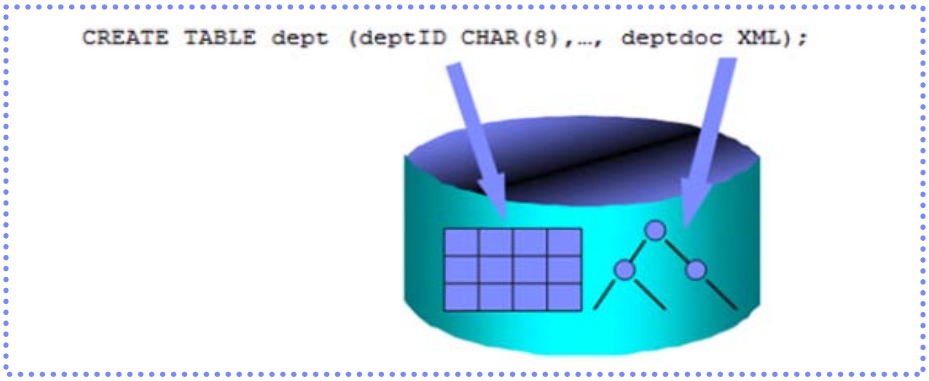


Figure 2101B ··· 9.1 이후 XML 컬럼이 있는 테이블 구조

TOPIC 21 02 XML 테이블 및 인덱스 생성

Point

XML 데이터는 XML 데이터타입으로 저장된 후 일반 테이블과 같이 인덱스를 생성할 수 있습니다.

Tip

XML 인덱스는 단일 컬럼에서만 적용됩니다. 결합인덱스는 사용할 수 없습니다.

Tip

XML 인덱스가 가능한 타입은 DOUBLE,VARCHAR(n),VARCHAR HASHED, DATE ,TIMESTAMP이 있습니다.

Tip

모든 텍스트 노드값에 인덱스를 생성하는 것은 인덱스 사이즈가 필요이상으로 커지게 되므로 권장하지 않습니다.

1 테이블 생성을 위한 샘플 XML 문장입니다.

```
<customerinfo Cid="1004">
<name>Matt Foreman</name>
<addr country="Canada">
<street>1596 Baseline</street>
<city>Toronto</city>
<state>Ontario</state>
<pcode>M3Z-5H9</pcode>
</addr>
<phone type="work">905-555-4789</phone>
<phone type="home">416-555-3376</phone>
<assistant>
<name>Peter Smith</name>
<phone type="home">416-555-3426</phone>
</assistant>
</customerinfo>
```

2 테이블 생성 문입니다.

```
CREATE TABLE customer(info XML)
```

2 인덱스 생성 문입니다. XML 문의 element, attribute, 그리고 text node에 대해서 인덱스를 생성할 수 있습니다.

```
-- Attribute Cid에  인덱스 생성

CREATE UNIQUE INDEX idx1 ON customer(info)
GENERATE KEY USING
xmlpattern '/customerinfo/@Cid'
AS sql DOUBLE

-- Element name에 인덱스 생성

CREATE INDEX idx2 ON customer(info)
GENERATE KEY USING
xmlpattern '/customerinfo/name'
AS sql VARCHAR(40)

-- 모든 elements name에 인덱스 생성

CREATE INDEX idx3 ON customer(info)
GENERATE KEY USING
xmlpattern '//name'
AS sql VARCHAR(40);
```

TOPIC 21 03 XML 쿼리문

Point

XML문서를 파싱된 계층형 포맷으로 이미 저장하여 쿼리 수행 시 XML 파싱이 필요 없이, Xquery,Xpath,SQL/XML를 이용할 수 있습니다.

1 XML 문의 트리 구조에서 document, element, attribute, text, comment 노드별 구조입니다.

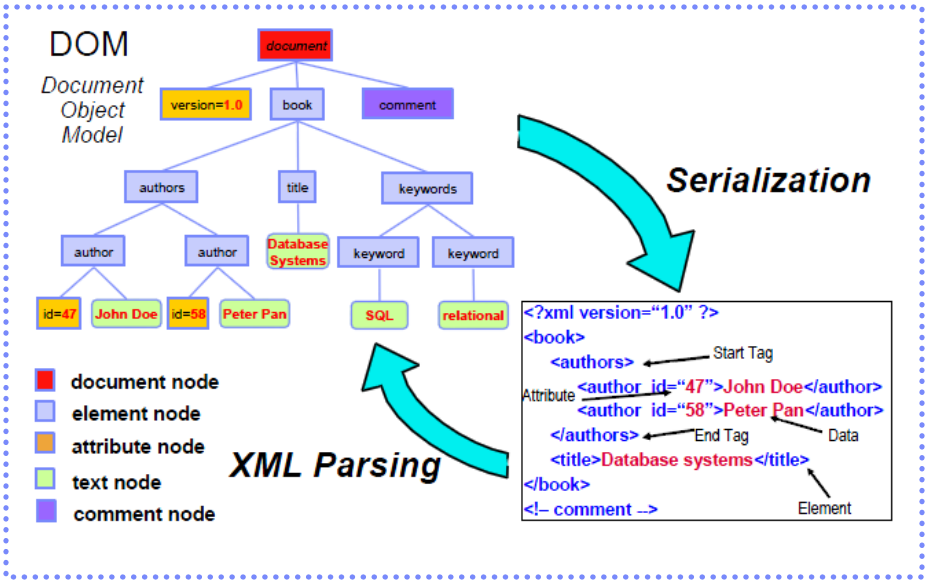


Figure 2103A • • • DB2의 XML 문 트리 구조

2 SQL/Xml문과 XQuery는 각각 ISO 및 W3C에서 인증하는 언어로써, 사용법이 다릅니다.

-- SQL : 관계형 데이터에 접근하기 위한 표준 언어입니다.
-- Xquery : XML 쿼리언어로 W3C에서 인증하는 언어입니다.
-- SQL/XML : SQL과 XML를 같이 다루는 ISO에서 인증하는 표준 언어입니다.

Tip

Before 트리거 문으로 문서 validation을 확인할 수 있습니다.

3 SQL/Xml문에서 사용하는 주요 함수입니다.

함수 이름	설명
XMLPARSE	XML 값을 만들기 위해 character 형이나 오브젝트 바이너리 데이터 형식으로 파싱합니다.
XMLSERIALIZE	Character 또는 Large 오브젝트 형식으로 XML value 값을 변환합니다.
XMLVALIDATE	디폴트값을 포함하여 XML 스키마 유효성 확인에서 확보한 정보에 의해 증가된 입력 XML 값의 사본을 리턴합니다
XMLEXISTS	업무가 수행되는 동안 사용될 시스템 자원의 허용 범위 지정. 허용된 범위를 초과할 경우 지정된 문장이 실행됩니다.
XMLQUERY	연결속성이 아닌 실행 유형(read, write, call등)에 따라 임계값 설정 및 특정 서비스 클래스로 매핑합니다.

TOPIC 2104 XML 쿼리문 - XPath

Point

XPath는 XML문서를 조회하는데 사용하는 언어입니다. XML의 attibute와 element를 상세하게 표현할 수 있습니다.

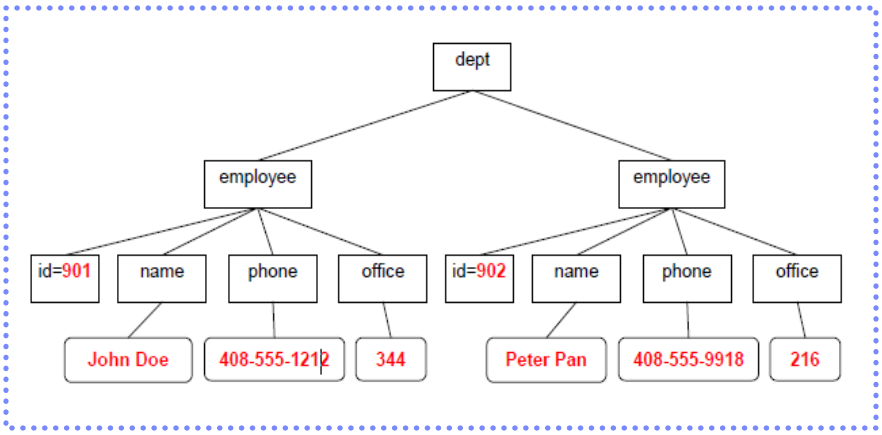
Tip

응용 프로그램명은 반드시 대소문자를 구분하여 사용하며 사용자 명은 반드시 대문자를 사용할 것을 권장합니다.

1 XPath를 사용하기 위해 XML 문서 예제를 살펴보겠습니다.

```
<dept bldg="101">
<employee id="901">
<name>John Doe</name>
<phone>408 555 1212</phone>
<office>344</office>
</employee>
<employee id="902">
<name>Peter Pan</name>
<phone>408 555 9918</phone>
<office>216</office>
</employee>
</dept>
```

2 위의 문서를 구조적으로 표현하면 아래와 같습니다.



3 XPath를 사용하여 아래와 같이 결과값을 얻을 수 있습니다.

XPath	결과
/dept/@bldg	101
/dept/employee/@id	901 902
/dept/employee/name	<name>Peter Pan</name> <name>John Doe</name>
/dept/employee/name/text()	Peter Pan John Doe

TOPIC 2104 XML 쿼리문 - XPath

Point

XPath에서 조건문 사용 방법입니다.

Tip

- Xpath와 Xquery는 대소문자를 구분합니다.

4 XPath에서 "[]"를 사용하면 SQL문에서 where 구문을 사용하는 것처럼 조건문으로 사용할 수 있습니다.

XPath	결과
/dept/employee[@id="902"]/name	<name>Peter Pan</name>
/dept[@bldg="101"]/employee[office >"300"]/name	<name>John Doe</name>
//employee[office="344" OR office="216"]/@id	901 902
/dept/employee[2]/@id	902

5 XPath에서 "." 은 현재 위치를, ".."는 상위 정보를 알려주는 데 사용합니다.

XPath	결과
/dept/employee/name[../@id="902"]	<name>Peter Pan</name>
/dept/employee/office[.>"300"]	<office>344</office>
/dept/employee[office > "300"]/office	101
/dept/employee[name="John Doe"]/../@bldg	101

TOPIC 21 05 XML 쿼리문 - Xquery

Point

Xquery는 XML를 사용하기 위한 쿼리문 입니다. SQL문의 select – from- where 표현식처럼 Xquery에서는 FLWOR 표현식을 사용합니다.

1 FLOWR 표현식은 아래와 같습니다.

XPath	설명
FOR:	시퀀스를 반복하여 변수를 아이템에 바인딩
LET:	하나의 변수를 시퀀스에 바인딩
WHERE:	반복 아이템에 대한 제한 조건
ORDER:	반복 아이템을 재정렬
RETURN:	쿼리 결과값을 구성

2 Xquery 사용 예제입니다.

조회문)

```
xquery
for $d in db2-fn:xmlcolumn('dept.deptdoc')/dept
let $emp := $d//employee/name
where $d/@bldg > 95
order by $d/@bldg
return
<EmpList>
{$d/@bldg, $emp}
</EmpList>
```

조회결과)

```
<EmpList bldg="101">
<name>
John Doe
</name>
<name>
Peter Pan
</name>
</EmpList>
```

TOPIC 21 06 XML 쿼리문 – SQL/XML

Point

SQL/XML 함수를 사용하여 XML 데이터를 조회하는 방법입니다.

1 Xmlexists 사용 예입니다.

```
-- xmlexists 사용 예제

SELECT name FROM clients
WHERE xmlexists(
'$c/Client/Address[zip="95116"]'
passing clients.contact as "c"
)
```

2 Xmlquery 사용 예입니다.

```
SELECT xmlquery('$c/Client/email' passing contact as "c")
FROM clients
WHERE status = 'Gold'
```

3 Xmltable 사용 예입니다.

```
SELECT t.comment#, i.itemname, t.customerID, Message
FROM items i,
xmltable('$c/Comments/Comment' passing i.comments as "c"
columns Comment# integer path 'CommentID',
CustomerID integer path 'CustomerID',
Message varchar(100) path 'Message') AS t
```

Tip

잦은 쿼리문은 뷰를 작성하여 사용하시면 편리합니다.

4 Clients 테이블의 contract xml 컬럼의 데이터 조회

```
xquery db2-fn:xmlcolumn('CLIENTS.CONTACT'))
→위 문장의 경우 "SELECT contact FROM clients" 와 동일합니다.
```

5 클라이언트 팩스 데이터 조회

```
xquery
for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/fax
return $y
```

6 두 테이블(dept, unit)에 대해서 조인문 쿼리

```
SELECT u.unitID
FROM dept d, unit u
WHERE XMLEXISTS (
'$e//employee[name = $m]'
passing d.deptdoc as "e", u.manager as "m")
```

UNIT 22

SQL 컴파일러

DB2는 시스템 카탈로그의 통계 자료를 이용하여 비용 기반의 옵티마이저를 제공합니다. 요청된 SQL문은 효율적인 액세스 플랜을 위하여 컴파일 과정에서 재작성됩니다. 액세스 플랜은 익스플레인 테이블, Visual Explain, db2expln, db2exfmt 등을 이용하여 확인합니다.

DB2 9.7 개발자 가이드

Developer Edition

- SQL 컴파일러
- 최적화 클래스
- 최적화 클래스 지정 방법
- 쿼리 재작성
- 뷰 병합
- 서브쿼리에서 조인으로의 변환
- 중복 조인 제거
- 공유 총계
- DISTINCT 제거
- 일반 술어 푸시다운
- 상관 해제
- 암시적 술어
- OR에서 IN으로의 변환
- 패키지
- 익스플레인 도구
- Visual Explain
- db2expln 유틸리티
- 동적 SQL문에 대한 db2expln 출력
- 정적 SQL문에 대한 db2expln 출력
- db2exfmt 유틸리티
- 동적 SQL문에 대한 db2exfmt 출력
- 정적 SQL문에 대한 db2exfmt 출력

TOPIC 2201 SQL 컴파일러

Point

비용 기반 (Cost-based)의 최적화기(Optimizer)를 이용하여 요청된 SQL문에 대한 최소 비용의 액세스 플랜을 선택하는 과정을 SQL 컴파일이라고 합니다. 옵티마이저는 필요에 따라 요청된 SQL문을 보다 효율적인 SQL문으로 재작성하기도 합니다.

Tip

쿼리 그래프 모델(QGM, Query Graph Model)은 SQL문의 컴파일 과정 동안 사용되는 내부적인 데이터베이스로 메모리에 저장됩니다.

1 SQL 컴파일러는 여러 단계를 수행하여 입력된 SQL문을 컴파일하고, 실행할 수 있는 액세스 플랜을 생성합니다. 비용 기반의 옵티마이저를 이용하여 최적의 액세스 플랜을 결정하게 됩니다.

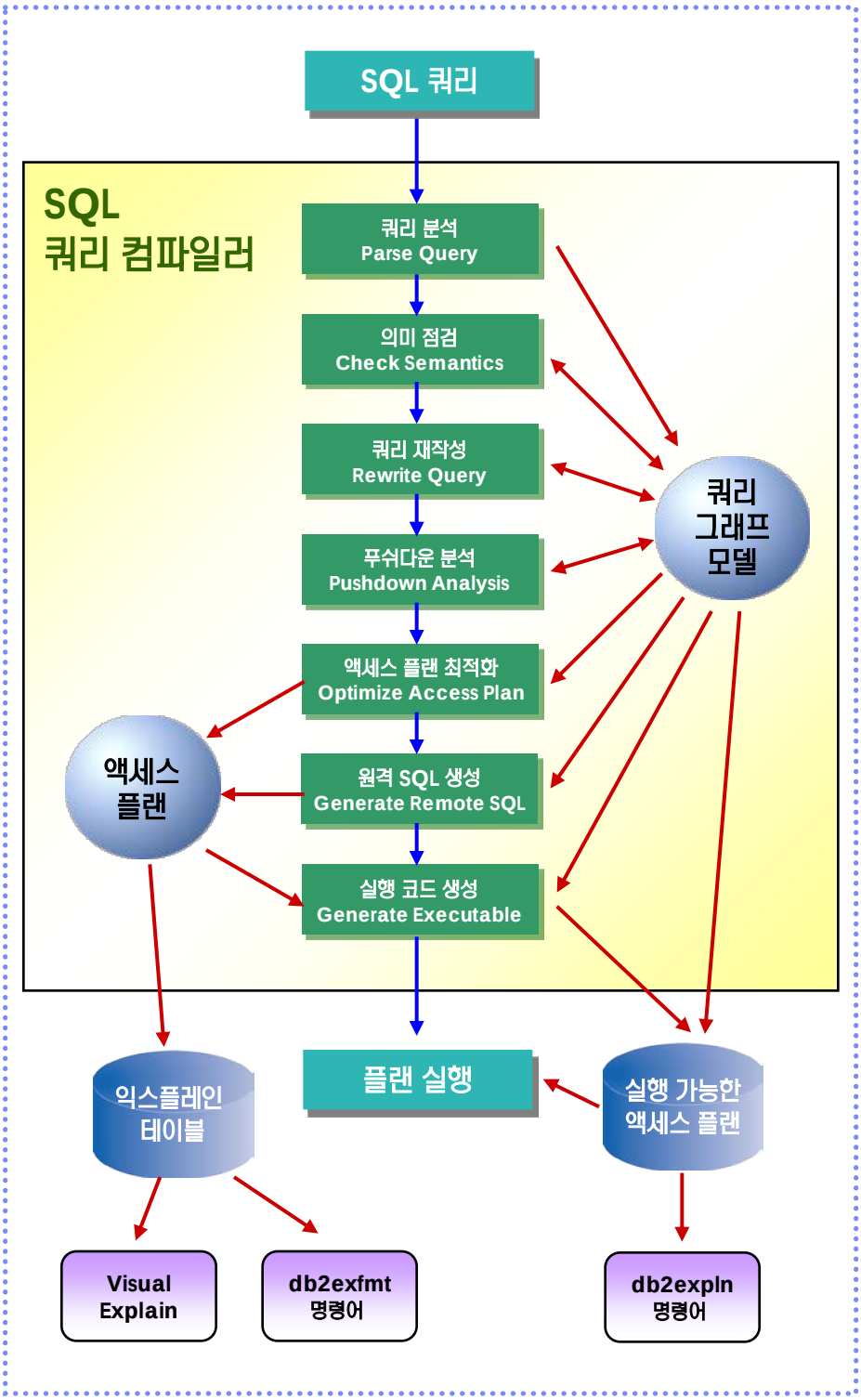


Figure 2201A SQL 컴파일 과정

TOPIC 22-01 SQL 컴파일러

Point

비용 기반 (Cost-based)의 최적화기(Optimizer)를 이용하여 요청된 SQL문에 대한 최소 비용의 액세스 플랜을 선택하는 과정을 SQL 컴파일이라고 합니다. 옵티마이저는 필요에 따라 요청된 SQL문을 보다 효율적인 SQL문으로 재작성하기도 합니다.

Tip

옵티마이저가 가장 효율적인 액세스 플랜을 작성하려면 테이블 및 해당 데이터에 대한 최신 통계 정보가 필요하므로, 자주 변경하는 테이블에 대해서 적절한 간격으로 RUNSTATS를 실행하여 최신 통계 자료를 유지합니다.

RUNSTATS는 automatic으로 운영이 가능합니다. 수동으로 운영해야 할 이유가 없다면 automatic으로 설정하시기 바랍니다.

Tip

조인의 유형에는 Nested Loop 조인, Merge Scan 조인, Hash 조인이 있고, 액세스 유형에는 테이블 스캔과 인덱스 스캔이 있습니다.

Tip

'푸쉬다운 분석'과 '원격 SQL 생성' 단계는 Federated 된 데이터베이스에만 적용됩니다.

Tip

시스템의 통계 자료가 변경되거나, 새로운 인덱스가 추가된 경우에는 정적 SQL문에 대한 액세스 플랜을 명시적으로 재생성하도록 합니다.

2 SQL 컴파일러가 실행하는 각 과정에 대한 설명은 다음과 같습니다.

과 정	설 명
쿼리 구문 분석 Parse Query	SQL 쿼리를 분석하여 구문의 유효성을 확인합니다. 구문 오류가 검출되면, 처리를 중단하고 해당 SQL 오류를 리턴합니다. 쿼리 구문 분석이 완료되면, 쿼리에 대한 내부적인 표현이 QGM에 작성되고 저장됩니다.
의미 점검 Check Sementics	명령문의 각 부분에 의미상 불일치되는 부분이 있는지 확인합니다. 참조 제한 조건, 테이블 점검 제한 조건, 트리거와 뷰 등의 작동 의미를 QGM에 추가합니다. QGM은 쿼리 블록, 서브쿼리, 상관 관계, 파생된 테이블, 표현식, 데이터 유형, 데이터 유형 변환, 코드 페이지 변환, 파티션 키를 포함한 쿼리에 대한 모든 의미를 기록합니다. 의미 점검의 예로 YEAR 스칼라 함수에서 인수로 지정된 컬럼의 데이터 유형이 날짜 시간 데이터 유형인지 확인하는 것을 들 수 있습니다.
쿼리 재작성 Rewrite Query	QGM에 저장된 여러 가지 정보를 이용하여 쿼리를 최적화된 형식으로 변환하여 QGM에 저장합니다.
푸쉬다운 분석 Pushdown Anaysis	SQL문을 실행하는데 필요한 조작을 Federated 된 데이터 소스에서 원격으로 평가할 수 있는지 또는 푸시다운 하는지 여부를 분석합니다.
액세스 플랜 최적화 Optimize Access Plan	QGM에 저장된 정보를 기반으로 쿼리의 의미, 인덱스, 기본 테이블, 파생된 테이블, 서브쿼리, 상관 관계 등을 분석하고, 입력된 SQL문을 실행할 수 있는 다양한 액세스 플랜을 생성합니다. 옵티마이저는 시스템 카탈로그의 통계 자료를 이용하여 실행 비용을 측정하고, 생성된 액세스 플랜 중에서 최소 비용의 액세스 플랜을 선택합니다. 액세스 플랜에는 테이블의 액세스 순서, 사용할 인덱스 선택, 조인 방법, 액세스 유형 등이 포함됩니다.
원격 SQL 생성 Generate Remote SQL	선택한 액세스 플랜이 Federated 된 데이터 소스에서 작동할 수 있는 단계 세트로 구성되어야 하는 경우에는 데이터 소스 SQL dialect를 기본으로 하여 효율적인 SQL문을 작성합니다.
실행 코드 생성 Generate Executable	선택된 액세스 플랜과 QGM을 사용하여 쿼리에 대한 실행 액세스 플랜과 섹션을 작성합니다. 정적 SQL문에 대한 액세스 플랜은 데이터베이스의 시스템 카탈로그 테이블에 패키지로 저장되고, 동적 SQL문에 대한 액세스 플랜은 패키지 캐시에 저장됩니다.

3 패키지가 실행되면 데이터베이스 관리 프로그램은 시스템 카탈로그 테이블 또는 패키지 캐시에 저장된 정보를 사용하여 데이터에 액세스할 방법을 결정하고 쿼리를 실행하여 실행 결과를 반환합니다.

4 호스트 변수, 특수 레지스터, 매개 변수 표시 문자를 가지는 정적 SQL문과 동적 SQL문에 대한 액세스 플랜은 SQL 컴파일러가 선택한 기본 예상치 (Default Filtering Factor)를 이용합니다. 실행시 실제로 입력된 값을 이용한 액세스 플랜을 생성하려면, 바인드시 REOPT 옵션을 사용합니다.

TOPIC 22.02 최적화 클래스

Point

옵티마이저가 SQL문을 컴파일할 때 적용하는 최적화의 심화 정도를 결정하는 수치입니다. 숫자가 높을수록 다양한 알고리즘을 시도하여 액세스 플랜을 결정합니다. 기본 클래스는 5입니다.

Tip

Federated된 데이터베이스에 대한 쿼리에서, 최적화 클래스는 리모트 옵티마이저에 적용되지 않습니다.

Tip

클래스 2는 동적 프로그래밍 대신 Greedy 조인 열거를 사용하는 것을 제외하면 클래스 5와 매우 유사합니다.

1. 사용자는 SQL문이 컴파일될 때 적당한 최적화 클래스를 지정하여 옵티마이저가 가장 효율적인 액세스 플랜을 선택하도록 조절할 수 있습니다. 최적화 클래스를 개별적으로 지정하여 쿼리에 대한 런타임 성능을 향상시킬 수 있으나, 높은 최적화 클래스를 지정할수록 더 많은 컴파일 시간과 시스템 자원을 필요로 하게 됩니다.

2. 최적화 클래스는 다음과 같이 7가지로 구분됩니다.

클래스	설명
0	최소의 최적화 알고리즘을 사용하므로, 쿼리 컴파일 오버헤드를 최소화하는 경우에만 지정하도록 합니다. 인덱스화가 잘 되어 있는 테이블에 액세스하는 매우 간단한 동적 SQL문으로 구성된 응용프로그램에 적합합니다. • 불균등 분포 분산 통계를 고려하지 않습니다. • 기본적인 쿼리 재작성 규칙만이 적용됩니다. • Greedy 조인 열거 기능이 사용됩니다. • Nested Loop 조인, Index Scan 액세스 메소드만이 작동 가능합니다. • 리스트 프리페치, 스타 조인 전략은 고려되지 않습니다.
1	추가적으로 Merge Scan 조인, Table Scan이 사용되는 점을 제외하면 클래스 0과 유사합니다. • 불균등 분포 분산 통계를 고려하지 않습니다. • 쿼리 재작성 규칙의 일부 기능만 적용됩니다. • Greedy 조인 열거 기능이 사용됩니다. • Merge Scna 조인, Table Scan이 사용됩니다. • 리스트 프리페치가 사용되지 않습니다.
2	모든 통계 자료와 쿼리 재작성 규칙, 리스트 프리페치, 요약 테이블 등이 적용됩니다. 복합 쿼리에 대한 대체 방법을 거의 고려하지 않으므로, 3 이상의 클래스보다 컴파일 시간이 적게 걸립니다. 특정 쿼리가 거의 반복되지 않는 DSS 또는 OLAP 환경의 복잡한 쿼리에 적합합니다. • 빈도 및 qantile 불균등 분포 분산 통계를 포함한 모든 통계가 사용됩니다. • MQT 라우팅을 포함한 대부분의 쿼리 재작성 규칙이 적용됩니다. • Greedy 조인 열거 기능이 사용됩니다. • 리스트 프리페치, MQT 라우팅을 포함한 액세스 메소드가 고려됩니다. • 스타 조인 전략은 고려되지 않습니다.
3	MVS/ESA용 DB2, OS/390 또는 z/OS의 쿼리 최적화 특성에 가장 유사하며, 중간 수준의 최적화를 위해 적당한 양의 리소스를 사용하므로 대부분의 응용프로그램에 적합합니다. • 불균등 분포 분산 통계는 발생하는 값을 추적하는 데 사용됩니다. • 조인의 서브쿼리 변환 등을 포함한 대부분의 쿼리 재작성 규칙이 적용됩니다. • 동적 프로그래밍 조인 열거 기능이 사용됩니다. • 리스트 프리페치, 인덱스 추가, 스타 조인 등의 액세스 메소드가 고려됩니다.
5	대부분의 다양한 쿼리 변환과 쿼리 최적화 기술을 효과적으로 적용하도록 설계되어 상당한 수준의 최적화를 실행하므로, 트랜잭션과 복합 쿼리를 모두 구성하는 혼합된 환경에 적합합니다. 복잡한 동적 SQL 쿼리에 대해서는 고려되는 액세스 플랜 조합 수를 줄입니다. • 빈도 및 quantile 불균등 분포 분산 통계를 포함한 모든 통계가 사용됩니다. • MQT 라우팅을 포함한 대부분의 쿼리 재작성 규칙이 적용됩니다. • 동적 프로그래밍 조인 열거 기능이 사용됩니다. • 리스트 프리페치, MQT 라우팅을 포함한 액세스 메소드가 고려됩니다.

TOPIC 22.02 최적화 클래스

Point

옵티마이저가 SQL문을 컴파일할 때 적용하는 최적화의 심화 정도를 결정하는 수치입니다. 숫자가 높을수록 다양한 알고리즘을 시도하여 액세스 플랜을 결정합니다. 기본 클래스는 5입니다.

Tip

장시간 실행되는 쿼리를 분석하려면, db2batch로 쿼리를 실행하여 SQL문을 컴파일하는 데 걸린 시간과 실행에 걸린 시간을 확인합니다. 컴파일 시간을 줄이려면 낮은 최적화 클래스를 지정하고, 실행 시간을 줄이려면 높은 최적화 클래스를 지정합니다.

클래스	설명
7	상당한 수준의 최적화를 실행합니다. 복잡한 동적 SQL 쿼리에 대해서도 모든 최적화 기법을 적용한다는 점을 제외하면 클래스 5와 유사합니다.
9	사용 가능한 모든 최적화 기법을 사용하도록 지시하므로 고려되는 가능한 액세스 플랜의 수를 크게 확장할 수 있습니다. 대형 테이블을 통해 매우 복잡하고 장시간에 걸쳐 실행되는 쿼리에 대해 향상된 액세스 플랜을 생성할 수 있는지 확인하는 용도로 사용합니다. • 사용 가능한 모든 통계를 이용합니다. • 모든 쿼리 재작성 규칙을 적용합니다. • 카테시안곱과 모든 가능한 조인 열거 기능을 사용합니다. • 모든 액세스 메소드를 적용합니다.

3 쿼리 최적화 클래스 1, 2, 3, 5, 7은 모두 일반적인 목적으로 사용하는 데 적합합니다. 쿼리 컴파일 시간을 더 줄여야 하거나, SQL문이 아주 단순한 경우에만 클래스 0을 고려합니다.

4 대부분의 SQL문은 기본 쿼리 최적화 클래스인 최적화 클래스 5를 사용하여 적합하게 최적화될 수 있습니다. 주어진 최적화 클래스에서 쿼리 컴파일 시간과 자원 소비는 쿼리의 복잡도, 특히 조인 및 서브쿼리 수에 의해 주로 영향을 받습니다. 그러나, 컴파일 시간과 자원의 사용은 수행된 최적화의 양에 의해서도 영향을 받습니다. 소규모의 데이터베이스, 간단한 동적 쿼리, 컴파일시 사용하는 메모리의 양을 제한, 컴파일 시간의 단축 등이 필요하다면, 최적화 클래스를 조절할 수 있습니다.

Tip

쿼리 생성 프로그램에 의해 생성된 SQL문에 대해서는 높은 쿼리 최적화 클래스가 유리합니다.

5 최적화 클래스를 선택하는 경우에는 일반적으로 다음과 같은 순서를 고려합니다.

- 기본값인 클래스 5로 시작하고, 다른 클래스를 사용하려면 클래스 1, 2, 3부터 시도합니다.
- 같은 컬럼에서 많은 Join 술어를 가진 테이블이 있거나 컴파일 시간이 중요하다면, 클래스 1 또는 2를 사용합니다.
- 하나 또는 적은 테이블만 액세스, 단일 또는 적은 행만 페치, 왼전하고 고유한 인덱스를 사용하는 1 초 이하의 OLTP 쿼리에 대해서는 클래스 0 또는 1을 사용합니다.
- 30초를 초과하여 장시간 실행되는 쿼리에 대해서는 클래스 3, 5, 7 을 사용합니다.
- 클래스 3 이상은 동적 프로그래밍 조인 열거 알고리즘을 사용하여 많은 액세스 플랜을 고려하므로, 테이블 수가 증가하면 클래스 0, 1, 2보다 컴파일 시간이 많이 소요됩니다.

6 결정 지원 쿼리 또는 월말 보고 쿼리 등의 복합 쿼리는 기본값인 클래스 5 이상을 적용합니다.

- 대형 테이블에 대한 액세스
- 많은 수의 술어
- 많은 서브쿼리
- 많은 조인
- UNION 및 INTERSECT와 같은 많은 집합 연산자
- 많은 규정 행
- GROUP BY 및 HAVING 조작
- 중첩 테이블 표현식
- 많은 수의 뷰

TOPIC 22.03 최적화 클래스 지정 방법

Point

정적 SQL문은 PREP 또는 BIND 명령어의 QUERYOPT 옵션으로 지정하며, 동적 SQL문은 특수 레지스터리인 CURRENT QUERY OPTIMIZATION 으로 지정합니다. 데이터베이스 구성 변수인 dft_queryopt로 기본 최적화 클래스를 변경합니다.

1 최적화 클래스는 dft_queryopt 데이터베이스 구성 변수로 조절합니다. get db cfg 명령어로 확인하고, update db cfg 명령어로 변경합니다. 기본값은 5 로 설정되어 있습니다.

```
$ db2 get db cfg for <DB명> | grep DFT_QUERYOPT
$ db2 update db cfg for <DB명> using DFT_QUERYOPT <최적화클래스>
```

2 정적 SQL문에 대한 컴파일은 한 번만 실행되므로, 컴파일 시간과 자원은 한 번만 소요됩니다. 생성된 액세스 플랜은 시스템 카탈로그에 저장되어 여러 번 사용될 수 있습니다. 일반적으로, 정적 SQL은 기본 쿼리 최적화 클래스를 사용합니다.

3 정적 SQL문에 대한 최적화 클래스는 프리컴파일이나 바인드 과정에서 QUERYOPT 라는 옵션을 이용하여 패키지 단위로 변경할 수 있습니다. 옵션을 지정하지 않으면, dft_queryopt 데이터베이스 구성 변수에 지정된 값을 사용합니다.

```
$ db2 prep <ESQL 소스파일명> QUERYOPT <최적화 클래스> bindfile
$ db2 bind <바인드 파일명> QUERYOPT <최적화 클래스>
```

4 정적 SQL문에 대해 적용된 최적화 클래스는 SYSCAT.PACKAGES 카탈로그 테이블의 QUERYOPT 컬럼에서 확인할 수 있습니다.

```
$ db2 "select pkgname, QUERYOPT from syscat.packages"
```

5 동적 SQL문은 실행시에 액세스 플랜이 생성되어 실행되므로, 동적 SQL문에 대한 컴파일 시간과 자원은 실행시마다 소요될 수도 있습니다. 동적 SQL문에 대한 액세스 플랜은 시스템 카탈로그에 저장되지 않고, 패키지 캐시에 저장됩니다. 액세스 플랜이 캐시된 이후에 환경이 변경되지 않았다면, 이후의 PREPARE문에 포함된 동일한 동적 SQL문은 패키지 캐시의 액세스 플랜을 재사용하기 때문에 동일한 동적 SQL문은 재컴파일할 필요가 없습니다. 만약, 환경적인 변경이 발생하면, 동적 SQL문은 재컴파일되어 패키지 캐시에 다시 저장됩니다.

6 패키지 캐시의 크기는 PCKCACHESZ 데이터베이스 구성 변수로 조절합니다. get db cfg 명령어로 확인하고, update db cfg 명령어로 변경합니다. 기본값은 -1로 설정되어 있으므로 maxappls 데이터베이스 구성 변수에 지정된 값의 8배가 할당됩니다.

```
$ db2 get db cfg for <DB명> | grep PCKCACHESZ
$ db2 update db cfg for <DB명> using PCKCACHESZ <최적화클래스>
```

7 동적 SQL문은 최적화 클래스는 SET CURRENT QUERY OPTIMIZATION 이라는 특수 레지스터리를 이용하여 변경합니다. 레지스터리의 값을 지정하지 않으면, dft_queryopt 데이터베이스 구성 변수에 지정된 값이 사용됩니다.

```
$ db2 set CURRENT QUERY OPTIMIZATION = <최적화 클래스>
```

Tip

PREP 명령어에서 최적화 클래스는 BIND 명령어에서 지정한 최적화 클래스의 값으로 대체됩니다.

Tip

REBIND 명령어로 패키지를 리바인드하면 정적 SQL문에 대해 기존의 최적화 클래스가 사용됩니다. 최적화 클래스를 변경하려면 BIND 명령어를 사용합니다.

Tip

FLUSH PACKAGE CACH문은 현재 패키지 캐시에 캐시된 모든 동적 SQL문을 제거합니다. 캐시된 동적 SQL문은 논리적으로 무효화되었으므로, 동일한 동적 SQL문이 요청되면 재컴파일됩니다.

Tip

동적 SQL문에 대해 항상 특정한 최적화 클래스가 적용되도록 하려면, 응용프로그램의 소스에 SET문을 포함시킬 수 있습니다.

TOPIC 22-04 쿼리 재작성

Point

SQL 컴파일러는 최적의 액세스 플랜을 선택하기 위하여 쿼리를 재작성합니다. 쿼리의 재작성의 유형으로는 조작 병합, 조작 이동, 술어 변환 등이 있습니다.

1 주요한 쿼리 재작성의 유형은 다음과 같습니다.

유 형	설 명
조작 병합	가장 적은 수의 조작, 특히 SELECT 조작을 갖도록 쿼리를 구성하기 위해, 쿼리를 재작성하여 쿼리 조작을 병합합니다. 뷰 병합, 서브쿼리에서 조인으로의 변환, 중복 조인 제거, 공유 집계 등을 예로 들 수 있습니다.
조작 이동	조작 및 술어의 개수를 최소로 하는 쿼리를 구성하기 위해 쿼리를 재작성하여 쿼리 조작을 이동시킵니다. DISTINCT 제거, 일반 술어 푸시다운, 상관 해제 등을 예로 들 수 있습니다.
술어 변환	기존의 술어를 특정 쿼리에 대해 보다 최적화된 술어로 변환하도록 쿼리를 재작성합니다. 암시적 술어의 추가, OR에서 IN으로의 변환을 예로 들 수 있습니다.

2 조작 병합의 예는 다음과 같습니다.

방 법	설 명
뷰 병합	뷰를 사용하는 SELECT문은 테이블의 조인 순서를 제한하며, 테이블의 중복 조인을 유발할 수도 있습니다. 뷰가 병합되면, 이러한 제한 사항을 없앨 수 있습니다.
서브쿼리에서 조인으로의 변환	SELECT문에 서브쿼리가 들어 있는 경우, 서브쿼리를 제거하여 조인으로 변환합니다.
중복 조인 제거	중복 조인을 제거하여 SELECT문을 단순화할 수 있습니다.
공유 집계	쿼리에서 사용된 특정 집계 함수를 쿼리에 포함된 기존의 다른 집계 함수들의 실행 결과를 이용하여 재작성함으로써 수행해야 할 계산의 개수를 줄일 수 있습니다.

3 조작 이동의 예는 다음과 같습니다.

방 법	설 명
DISTINCT 제거	DISTINCT 조작이 수행되는 지점을 이동시키거나, 조작을 완전히 제거하여 실행 비용을 줄일 수 있습니다.
일반 술어 푸시다운	조건에 맞는 자료의 범위 또는 개수를 더 많이 줄일 수 있는 술어가 먼저 쿼리에 적용될 수 있도록 하기 위하여 적용되는 술어의 순서를 변경할 수 있습니다.
상관 해제	파티션된 데이터베이스 환경에서 데이터베이스 파티션 사이에서의 결과 세트 이동은 비효율적입니다. 다른 데이터베이스 파티션으로 브로드캐스트해야 하는 크기를 또는 개수를 줄이는 것이 쿼리 재작성의 목적입니다.

4 술어 이동의 예는 다음과 같습니다.

방 법	설 명
암시적 술어의 추가	추가적인 테이블 조인을 고려할 수 있도록 숨어있는 술어가 해당 쿼리에 추가될 수 있습니다.
OR에서 IN으로의 변환	OR 술어를 IN 술어로 변환할 수 있습니다.

Tip

필요에 따라 IN 술어를 OR 술어로 변환할 수도 있습니다.

TOPIC 22 05 뷰 병합

Point

쿼리의 재작성 유형 중에 조작 병합의 예로 뷰 병합을 들 수 있습니다. 뷰를 사용하는 SELECT문의 테이블의 조인 순서에 대한 제한과 테이블의 중복 조인을 제거할 수 있습니다.

1 뷰를 병합하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 KES.EMPL입니다.

```
$ db2 "SELECT id, name, salary FROM kes.empl"
ID          NAME                SALARY
----------------------------------------------------------
     1      KES                 100
     2      KHY                 250
     3      JHS                 300
     4      JJY                 280
```

2 KES.EMPL 테이블에서 NAME 컬럼의 값이 'K' 로 시작되는 정보만 표시하는 뷰를 생성합니다.

```
$ cat view.sql
CREATE VIEW kes.vname AS
    SELECT * FROM kes.empl WHERE name LIKE 'K%';
$ db2 –tf view.sql
```

3 KES.EMPL 테이블에서 SALARY 컬럼의 값이 200 이상인 정보만 표시하는 뷰를 생성합니다.

```
$ cat view.sql
CREATE VIEW kes.vsalary AS
    SELECT * FROM kes.empl WHERE salary >= 200;
$ db2 –tf view.sql
```

4 KES.EMPL 테이블에서 NAME 컬럼의 값이 'K' 로 시작되고, SALARY 컬럼의 값이 200 이상인 사원의 정보만 표시하기 위하여 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   v1.id, v1.name, v2.salary
FROM     kes.vname v1, kes.vsalary v2
WHERE    v1.id = v2.id;
$ db2 –tf query.sql
```

Tip

사용자가 작성한 SELECT문을 포함하는 두 개의 뷰로부터 SELECT문을 병합하면, 액세스 플랜 선택시에 보다 많은 선택 사항을 고려할 수 있습니다. 병합된 두 개의 뷰가 동일한 기본 테이블을 사용하는 경우에는 추가 재작성 작업을 수행할 수도 있습니다.

5 쿼리 재작성 과정에서 두 개의 뷰는 병합되어 다음과 같이 재작성될 수 있습니다.

```
SELECT   v1.id, v1.name, v2.salary
FROM     kes.empl v1, kes.empl v2
WHERE    v1.id = v2.id
AND      v1.name LIKE 'K%'
AND      v2.salary >= 200
```

TOPIC 22-06 서브쿼리에서 조인으로의 변환

Point

쿼리의 재작성 유형 중에 조작 병합의 예로 서브쿼리에서 조인으로의 변환을 들 수 있습니다. 서브쿼리가 제거되어 조인으로 변환될 수도 있습니다.

Tip

SAMPLE 데이터베이스를 생성하려면, 인스턴스 사용자로 로그인한 후에 db2sampl 명령어를 실행합니다.

1 서브쿼리에서 조인으로 변환하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE와 DEPARTMENT 입니다.

```
$ db2 "SELECT empno, lastname, workdept FROM employee"
EMPNO  LASTNAME       WORKDEPT
-------------------------------------------------------
000010 HAAS           A00
000020 THOMPSON       B01
000030 KWAN           C01
000050 GEYER          E01
000060 STERN          D11

$ db2 "SELECT deptno, deptname FROM department"
DEPTNO DEPTNAME
-------------------------------------------------------
A00    SPIFFY COMPUTER SERVICE DIV.
B01    PLANNING
C01    INFORMATION CENTER
D01    DEVELOPMENT CENTER
D11    MANUFACTURING SYSTEMS
```

2 서브쿼리를 포함한 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   empno,  lastname
FROM     employee
WHERE    workdept IN
           (SELECT   deptno
            FROM     department
            WHERE    deptname = 'PLANNING');
$ db2 –tf query.sql
```

Tip

일반적으로 조인은 서브쿼리보다 더 효율적으로 실행됩니다.

3 서브쿼리는 다음과 같이 조인으로 변환될 수 있습니다.

```
SELECT   DISTINCT empno, lastname
FROM     employee emp,
         department dept
WHERE    emp.workdept = dept.deptno
AND      dept.deptname = 'PLANNING'
```

TOPIC 2207 중복 조인 제거

Point

쿼리의 재작성 유형 중에 조작 병합의 예로 중복 조인 제거를 들 수 있습니다. 중복 조인을 제거하여 SELECT문을 단순화하면 실행 비용을 줄일 수 있습니다.

1. 서브쿼리에서 조인으로 변환하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE와 DEPARTMENT 입니다. 다음의 쿼리는 EMPLOYEE 테이블에 대한 불필요한 중복 조인을 포함하고 있습니다.

```
$ cat query.sql
SELECT   e1.empno, e1.firstnme, e1.lastname, e1.edlevel
FROM     employee e1, employee e2
WHERE    e1.empno = e2.empno
AND      e1.edlevel > 17
AND      e2.salary  > 35000 ;
$ db2 –tf query.sql
```

2. EMPLOYEE 테이블에 대한 중복 조인을 제거하여 다음과 같이 단순화될 수 있습니다.

```
SELECT   empno, firstnme, lastname, edlevel
FROM     employee
WHERE    edlevel > 17
AND      salary  > 35000
```

3. 또 다른 예를 위해 다음과 같이 뷰를 작성합니다. EMPLOYEE와 DEPARTMENT 테이블에는 WORKDEPT와 DEPTNO 컬럼을 통해 참조 제한 조건이 설정되어 있다고 가정합니다.

```
$ cat view.sql
CREATE VIEW vpeople AS
      SELECT   lastname, salary, deptno, deptname, mgrno
      FROM     employee e, department d
      WHERE    e.workdept = d.deptno ;
$ db2 –tf view.sql
```

4. 위에서 정의한 뷰를 이용하여 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT  lastname, salary FROM vpeople;
$ db2 –tf query.sql
```

5. VPEOPLE 뷰에서 사용되었던 EMPLOYEE와 DEPARTMENT 테이블에 대한 조인이 제거되고 아래와 같이 재작성될 수 있습니다.

```
SELECT   lastname, salary
FROM     employee
WHERE    workdept IS NOT NULL
```

Tip

쿼리를 실행하는 사용자에게 테이블이 아닌 뷰에 대한 액세스 권한만이 있다면, 이러한 상황에서는 쿼리를 재작성할 수 있다는 사실을 알더라도 테이블에 대한 액세스 권한이 없기 때문에 재작성하지 못할 수도 있습니다.

TOPIC 22-08 공유 총계

Point

쿼리의 재작성 유형 중에 조작 병합의 예로 공유 총계를 들 수 있습니다. 쿼리에서 사용된 특정 집계 함수를 쿼리에 포함된 기존의 다른 집계 함수들의 실행 결과를 이용하여 재작성함으로써 수행해야 할 계산의 개수를 줄여서 쿼리의 실행 비용을 줄일 수 있습니다.

Tip

AVG 함수를 대체하기 위해 사용된 osum / ocount 표현식에서 나눗셈 연산은 한 번만 실행됩니다.

1. 쿼리에 사용되는 함수의 종류가 많으면 쿼리의 실행 속도는 당연히 떨어집니다. 쿼리에 이미 사용된 다른 함수를 이용하여 필요한 함수의 기능을 대체할 수 있는 있다면, 쿼리 내에서 수행될 계산의 개수를 줄이면서 더 개선된 액세스 플랜을 생성할 수 있습니다.

2. 공유 총계를 이용하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE입니다. SUM, AVG, COUNT 함수를 사용하는 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   SUM(salary+bonus+comm) AS osum,
         AVG(salary+bonus+comm) AS oavg,
         COUNT(*) AS ocount
FROM     employee;
$ db2 –tf query.sql
```

3. SUM과 AVG 함수의 대상으로 사용되는 계산식인 SALARY+BONUS+COMM 이 동일하므로 AVG 함수는 SUM 함수와 COUNT 함수를 이용하여 SUM/COUNT 형식으로 재작성할 수 있습니다. AVG 함수의 대상으로 사용되었던 계산식이 제거되어 실행해야할 계산식의 개수를 줄였으므로 쿼리의 실행 비용을 줄일 수 있습니다.

```
SELECT   osum,
         osum / ocount
         ocount
FROM   (SELECT   sum(salary+bonus+comm) as osum,
                 count(*) as ocount
         FROM employee
        ) AS shared_agg;
```

TOPIC 22.09 DISTINCT 제거

Point

쿼리의 재작성 유형 중에 조작 이동의 예로 DISTINCT 제거를 들 수 있습니다. DISTINCT 조작이 수행되는 지점을 이동시키거나, 조작을 완전히 제거하여 실행 비용을 줄일 수 있습니다.

1. DISTINCT를 제거하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE입니다.

2. EMPNO 컬럼은 EMPLOYEE 테이블의 기본키로 정의되어 있다고 가정합니다. 기본키를 생성하면, 자동적으로 고유한 인덱스가 생성됩니다.

```
$ db2 describe indexes for table employoee
인덱스          인덱스                          고유한  컬럼  컬럼
스키마          이름                            규칙    수    이름
------------------------------------------------------------------------------------------
SYSIBM          SQL060511155131630              P       1     +EMPNO
```

3. 다음과 같이 불필요한 DISTINCT 키워드가 포함된 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   DISTINCT empno, firstnme, lastname
FROM     employee;
$ db2 -tf query.sql
```

4. EMPNO 컬럼이 EMPLOYEE 테이블의 기본 키로 정의되었고, 쿼리의 SELECT 절에 기본키인 EMPNO가 선택되기 때문에 리턴되는 각 행이 고유할 것이라는 것을 이미 알고 있습니다. 이러한 경우에는, DISTINCT 키워드를 사용한 경우와 사용하지 않은 경우의 실행 결과가 동일하다는 것이 보장되므로 위의 쿼리는 DISTINCT절을 제거하여 다음과 같이 재작성됩니다.

```
SELECT   empno, firstnme, lastname
FROM     employee
```

Tip

UNIQUE 옵션을 가진 고유 인덱스를 생성하고, SELECT 절에 고유 인덱스의 키 컬럼이 모두 포함되었다면, 기본키와 마찬가지로 DISTINCT 키워드는 제거될 수 있습니다.

TOPIC 22.10 일반 술어 푸시다운

Point

쿼리의 재작성 유형 중에 조작 이동의 예로 일반 술어 푸시다운을 들 수 있습니다. 조건에 맞는 자료의 범위 또는 개수를 더 많이 줄일 수 있는 술어가 먼저 쿼리에 적용될 수 있도록 하기 위하여 적용되는 술어의 순서를 변경하여 비용을 줄일 수 있습니다.

1 일반 술어를 푸시다운하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE 입니다. 푸쉬다운이란 쿼리에서 표현된 여러 개의 술어 중에서 순서적으로 나중에 적용될 술어를 먼저 적용될 수 있도록 해당 술어의 적용 순서 레벨을 낮추는 방법입니다. 즉, WHERE 절의 AND 또는 OR로 연결된 여러 개의 술어 중에서 데이터 필터링의 정도가 높은 술어를 더 앞쪽으로 배치하여 술어의 적용 순서를 변경합니다.

2 EMPLOYEE 테이블에서 WORKDEPT 컬럼의 값이 'D11' 인 부서에 속한 모든 사원의 목록을 제공하는 다음과 같은 뷰를 생성합니다.

```
$ cat view.sql
CREATE VIEW d11_empl (empno, firstnme, lastname, phoneno) AS
SELECT   empno, firstnme, lastname, phoneno
FROM     employee
WHERE    workdept = 'D11';
$ db2 –tf view.sql
```

3 뷰를 이용하여 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   firstnme, phoneno
FROM     d11_empl
WHERE    lastname = 'BROWN';
$ db2 –tf query.sql
```

4 D11_EMPL 뷰에 대한 쿼리는 단순히 EMPLOYEE 테이블에 대한 쿼리로 재작성될 수도 있습니다. WORKDEPT = 'D11' 이라는 술어를 먼저 적용하는 것이 쿼리를 더 빠르게 실행할 수 있다고 판단되는 경우에는 일반 술어의 푸쉬다운이 발생하지 않습니다.

```
SELECT   firstnme, phoneno
FROM     employee
WHERE    workdept = 'D11'
AND      lastname = 'BROWN'
```

Tip

UNION, GROUP BY, 중첩 테이블 표현식, 공통 테이블 표현식이 사용된 쿼리에서도 술어가 푸시다운될 수 있습니다.

5 푸시다운이 유리하다고 판단되면, 뷰 D11_EMPLOYEE의 조건문에 원래의 쿼리에서 지정된 LASTNAME = 'BROWN' 이라는 술어가 먼저 적용될 수 있도록 푸시다운하여 다음과 같이 재작성될 수 있습니다.

```
SELECT   firstnme, phoneno
FROM     employee
WHERE    lastname = 'BROWN'
AND      workdept = 'D11'
```

TOPIC 22.11 상관 해제

Point

쿼리의 재작성 유형 중에 조작 이동의 예로 상관 해제를 들 수 있습니다. 파티션된 데이터베이스 환경에서 다른 데이터베이스 파티션으로 브로드캐스트해야 하는 크기를 또는 개수를 줄이면 쿼리의 실행 비용을 줄일 수 있습니다.

1. 파티션된 데이터베이스 환경에서 PROJECT 테이블의 PROJNAME 컬럼과 EMPLOYEE 테이블의 SALARY, BONUS, COMM 컬럼을 이용하여 '%PROGRAMMING%' 프로젝트에서 작업 중이면서 총급여가 평균보다 적은 사원을 조회하는 쿼리를 실행합니다.

```
SELECT   p.projno, e.empno, e.lastname,
         e.salary+e.bonus+e.comm AS pay
FROM     employee e, project p
WHERE    p.respemp = e.empno
AND      p.projname like '%PROGRAMMING%'
AND      e.salary+e.bonus+e.comm <
            (SELECT   avg(e1.salary+e1.bonus+e1.comm)
             FROM     employee e1, project p1
             WHERE    p1.projname like '%PROGRAMMING%'
             AND      p1.projno = p.projno
             AND      e1.empno = p1.respemp)
```

2. 이 쿼리는 PROJECT 테이블에 대해 상관 관계가 있고, PROJECT와 EMPLOYEE 테이블의 데이터는 PROJNO 컬럼을 기준으로 동일한 파티션에 저장되어 있지 않으므로, PROJECT 테이블의 데이터를 다른 데이터베이스 파티션으로 브로드캐스트하게 될 수도 있습니다. 또한, 서브쿼리가 여러 번 평가되어야 하므로 실행 비용이 높습니다. EMPLOYEE와 PROJECT 테이블을 조인하여 PROJNO 컬럼별로 총급여를 계산하는 AVG_PROJ 라는 공통 테이블을 생성하여 AVG_COMP 컬럼에 PROJNO 컬럼의 값별로 평균 총급여를 저장하도록 합니다.

```
WITH     avg_proj(projno, avg_pay) AS
        (SELECT      p2.projno, avg(e1.salary+e1.bonus+e1.comm)
         FROM        employee e1, project p2
         WHERE       e1.empno = p2.respemp
         AND         p2.projname LIKE '%PROGRAMMING%'
         GROUP BY    p2.projno
```

3. 공통 테이블을 이용하여 재작성된 쿼리에서 AVG_PROJ 공통 테이블로 PROJNO 컬럼별 평균 총급여인 AVG_COMP를 먼저 계산하고, 그 결과를 EMPLOYEE 테이블이 포함된 모든 데이터베이스 파티션으로 브로드캐스트할 수 있습니다.

```
SELECT   p.projno, e.empno, e.lastname,
         e.salary+e.bonus+e.comm as compensation
FROM     project p, employee e, avg_proj a
WHERE    p.empno = e.empno
AND      p.projname LIKE '%programming%'
AND      p.projno = a.projno
AND      e.salary+e.bonus+e.comm < a.avg_comp
```

TOPIC 22.12 암시적 술어

Point

쿼리의 재작성 유형 중에 술어 이동의 예로 암시적 술어를 들 수 있습니다. 추가적인 테이블 조인을 고려할 수 있도록 숨어있는 술어를 해당 쿼리에 추가하여 쿼리의 실행 비용을 줄일 수 있습니다.

1. 암시적 술어를 추가하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 DEPARTMENT, EMPLOYEE, PROJECT 입니다. 다음의 쿼리는 DEPARTMENT 테이블의 ADMRDEPT 컬럼을 이용하여 'E01' 부서에게 보고하는 부서의 관리자들에 대한 정보와 해당 관리자가 담당하는 프로젝트의 목록을 표시합니다.

```
$ cat query.sql
SELECT   d.deptname d.mgrno, e.lastname, p.projname
FROM     department d,
         employee e,
         project p
WHERE    d.admrdept = 'E01'
AND      d.mgrno = e.empno
AND      e.empno = p.respemp;
$ db2 –tf query.sql
```

2. WHERE 절의 D.MGRNO = E.EMPNO 술어와 E.EMPNO = P.RESPEMP 술어로부터 DEPARTMENT 테이블과 PROJECT 테이블에 대해 MGRNO 컬럼과 RESPEMP 컬럼을 이용하여 DEPT.MGRNO = PROJ.RESPEMP 라는 암시적인 술어를 유추할 수 있습니다. 다음과 같은 암시적 술어를 추가하면, DEPARTMENT 테이블과 PROJECT 테이블에 대해 추가적으로 조인을 고려할 수 있습니다.

```
DEPT.MGRNO = PROJ.RESPEMP
```

3. 또 다른 예를 위해 EMPLOYEE 테이블의 WORKDEPT 컬럼과 DEPARTMENT 테이블의 DEPTNO 컬럼을 이용하여 부서 번호가 'E00' 보다 큰 부서와 이 부서에서 일하는 사원의 이름을 표시하는 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   empno, lastname, firstname, deptno, deptname
FROM     employee emp,
         department dept
WHERE    emp.workdept = dept.deptno
AND      dept.deptno > 'E00';
$ db2 –tf query.sql
```

4. 원래의 쿼리에 표현된 EMP.WORKDEPT = DEPT.DEPTNO 술어에 의하면 EMPLOYEE 테이블의 WORKDEPT 컬럼과 DEPARTMENT 테이블의 DEPTNO 컬럼은 동일한 것을 보장할 수 있으므로, DEPT.DEPTNO > 'E00' 술어로부터 유추된 EMP.WORKDEPT > 'E00' 이라는 암시적 술어를 추가하면, EMPLOYEE 테이블에서 조인되는 행의 수를 줄일 수 있으므로 쿼리의 WHERE 절에는 다음과 같은 술어가 추가됩니다.

```
EMP.WORKDEPT > 'E00'
```

TOPIC 22.13 OR에서 IN으로의 변환

Point

쿼리의 재작성 유형 중에 술어 이동의 예로 OR에서 IN으로의 변환을 들 수 있습니다. OR 절을 IN 술어로 변환하거나, IN 술어를 OR 절로 변환하여 쿼리의 실행 비용을 줄일 수 있습니다.

1. OR 술어를 IN 술어로 변환하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE 입니다. 동일한 컬럼으로 OR 절을 이용하여 표현한 조건식은 IN 술어로 변환되어 쿼리의 실행 비용을 줄일 수 있습니다.

2. OR 술어를 IN 술어로 변환하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE 입니다. 다음의 쿼리는 EMPLOYEE 테이블의 WORKDEPT 라는 동일한 컬럼으로 OR절을 이용하여 'D11', 'D21', 'E21' 부서에 속한 사원들의 정보를 표시합니다.

```
$ cat query.sql
SELECT *
FROM    employee
WHERE   workdept = 'D11'
OR      workdept = 'D21'
OR      workdept = 'E21';
$ db2 –tf query.sql
```

Tip

조건식에 사용된 컬럼에 대한 인덱스가 존재한다면, 상황에 따라서 인덱스 ORING을 수행할 수 있도록 하기 위하여 IN 술어가 OR절로 재작성될 수도 있습니다.

3. WORKDEPT 컬럼에 대한 인덱스가 없는 경우에 OR절을 다음과 같이 IN 술어로 재작성되어 쿼리를 보다 효율적으로 처리할 수 있습니다.

```
SELECT *
FROM    employee
WHERE   workdept IN ('D11', 'D21', 'E21')
```

TOPIC 22.14 패키지

Point

SQL 컴파일러가 정적 SQL문을 컴파일하여 생성한 액세스 플랜은 SYSIBM.SYSPACKAGES를 비롯한 여러 개의 데이터베이스 시스템 카탈로그 테이블에 패키지로 저장되어 관리되고, 패키지 캐시를 통해서 실행됩니다.

Tip

패키지명의 최대 길이는 8자입니다. BIND 명령어의 PACKAGE 옵션으로 변경할 수 있습니다. 기본값은 소스 파일명의 최초 8글자입니다.

Tip

바인드 파일에 포함된 SQL문은 db2bfd 명령어로 확인할 수 있습니다.

Tip

액세스 플랜은 SYSIBM.SYSPLAN 테이블에 바이너리 형식으로 저장되므로 익스플레인 유틸리티를 이용하여 확인합니다.

1 데이터베이스에 저장된 패키지 정보는 list packages 명령어를 이용하여 확인합니다.

```
>>-LIST--+-PACKAGES-+--+-----------------------------+---+------------+--><
         '-TABLES---'  |      .-USER----------------. |   '-SHOW DETAIL-'
                       '-FOR--+-ALL-----------------+-'
                              +-SCHEMA--schema-name-+
                              '-SYSTEM--------------'
```

Figure 2214A ··· LIST PACKAGES 명령어

2 list packages 명령어의 옵션은 다음과 같습니다.

옵션	설명
FOR USER	스키마명이 현재 세션의 사용자 ID와 동일한 패키지에 대한 정보를 표시합니다. 기본값이므로 생략할 수 있습니다.
FOR ALL	모든 패키지에 대한 정보를 표시합니다.
FOR SCHEMA <스키마명>	특정 스키마명을 가지는 패키지에 대한 정보를 표시합니다.
FOR SYSTEM	시스템 패키지에 대한 정보를 표시합니다.

3 list packages 명령어의 '유효' 컬럼에 표시되는 패키지의 상태는 다음 3가지로 구분됩니다.

상태	설명
Y	정상 상태의 패키지로 원하는 시점에 실행할 수 있습니다.
N	패키지가 종속되어 있는 테이블 등의 데이터베이스 오브젝트가 삭제된 패키지를 유효하지 않은(Invalid Package) 패키지라고 합니다. 이러한 패키지는 최초의 실행시에 자동적으로 리바인드됩니다.
X	패키지가 종속되어 있는 사용자 정의 함수나 저장 프로시저가 삭제되었기 때문에 사용할 수 없는 패키지를 작동 불능(Inoperative Package) 패키지라고 합니다. 이러한 패키지는 명시적으로 리바인드되어야 실행할 수 있습니다.

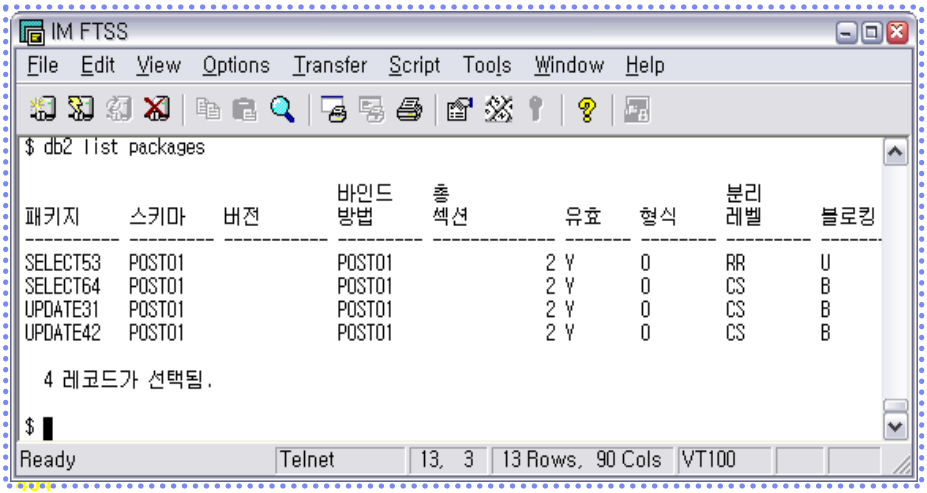


Figure 2214B ··· 패키지 정보

TOPIC 22.15 익스플레인 도구

Point

SQL 컴파일러가 정적 SQL문과 동적 SQL문에 대해 생성한 액세스 플랜을 확인하는 익스플레인 도구에는 GUI기반의 Visual Explain와 텍스트 기반의 db2exfmt, db2expln 등의 유틸리티가 있습니다.

1 SQL 컴파일러는 정적 또는 동적 SQL문의 액세스 플랜 및 환경에 대한 정보를 수집할 수 있습니다. 수집된 익스플레인 정보는 쿼리 처리를 위한 조작 시퀀스, 비용 정보, 각 술어에 대한 술어 및 선택성 추정치, SQL문에서 참조된 모든 오브젝트에 대한 통계 정보 등을 포함하므로, 개별 SQL 명령문의 실행 방법을 파악과 성능 향상을 위한 SQL문의 변경 또는 실행 환경 조절에 도움을 줄 수 있습니다.

Tip

이전 버전에서 사용하던 dynexpln 명령어 대신에 db2expln 명령어를 사용하도록 합니다.

Tip

db2expln 명령어는 매개 변수 표시 문자인 ? (물음표)를 포함한 동적 SQL문을 지원합니다.

2 지원되는 익스플레인 도구와 특성은 다음과 같습니다.

특 성	Explain 테이블	Visual Explain	db2exfmt	db2expln
GUI 출력		O		
텍스트 출력			O	O
'Quick and Dirty' 정적 SQL문 분석				O
지원되는 정적 SQL	O	O	O	O
지원되는 동적 SQL	O	O	O	O
지원되는 CLI 응용프로그램	O	O	O	
DRDA Application Requester 지원	O			
상세 옵티마이저 정보	O	O	O	
다중 명령문 분석	O		O	O
응용프로그램 내에서 액세스 가능한 정보	O			

Tip

Visual Explain 유틸리티를 최초로 실행할 때, 익스플레인 테이블이 자동적으로 생성됩니다.

3 익스플레인 테이블은 익스플레인 과정에서 생성되는 각종 정보를 보관하기 위한 테이블들입니다. 생성 스크립트는 인스턴스 사용자의 홈디렉토리에 있는 sqllib/misc/EXPLAIN.DDL 입니다. Visual Explain 또는 db2exfmt 유틸리티를 사용하려면, 익스플레인 테이블을 미리 생성하여야 합니다.

```
$ db2 connect to <DB명>
$ db2 -tvf $HOME/sqllib/misc/EXPLAIN.DDL
$ db2 list tables for all | grep EXPLAIN
```

Tip

실행하는 사용자의 ID를 스키마명으로 하는 익스플레인 테이블이 있어야 합니다.

4 제어 센터에서 실행되는 Visual Explain은 GUI 방식으로 익스플레인 정보를 제공합니다.

5 db2exfmt 명령어는 익스플레인 테이블의 내용을 정해진 형식의 텍스트로 생성하는 도구입니다. 액세스 플랜 정보와 옵티마이저에 대한 정보를 확인하는데 사용됩니다.

6 db2expln 명령어는 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용됩니다. 하나 이상의 패키지에 사용 가능한 액세스 플랜 정보를 표시하며, 옵티마이저에 대한 정보는 표시하지 않습니다.

TOPIC 22.16 Visual Explain

Point

제어 센터를 이용하면 GUI 형식의 익스플레인 도구인 Visual Explain 을 이용하여 SQL문의 액세스 플랜에 대한 익스플레인 정보를 확인할 수 있습니다.

Tip

기본적으로 원격 데이터베이스인 경우에는 데이터베이스 접속시에 사용자명과 암호를 제공해야 합니다.

1 제어 센터에서 원하는 데이터베이스를 선택한 후 SQL Explain 메뉴를 클릭합니다.

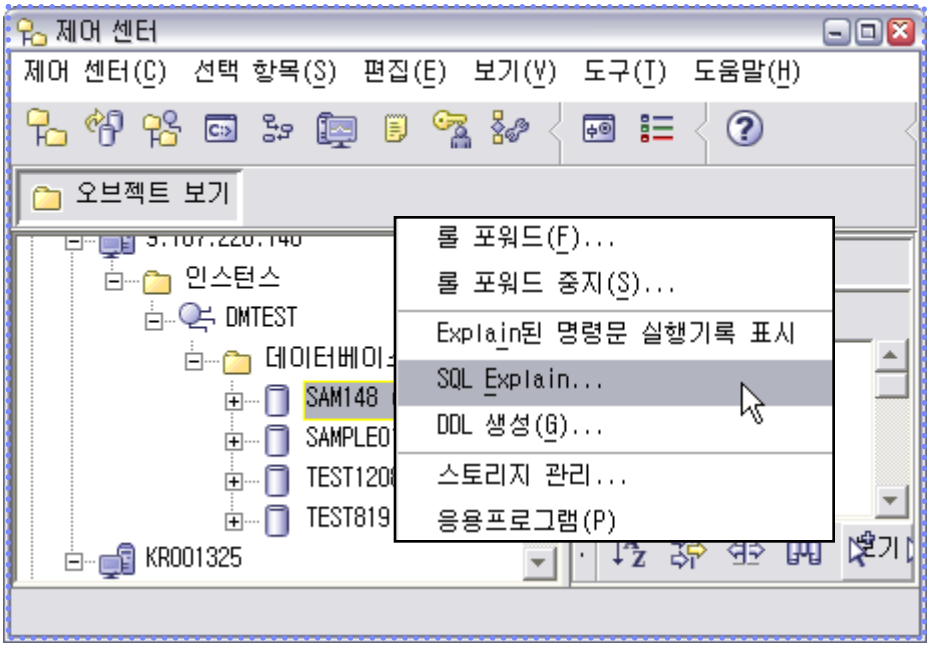


Figure 2216A ••• Visual Explain 메뉴 선택

2 동적 SQL문을 직접 입력하거나 가져오기를 눌러 동적 SQL문이 포함된 파일을 읽어옵니다. 확인 버튼을 클릭하면 익스플레인 정보가 생성됩니다. 접속한 사용자가 Visual Explain 을 최초로 실행할 때는 익스플레인 테이블이 자동적으로 생성됩니다.

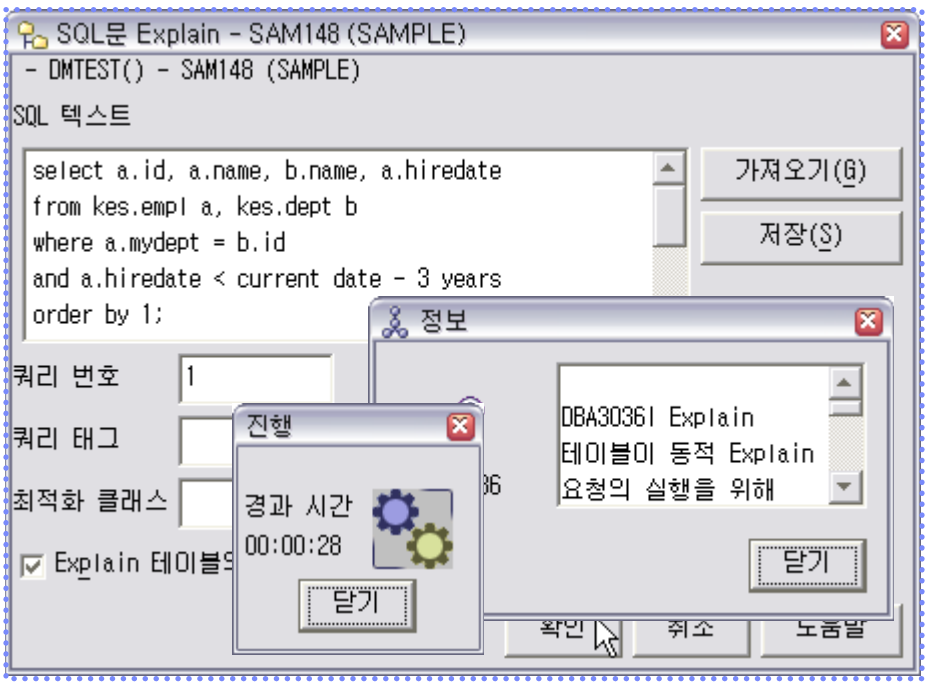


Figure 2216B ••• 동적 SQL문 입력

TOPIC 22.16 Visual Explain

Point

제어 센터를 이용하면 GUI 형식의 익스플레인 도구인 Visual Explain 을 이용하여 SQL문의 액세스 플랜에 대한 익스플레인 정보를 확인할 수 있습니다.

Tip

괄호 안에 표시된 숫자는 각 조작별로 부여되는 일련 번호로, 큰 번호부터 화살표를 따라 역순으로 확인합니다.

Tip

각 다이어그램에는 조작명, 조작 일련 번호, 실행 비용 등이 표시됩니다.

3 생성된 액세스 플랜을 확인합니다.

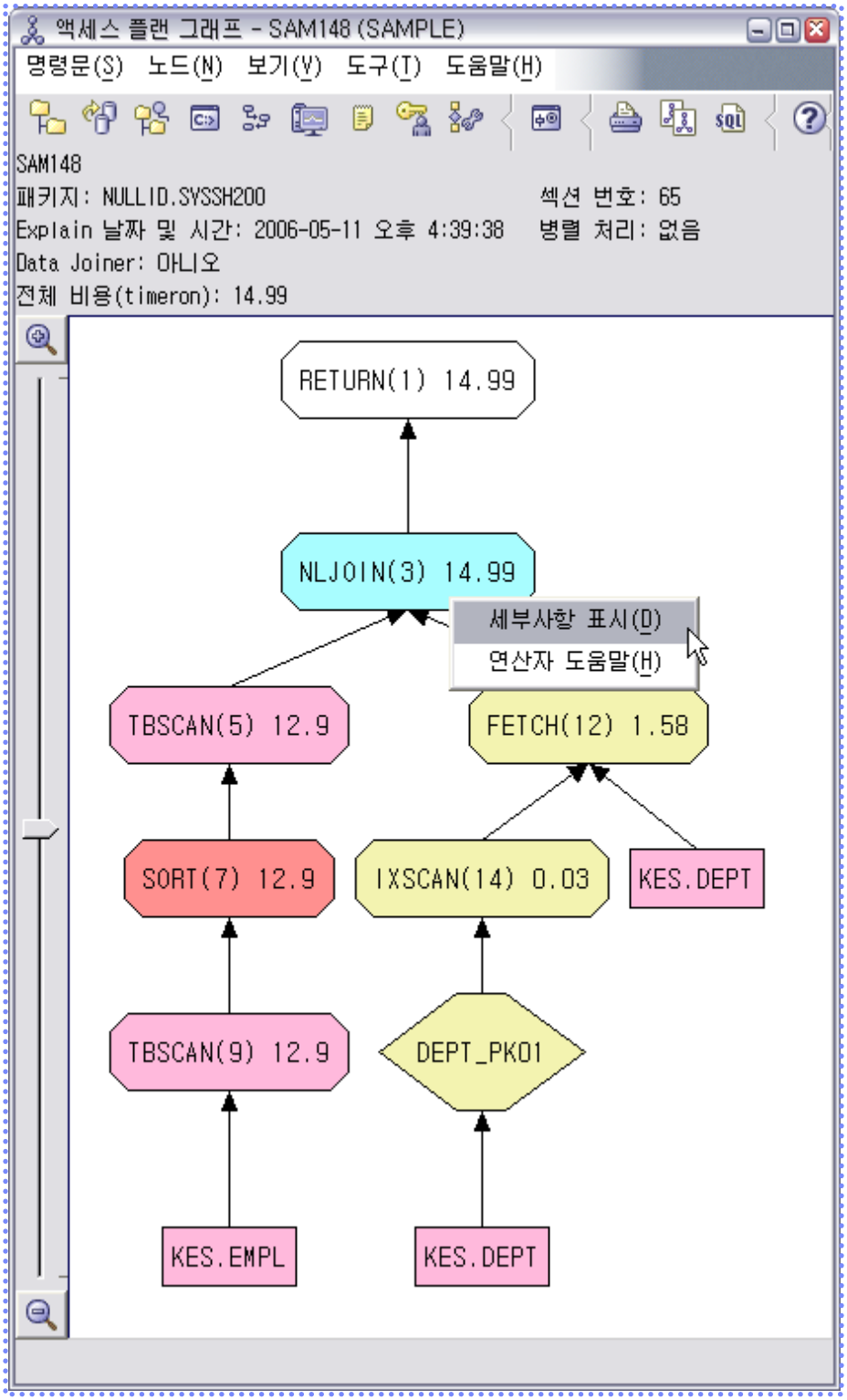


Figure 2216C 액세스 플랜 그래프

TOPIC 22.16 Visual Explain

Point

제어 센터를 이용하면 GUI 형식의 익스플레인 도구인 Visual Explain 을 이용하여 SQL문의 액세스 플랜에 대한 익스플레인 정보를 확인할 수 있습니다.

Tip

비용은 SQL문에 대한 액세스 플랜을 실행하는 데 필요한 자원의 추정 사용량으로 단위는 timeron 입니다.

Tip

timeron은 실제 경과 시간이 아니라, CPU 비용과 I/O 비용을 합하여 추산된 상대적인 비용입니다.

Tip

CPU 비용은 명령어의 수를 표시합니다. 현 시스템이 한 개의 명령어를 실행하는 CPU의 속도와 관련됩니다.

Tip

I/O 비용은 검색 및 전송된 페이지 수를 표시합니다. 현 시스템이 한 페이지를 처리하는 속도와 관련됩니다.

Tip

일반적으로 동일한 쿼리에 대한 여러 액세스 플랜을 비교할 때, timeron 단위의 비용이 적은 액세스 플랜이 빠른 것으로 간주하면 됩니다.

4 각 다이어그램에서 마우스의 오른쪽 버튼으로 조작별 세부 사항 정보를 확인할 수 있습니다.

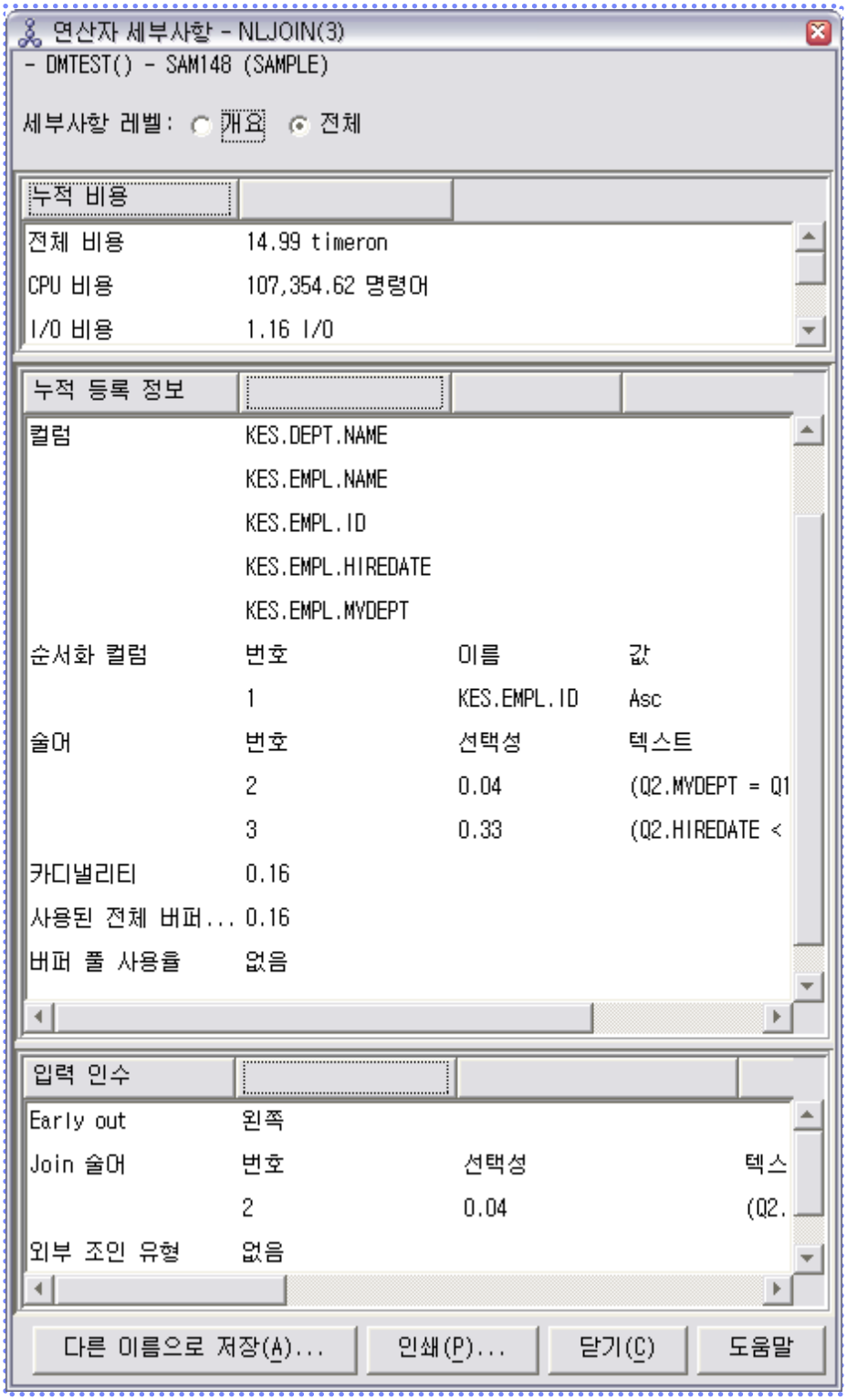


Figure 2216D ••• 연산자 세부 사항 표시

TOPIC 22.16 Visual Explain

Point

제어 센터를 이용하면 GUI 형식의 익스플레인 도구인 Visual Explain 을 이용하여 SQL문의 액세스 플랜에 대한 익스플레인 정보를 확인할 수 있습니다.

Tip

연산자 유형 오른쪽 대괄호 안의 수는 각 노드에 대해 고유한 ID입니다. ID 옆의 숫자는 누적 비용을 표시합니다.

Tip

쿼리가 병렬로 처리되면, 한 개의 쿼리를 처리하는데 여러 개의 데이터베이스 에이전트가 사용됩니다.

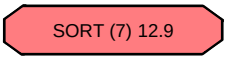
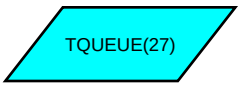
Tip

테이블 큐가 단일 노드 내의 데이터베이스 에이전트 사이에서 데이터를 전달하는데 사용되면 로컬 테이블 큐라고 합니다. 로컬 테이블 큐는 파티션 내 병렬 처리에 사용됩니다.

Tip

테이블 큐가 서로 다른 노드의 데이터베이스 에이전트 사이에서 데이터를 전달하는데 사용되면 로컬 테이블 큐라고 하지 않으며, 파티션간의 병렬 처리에 사용됩니다.

5 액세스 플랜 그래프는 액세스 플랜을 트리 구조로 표시합니다. 트리의 각 노드에 표시되는 다이어그램의 의미는 다음과 같습니다.

다이어그램	설명
	직사각형. 연산자의 대상이 되는 테이블을 표시합니다.
	다이아몬드. 연산자의 대상이 되는 인덱스를 표시합니다.
	팔각형. 데이터에 대해 수행되어야 하는 조치 또는 인덱스와 테이블로부터의 출력을 저장하는 연산자를 표시합니다.
	평행사변형. 쿼리를 처리하는 에이전트가 여러 개 있는 경우, 한 개의 데이터베이스 에이전트에서 다른 데이터베이스 에이전트로 테이블의 데이터를 전달하는데 사용되는 테이블 큐를 표시합니다.
	육각형. 테이블 함수를 표시합니다.

5 주요 연산자에 대한 설명은 다음과 같습니다.

연산자	설명
DELETE	테이블에서 행을 지웁니다.
FETCH	특정 레코드 ID를 사용하여 테이블에서 컬럼을 페치합니다.
FILTER	한 개 이상의 조건으로 데이터를 필터합니다.
GRPBY	지정된 컬럼과 함수의 공통값에 따라 행을 구분하고, 집합 함수를 평가합니다.
HSJOIN	둘 이상의 테이블이 조인 컬럼에서 해시된 해시 조인을 표시합니다.
INSERT	테이블에 행을 삽입합니다.
IXAND	두 개 이상의 인덱스를 스캔하여 나온 RID에 대해 AND 작업을 수행합니다.
IXSCAN	테이블의 인덱스를 스캔하여 일련의 순서화된 행을 산출합니다.
MSJOIN	병합 조인으로 외부 및 내부 테이블이 모두 조인-술어 순서로 되어야 합니다.
NLJOIN	중첩 루프 조인으로 외부 테이블 행마다 한 번씩 내부 테이블에 액세스합니다.
RETURN	쿼리로부터 사용자에게 데이터를 리턴함을 의미합니다.
RIDSCN	한 개 이상의 인덱스로부터 얻어진 일련의 행 ID(RID)를 스캔합니다.
SORT	지정된 컬럼의 순서로 행을 정렬하며, 중복된 항목을 제거할 수 있습니다.
TBSCAN	데이터 페이지에서 직접 필요한 모든 데이터를 읽어 행을 검색합니다.
TEMP	여러 번 다시 읽을 수 있도록 데이터를 임시 테이블에 저장합니다.
TQUEUE	데이터베이스 에이전트들 사이에 테이블 데이터를 전송합니다.
UNION	여러 테이블의 행을 이어 붙입니다.
UNIQUE	특정 컬럼에 대해 중복된 값을 가진 행을 제거합니다.
UPDATE	테이블에 있는 행을 갱신합니다.

TOPIC 22-16 Visual Explain

Point

제어 센터를 이용하면 GUI 형식의 익스플레인 도구인 Visual Explain 을 이용하여 SQL문의 액세스 플랜에 대한 익스플레인 정보를 확인할 수 있습니다.

Tip

과거에 실행했던 동적 SQL문에 대한 익스플레인 정보는 실행 기록 정보를 통해서 다시 확인할 수 있습니다.

Tip

정적 SQL문에 대한 익스플레인 정보는 실행 기록 정보에서 확인할 수 있습니다. PREP 또는 BIND 명령어로 패키지를 생성할 때, EXPLSNAP 옵션이 YES 또는 ALL로 설정되어야 합니다.

6 제어 센터에서 원하는 데이터베이스를 선택한 후, Explain된 명령문 실행 기록 표시 메뉴를 선택하면 지금까지 실행했던 익스플레인 정보를 확인할 수 있습니다.

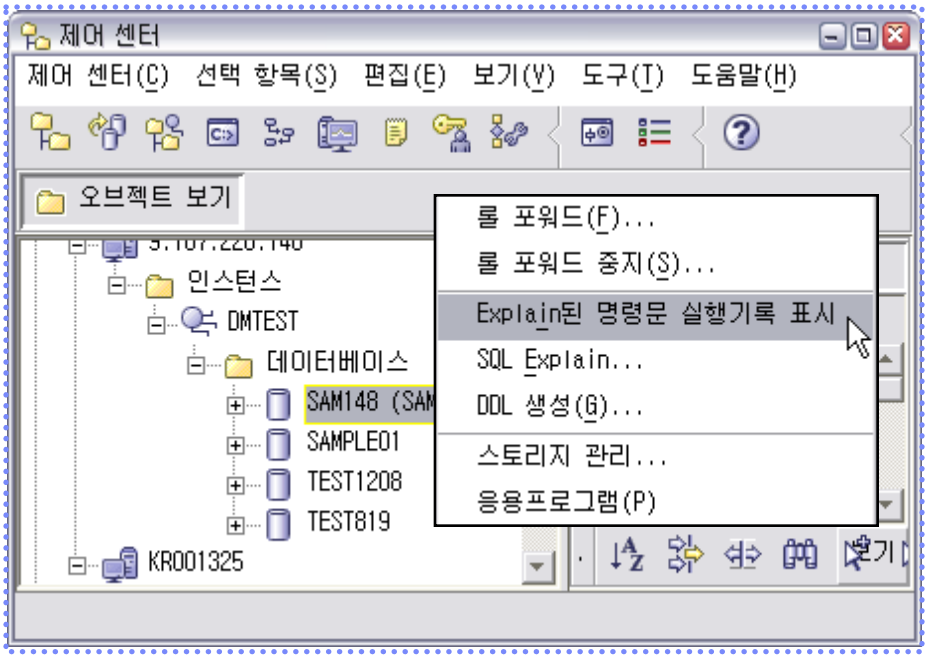


Figure 22-16E ••• Explain된 명령 실행 기록 메뉴 선택

7 원하는 실행 기록 항목을 선택하여 해당하는 SQL 텍스트를 확인하거나, 기존의 액세스 플랜을 다시 표시할 수 있습니다. 쿼리 태그를 변경하거나, 실행 기록을 제거할 수 있습니다.

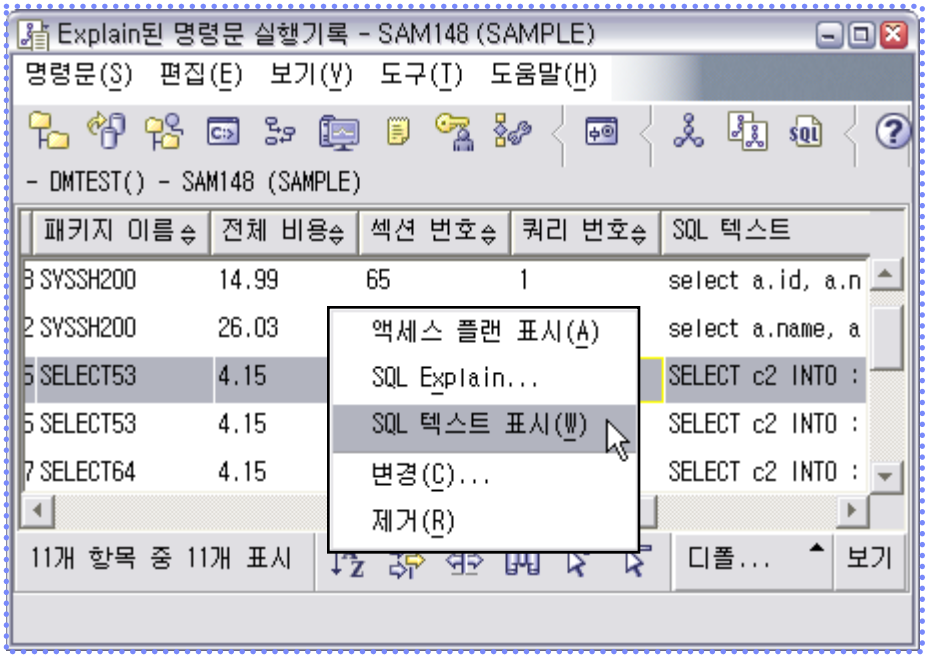


Figure 22-16F ••• 동적 또는 정적 SQL문에 대한 익스플레인 실행 기록 선택

TOPIC 22.17 db2expln 유틸리티

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip

dynexpln 명령어는 매개 변수 표시 문자로 사용되는 ? (물음표)를 포함한 동적 SQL문을 지원하지 않으므로, db2expln 명령어를 사용하도록 합니다.

1 db2expln 명령어를 이용하여 옵티마이저가 선택한 액세스 플랜을 표시합니다.

```
>>-db2expln--+-----------------------+--+-------------------+-------------->
             '-| connection-options |-'  '-| output-options |-'

>--+---------------------+--+---------------------+--+---------------------+->
   '-| package-options |-'  '-| dynamic-options |-'  '-| explain-options |-'
```

Connection Options:
-d <db> = Connect to the database named <db>.
-u <name> <pw> = Connect as user <name> with password <pw>.

Output Options:
-t = Send output to the terminal.
-o <file> = Write output to the file named <file>.

Package Options:
-c <pattern> = The package creator must match <pattern>.
-p <pattern> = The package name must match <pattern>.
-s <number> = The section number is <number>. Use 0 (zero) for all.

Dynamic Statement Options:
-q <sql> = The dynamic statement <statement> will be explained.
-f <file> = The dynamic statements contained in the <file> will be explained.

Explain Options:
-z <character> = Each SQL statement ends at <character>.
-g = Reconstruct the original optimizer plan graph.
-i = Show the operator ID numbers.

Figure 2217A db2expln 명령어

Tip

섹션 번호에 0을 지정하면 모든 섹션에 대한 정보를 확인할 수 있습니다.

Tip

동적 SQL문을 포함하는 –q 옵션과 –f 옵션은 함께 사용할 수 없습니다.

Tip

-z 옵션을 지정할 때는 공백 문자를 입력한 후에, " (쌍따옴표) 사이에 SQL 구분 문자를 지정합니다.

Tip

출력 방향을 지정하는 -o 옵션과 –t 옵션은 함께 사용할 수 없습니다.

2 db2expln 명령어의 옵션에 대한 설명은 다음과 같습니다.

옵 션	설 명
-d <DB명>	DB명을 지정합니다.
-u <사용자명> <암호>	데이터베이스에 접속할 사용자명과 암호를 지정합니다.
-c <스키마명>	패키지의 소유자를 지정합니다.
-p <패키지명>	패키지명을 지정합니다.
-s <섹션 번호>	섹션 번호를 지정합니다.
-f <입력 파일명>	동적 SQL문이 포함된 입력 파일명을 지정합니다.
-q <SQL문>	동적 SQL문을 직접 입력합니다.
-z "<SQL문 구분자>"	SQL문의 구분자를 지정합니다. 기본은 ; (세미콜론) 입니다.
-o <출력 파일명>	출력 파일명을 지정합니다.
-t	화면으로 익스플레인 정보를 표시합니다.
-g -i	텍스트 형식의 트리 형태의 액세스 플랜을 포함시킵니다.

TOPIC 22-18 동적 SQL문에 대한 db2expln 출력

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip

-q 옵션은 한 개의 동적 SQL문을 지정할 수 있습니다.

Tip

-f 옵션은 한 개 이상의 동적 SQL문을 지정할 수 있습니다. SQL문이 한 개 이상이거나, 한 줄 이상에 걸쳐서 표현될 때에는 반드시 –z 옵션을 이용하여 SQL 구분자를 지정해야 합니다. 일반적으로, SQL문의 구분자로 기본값인 ; (세미 콜론)을 사용합니다.

Tip

입력된 동적 SQL문에 대한 분리 수준, 블럭킹 옵션, 병렬 처리 여부 등의 정보가 표시됩니다.

Tip

입력된 동적 SQL문과 실행 비용, 액세스 플랜 등의 익스플레인 정보가 표시됩니다.

3 db2expln 명령어에서 –q 옵션을 이용하여 동적 SQL문을 직접 입력하면, 해당하는 동적 SQL문에 대한 익스플레인 정보를 확인할 수 있습니다.

```
$ db2expln –d <데이터베이스명> -q <동적 SQL문> -t -g -i
$ db2expln –d <데이터베이스명> -q <동적 SQL문> -o <출력 파일명> -g -i
```

4 db2expln 명령어에서 –f 옵션을 이용하여 동적 SQL문이 저장된 파일명을 입력하면, 해당하는 동적 SQL문에 대한 익스플레인 정보를 확인할 수 있습니다.

```
$ cat <입력 파일명>
<SQL문>;
$ db2expln –d <데이터베이스명> -f <입력 파일명> -z ";" -t -g -i
$ db2expln –d <데이터베이스명> -f <입력 파일명> -z ";" -o <출력 파일명> -g -i
```

5 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
$ db2expln -d sample -q "select * from kes.empl where id=1" -t -g

DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM
Corp. 1991, 2002
Licensed Material - Program Property of IBM
IBM DB2 Universal Database SQL Explain Tool

DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM
Corp. 1991, 2002
Licensed Material - Program Property of IBM
IBM DB2 Universal Database SQL Explain Tool

******************** DYNAMIC ***************************************

============== STATEMENT ============================

      Isolation Level               = Cursor Stability
      Blocking                      = Block Unambiguous Cursors
      Query Optimization Class      = 5
      Partition Parallel            = No
      Intra-Partition Parallel      = No
      SQL Path  = "SYSIBM", "SYSFUN", "SYSPROC", "POST01"

SQL Statement:

 select name
 from kes.empl
 where id =1

<다음 페이지에 계속됨>
```

TOPIC 22.18 동적 SQL문에 대한 db2expln 출력

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip

DPF 환경에서는 데이터베이스 파티션간의 정보 이동이 포함되므로, 동일한 쿼리에 대한 익스플레인 정보가 더 복잡하게 표시될 수 있습니다. 검색되는 데이터에 대해 list prefetch 기법이 선택된다면, 다음과 같은 액세스 플랜이 표시될 수도 있습니다.

```
      RETURN
       ( 1)
         |
        BTQ
       ( 2)
         |
       FETCH
       (----)
      /     \
    RIDSCN  Table:
    ( 4)    KES
      |     EMPL
     SORT
    ( 5)
      |
    IXSCAN
    ( 6)
   /       \
Index:     Table:
KES        KES
EMPL_PK01  EMPL
```

6 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
<이전 페이지에서 계속됨>

Estimated Cost = 12.869981
Estimated Cardinality = 1.000000

(   2) Access Table Name = KES.EMPL  ID = 2,15
        |  Index Scan:  Name = KES.EMPL_PK01  ID = 1
        |  |  Regular Index (Not Clustered)
        |  |  Index Columns:
        |  |  |  1: ID (Ascending)
        |  #Columns = 7
        |  Single Record
        |  Fully Qualified Unique Key
        |  #Key Columns = 1
        |  |  Start Key: Inclusive Value
        |  |  |  |  1: 1
        |  |  Stop Key: Inclusive Value
        |  |  |  |  1: 1
        |  Data Prefetch: Eligible 0
        |  Index Prefetch: None
        |  Lock Intents
        |  |  Table: Intent Share
        |  |  Row  : Next Key Share
        |  Sargable Predicate(s)
(   2) |  |  Return Data to Application
        |  |  |  #Columns = 7
(   1) Return Data Completion

End of section


Optimizer Plan:

      RETURN
      ( 1)
        |
      FETCH
      ( 2)
     /     \
  IXSCAN   Table:
  ( 2)     KES
    |      EMPL
Index:
KES
EMPL_PK01
```

TOPIC 22 19 정적 SQL문에 대한 db2expln 출력

Point 패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip 일반적으로, 섹션 번호에 0을 입력하여 패키지에 포함된 모든 섹션에 대한 익스플레인 정보를 확인합니다.

7 정적 SQL문은 이미 컴파일되어 시스템 카탈로그에 패키지로 저장되어 있습니다. db2expln 명령어에서 –p와 –s옵션을 이용하여 정적 SQL문이 포함된 패키지명과 섹션 번호를 입력하면, 해당하는 정적 SQL문에 대한 익스플레인 정보를 확인할 수 있습니다.

```
$ db2expln –d <데이터베이스명> -c <스키마명> -p <패키지명> -s <섹션번호> -t
```

8 정적 SQL문에 대한 익스플레인을 확인하기 위해 ESQL 소스 파일을 작성합니다. 소스 파일명은 select64.sqc 로 하겠습니다.

```
#include <stdio.h>

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
        char c2[30+1];
EXEC SQL END DECLARE SECTION;

int main()
{
        EXEC SQL CONNECT TO sample;

        EXEC SQL    SELECT name
                    INTO :name
                    FROM kes.empl
                    WHERE id = 1;

        printf("name = %s\n",name);

        EXEC SQL CONNECT RESET;

    exit(0);
}
```

Tip 익스플레인 정보만 간단하게 확인하려면, BIND 파일을 생성하지 않고, PREP 명령어로 패키지를 바로 생성합니다.

9 PREP 명령어를 이용하여 패키지를 생성합니다.

```
$ db2 connect to sample
$ db2 prep select64.sqc
```

10 패키지를 생성하는 사용자명이 post01 이라면, 패키지명은 POST01.SELECT64 이 됩니다. list packages 명령어로 패키지에 관련된 정보를 확인합니다.

```
$ db2 list packages
                                    바인드     총                         분리
패키지        스키마      버전     방법       섹션     유효   형식      레벨    블로킹
-----------------------------------------------------------------------------------------
SELECT64    POST01              POST01       1      Y      0        CS      U

1 레코드가 선택됨.
```

TOPIC 22.19 정적 SQL문에 대한 db2expln 출력

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

11 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
$ db2expln -d sample -c post01 -p select64 -s 0 -t -g

DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM Corp.
1991, 2002
Licensed Material - Program Property of IBM
IBM DB2 Universal Database SQL Explain Tool

DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM Corp.
1991, 2002
Licensed Material - Program Property of IBM
IBM DB2 Universal Database SQL Explain Tool

******************** PACKAGE ****************************************

Package Name = "POST01"."SELECT64"  Version = ""

      Prep Date = 2006/05/22
      Prep Time = 14:05:46

      Bind Timestamp = 2006-05-22-14.05.46.565000

      Isolation Level                = Cursor Stability
      Blocking                       = Block Unambiguous Cursors
      Query Optimization Class       = 5

      Partition Parallel             = No
      Intra-Partition Parallel       = No

      SQL Path  = "SYSIBM", "SYSFUN", "SYSPROC", "POST01"

---------- SECTION ----------------------------------------
Section = 1


SQL Statement:

 SELECT name INTO :H00002
 FROM kes.empl
 WHERE id =1


Section Code Page = 970

Estimated Cost = 12.869981
Estimated Cardinality = 1.000000

<다음 페이지에 계속됨>
```

Tip

패키지명을 비롯하여 분리 수준, 블럭킹 옵션, 병렬 처리 여부 등의 패키지 정보가 표시됩니다.

Tip

패키지에 포함된 섹션별로 SQL문과 실행 비용, 액세스 플랜 등의 익스플레인 정보가 표시됩니다.

TOPIC 22.19 정적 SQL문에 대한 db2expln 출력

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip

DPF 환경에서는 데이터베이스 파티션간의 정보 이동이 포함되므로, 동일한 쿼리에 대한 익스플레인 정보가 더 복잡하게 표시될 수 있습니다.

```
    RETURN
     ( 1)
      |
     BTQ
     ( 2)
     |
     FETCH
     ( 3)
     /   \
  IXSCAN  Table:
  ( 3)    KES
    |     EMPL
Index:
KES
EMPL_PK01
```

12 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
Access Table Name = KES.EMPL  ID = 2,15
|  Index Scan:  Name = KES.EMPL_PK01  ID = 1
|  |  Regular Index (Not Clustered)
|  |  Index Columns:
|  |  |  1: ID (Ascending)
|  #Columns = 1
|  Single Record
|  Fully Qualified Unique Key
|  #Key Columns = 1
|  |  Start Key: Inclusive Value
|  |  |  |  1: 1
|  |  Stop Key: Inclusive Value
|  |  |  |  1: 1
|  Data Prefetch: Eligible 0
|  Index Prefetch: None
|  Lock Intents
|  |  Table: Intent Share
|  |  Row  : Next Key Share
|  Sargable Predicate(s)
|  |  Return Data to Application
|  |  |  #Columns = 1
Return Data Completion

End of section


Optimizer Plan:

      RETURN
      ( 1)
        |
      FETCH
      ( 2)
      /   \
   IXSCAN   Table:
   ( 2)     KES
     |      EMPL
Index:
KES
EMPL_PK01
```

TOPIC 2220 db2exfmt 유틸리티

Point

익스플레인 테이블의 내용을 정해진 형식의 텍스트로 생성하는 도구입니다. 정적 SQL문과 동적 SQL문에 대한 액세스 플랜 정보와 옵티마이저에 대한 정보를 확인하는데 사용됩니다.

Tip

실행하는 사용자의 ID를 스키마명으로 하는 익스플레인 테이블이 있어야 합니다.

Tip

-e 옵션을 이용하여 익스플레인 테이블의 스키마명을 지정하지 않으면, DB에 접속한 사용자 ID가 스키마명으로 이용됩니다.

Tip

-g 옵션의 세부 옵션 설명입니다.
O – 그래프만 생성
T – 오퍼레이션별 총비용을 표시
I – I/O 비용을 표시
C – 예측되는 데이터 건수를 표시

Tip

-n 옵션과 –s 옵션에서는 %, * 등의 wild charcter를 이용하여 표시할 수 있습니다.

Tip

출력 방향을 지정하는 -o 옵션과 –t 옵션은 함께 사용할 수 없습니다.

Tip

-# 옵션에서 0을 지정하면, 모든 섹션에 대한 정보를 표시합니다.

1 필요시 <인스턴스 사용자의 홈디렉토리>/sqllib/misc/EXPLAIN.DDL 을 실행하여 익스플레인 테이블을 생성합니다.

```
$ db2 connect to <DB명>
$ db2 -tvf $HOME/sqllib/misc/EXPLAIN.DDL
```

2 db2exfmt 명령어를 이용하여 익스플레인 테이블에 저장된 정보를 파일로 저장합니다.

Figure 2220A db2exfmt 명령어

3 db2exfmt 명령어의 옵션에 대한 설명은 다음과 같습니다.

옵 션	설 명
-d <DB명>	패키지가 저장된 데이터베이스명을 지정합니다.
-e <스키마명>	익스플레인 테이블의 스키마명을 지정합니다.
-g OTIC	액세스 플랜을 트리 형식의 텍스트 그래프로 표시합니다.
-1	영문자 L 의 소문자로 패키지명의 대소문자를 무시합니다.
-n %	익스플레인의 대상이 되는 오브젝트명을 지정합니다. %를 지정하면, 모든 오브젝트를 대상으로 합니다.
-s %	익스플레인의 대상이 되는 스키마명을 지정합니다. %를 지정하면, 모든 스키마를 대상으로 합니다.
-o <출력 파일명>	익스플레인 정보를 저장할 출력 파일명을 지정합니다.
-t	익스플레인 정보를 화면으로 표시합니다.
-u <사용자명> <암호>	데이터베이스에 접속할 사용자명과 암호를 지정합니다.
-w	익스플레인된 시간 소인을 지정합니다. -1(숫자 1)을 지정하면, 최근에 실행된 SQL문에 대한 익스플레인 정보를 표시합니다.
-# <섹션 번호>	섹션 번호를 지정합니다.

TOPIC 22.21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

Tip

PREP 또는 BIND 명령어에서 지정한 EXPLAIN 옵션의 값이 ALL이고, CURRENT EXPLAIN MODE 레지스터 값이 NO이면, 실행시에 동적 SQL문의 익스플레인 정보를 캡쳐하여 저장합니다.

Tip

CURRENT EXPLAIN MODE 레지스터의 값이 NO가 아닌 경우에는 PREP 또는 BIND 명령어에서 지정한 EXPLAIN 옵션 값은 무시됩니다.

1 동적 SQL문에 대한 익스플레인 정보를 확인하려면, SQL문을 실행하기 전에 특수 레지스터인 CURRENT EXPLAIN MODE를 YES 또는 EXPLAIN 으로 설정합니다.

```
$ db2 set CURRENT EXPLAIN MODE <YES 또는 EXPLAIN>
$ db2 "SQL문"
```

2 CURRENT EXPLAIN MODE에 사용될 수 있는 특수 레지스터리의 값은 다음과 같습니다.

옵션	설명
NO	Explain 기능을 작동 불가능으로 설정합니다. 익스플레인 정보를 캡처하지 않으며, 기본값으로 사용됩니다.
YES	Explain 기능을 작동 가능으로 설정합니다. 동적 SQL문에 대한 익스플레인 정보를 캡처하여 익스플레인 테이블에 삽입하며, 동적 SQL문이 컴파일되고 실행됩니다.
EXPLAIN	Explain 기능을 작동 가능으로 설정합니다. 동적 SQL문에 대한 익스플레인 정보를 캡처하여 저장하며, 동적 SQL문 중에서 DML은 실행되지 않습니다.
REOPT	Explain 기능을 작동 가능으로 설정합니다. 실행 시간에 명령문 재최적화 되는 경우 (호스트 변수, 특수 레지스터, 매개 변수 표시 문자에 대한 실제 값이 사용 가능할 때)에 정적 또는 동적 SQL문에 대한 익스플레인 정보가 캡처되도록 합니다. SQL문이 실행 시간에 재최적화 되지 않을 경우 (명령문에 입력 변수가 없거나 REOPT 바인드 옵션이 NONE으로 설정된 경우)에 익스플레인 정보는 캡처되지 않습니다. REOPT 바인드 옵션을 ONCE로 설정하면, 익스플레인 정보는 명령문이 초기에 재최적화될 때 한 번만 캡처됩니다.
RECOMMEND INDEXES	SQL 컴파일러가 인덱스를 권장할 수 있게 합니다. 실행되는 쿼리는 ADVISE_INDEX 테이블에 권장되는 인덱스를 추가하고, 인덱스 사용에 대한 설명 정보를 캡처하여 설명 테이블에 저장합니다. 명령문은 컴파일되지도 않고, 실행되지도 않습니다.
EVALUATE INDEXES	SQL 컴파일러가 인덱스를 평가할 수 있게 합니다. 평가될 인덱스를 ADVISE_INDEX 테이블로부터 읽어 EVALUATE = Y로 표시해야 합니다. 옵티마이저는 카탈로그로부터의 값에 기초하여 가상 인덱스를 생성합니다. 실행되는 쿼리는 가상 인덱스에 기초하여 예측된 통계를 사용하여 컴파일되고 최적화됩니다. 명령문은 실행되지 않습니다.
RECOMMEND PARTITIONINGS	컴파일러가 특정 쿼리에 의해 액세스되는 각 테이블에 대해 가장 좋은 파티션을 권장하도록 지정합니다. 결과는 ADVISE_PARTITION 테이블에 저장되며, 쿼리는 실행되지 않습니다.
EVALUATE PARTITIONINGS	컴파일러가 ADVISE_PARTITION 테이블에 지정된 가상 파티션을 사용하여 쿼리의 예상 성능을 평가하도록 합니다.

3 CURRENT EXPLAIN MODE 의 값을 EXPLAIN 으로 설정하여 동적 SQL문을 실제로 실행하지 않고, 익스플레인 정보만 확인할 수 있습니다.

```
$ db2 set current explain mode EXPLAIN
```

TOPIC 22.21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

Tip

- 익스플레인 정보를 확인한 후에는 반드시 CURRENT EXPLAIN MODE 레지스터의 값을 NO 또는 YES로 설정되어야 SQL문을 실행할 수 있습니다.

4 익스플레인을 원하는 동적 SQL문을 실행합니다.

```
$ db2 "select name from kes.empl where id = 1"
SQL0217W  Explain 정보 요청만 처리 중이므로 명령문이 실행되지 않았습니다.
SQLSTATE=01604
```

5 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
$ db2exfmt -d sample -g TIC -w -1 -n % -s % -# 0 –t

DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM
Corp. 1991, 2002
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

******************** EXPLAIN INSTANCE ********************

DB2_VERSION:                    08.02.3
SOURCE_NAME:                    SQLC2E06
SOURCE_SCHEMA:                  NULLID
SOURCE_VERSION:
EXPLAIN_TIME:                   2006-05-22-16.29.01.594001
EXPLAIN_REQUESTER:              KR001325

Database Context:
----------------
        Parallelism:            None
        CPU Speed:              3.778754e-007
        Comm Speed:             0
        Buffer Pool size:       250
        Sort Heap size:         256
        Database Heap size:     600
        Lock List size:         50
        Maximum Lock List:      22
        Average Applications:   1
        Locks Available:        1122

Package Context:
---------------
        SQL Type:               Dynamic
        Optimization Level:     5
        Blocking:               Block All Cursors
        Isolation Level:        Cursor Stability

---------------- STATEMENT 1  SECTION 201 ----------------
        QUERYNO:          33

<다음 페이지에 계속됨>
```

TOPIC 22.21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

6 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
        QUERYTAG:         CLP
        Statement Type:   Select
        Updatable:        No
        Deletable:        No
        Query Degree:     1

Original Statement:
------------------
select name
from kes.empl
where id=1

Optimized Statement:
--------------------
SELECT Q1."NAME" AS "NAME"
FROM KES.EMPL AS Q1
WHERE (Q1."ID" = 1)

Access Plan:
-----------
        Total Cost:         12.87
        Query Degree:       1

          Rows
          RETURN
          (   1)
           Cost
            I/O
            |
            1
          FETCH
          (   2)
           12.87
            1
        /----+---\
      1          4
    IXSCAN    TABLE: KES
    (   3)              EMPL
   0.0155141
      0
      |
      4
 INDEX: KES
           EMPL_PK01

<다음 페이지에 계속됨>
```

TOPIC 22-21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

7 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
Extended Diagnostic Information:
--------------------------------
Diagnostic Identifier:     1
Diagnostic Details: EXP0022W  Index has no statistics.
The index "KES "."EMPL_PK01" has not had runstats run on it. This can
lead to poor cardinality and predicate filtering estimates.
Diagnostic Identifier:     2
Diagnostic Details: EXP0020W  Table has no statistics.  The table "KES
"."EMPL" has not had runstats run on it. This can lead to poor
cardinality and predicate filtering estimates.

Plan Details:
-------------
        1) RETURN: (Return Result)
                Cumulative Total Cost:          12.87
                Cumulative CPU Cost:            52876
                Cumulative I/O Cost:            1
                Cumulative Re-Total Cost:       0.00280195
                Cumulative Re-CPU Cost:         7415
                Cumulative Re-I/O Cost:         0
                Cumulative First Row Cost:      12.8696
                Estimated Bufferpool Buffers:   2

                Arguments:
                ---------
                BLDLEVEL: (Build level)   DB2 v8.1.10.812 : s050811
                STMTHEAP: (Statement heap size)    2048

                Input Streams:
                -------------
                        4) From Operator #2
                                Estimated number of rows:     1
                                Number of columns:            1
                                Subquery predicate ID: Not Applicable
                                Column Names:
                                ------------
                                +Q2."NAME"

        2) FETCH : (Fetch)
                Cumulative Total Cost:          12.87
                Cumulative CPU Cost:            52876
                Cumulative I/O Cost:            1
                Cumulative Re-Total Cost:       0.00280195
                Cumulative Re-CPU Cost:         7415

<다음 페이지에 계속됨>
```

TOPIC 22-21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

8 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
Cumulative Re-I/O Cost:        0
Cumulative First Row Cost:     12.8696
Estimated Bufferpool Buffers:  2

Arguments:
---------
MAXPAGES: (Maximum pages for prefetch)
        ALL
MAXPAGES: (Maximum pages for prefetch)
        ALL
PREFETCH: (Type of Prefetch)
        NONE
ROWLOCK : (Row Lock intent)
        NEXT KEY SHARE
TABLOCK : (Table Lock intent)
        INTENT SHARE
TBISOLVL: (Table access Isolation Level)
        CURSOR STABILITY

Input Streams:
-------------
        2) From Operator #3
                Estimated number of rows:     1
                Number of columns:            0
                Subquery predicate ID: Not Applicable

        3) From Object KES.EMPL
                Estimated number of rows:     4
                Number of columns:            1
                Subquery predicate ID: Not Applicable
                Column Names:
                ------------
                +Q1."NAME"

Output Streams:
--------------
        4) To Operator #1

                Estimated number of rows:     1
                Number of columns:            1
                Subquery predicate ID: Not Applicable
                Column Names:
                ------------
                +Q2."NAME"
```

<다음 페이지에 계속됨>

TOPIC 22 21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

9 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
        3) IXSCAN: (Index Scan)
                Cumulative Total Cost:          0.0155141
                Cumulative CPU Cost:            41056
                Cumulative I/O Cost:            0
                Cumulative Re-Total Cost:       0.00211421
                Cumulative Re-CPU Cost:         5595
                Cumulative Re-I/O Cost:         0
                Cumulative First Row Cost:      0.0155141
                Estimated Bufferpool Buffers:   1

                Arguments:
                ---------
                MAXPAGES: (Maximum pages for prefetch)
                        ALL
                PREFETCH: (Type of Prefetch)
                        NONE
                ROWLOCK : (Row Lock intent)
                        NEXT KEY SHARE
                SCANDIR : (Scan Direction)
                        FORWARD
                TABLOCK : (Table Lock intent)
                        INTENT SHARE

                Predicates:
                ----------
                2) Start Key Predicate
                        Relational Operator:            Equal (=)
                        Subquery Input Required:        No
                        Filter Factor:                  0.25

                        Predicate Text:
                        --------------
                        (Q1."ID" = 1)

                2) Stop Key Predicate
                        Relational Operator:            Equal (=)
                        Subquery Input Required:        No
                        Filter Factor:                  0.25

                        Predicate Text:
                        --------------
                        (Q1."ID" = 1)
```

<다음 페이지에 계속됨>

TOPIC 22.21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

10 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
                Input Streams:
                -------------
                        1) From Object KES.EMPL_PK01

                                Estimated number of rows:    4
                                Number of columns:           2
                                Subquery predicate ID: Not Applicable

                                Column Names:
                                ------------
                                +Q1.$RID$+Q1."ID"


                Output Streams:
                --------------
                        2) To Operator #2

                                Estimated number of rows:    1
                                Number of columns:           0
                                Subquery predicate ID: Not Applicable


Objects Used in Access Plan:
---------------------------

        Schema: KES
        Name:       EMPL_PK01
        Type:       Index
                    Time of creation:  2006-05-22-14.02.58.864000
                    Last statistics update:
                    Number of columns:                  1
                    Number of rows:                     4
                    Width of rows:                      -1
                    Number of buffer pool pages:        1
                    Distinct row values:                Yes
                    Tablespace name:                    USERSPACE1
                    Tablespace overhead:                12.670000
                    Tablespace transfer rate:           0.180000
                    Source for statistics:              Single Node
                    Prefetch page count:                32
                    Container extent page count:        32
                    Index clustering statistic:         20.000000
                    Index leaf pages:                   1
                    Index tree levels:                  1

<다음 페이지에 계속됨>
```

TOPIC 22.21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

11 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
                Index full key cardinality:                  4
                Index first key cardinality:                 4
                Index first 2 keys cardinality:              -1
                Index first 3 keys cardinality:              -1
                Index first 4 keys cardinality:              -1
                Index sequential pages:                      1
                Index page density:                          100
                Index avg sequential pages:                  -1
                Index avg gap between sequences:             -1
                Index avg random pages:                      -1
                Fetch avg sequential pages:                  -1
                Fetch avg gap between sequences:             -1
                Fetch avg random pages:                      -1
                Index RID count:                             0
                Index deleted RID count:                     0
                Index empty leaf pages:                      0
                Base Table Schema:                           KES
                Base Table Name:                             EMPL
                Columns in index:                            ID

        Schema: KES
        Name:       EMPL
        Type:       Table
                    Time of creation:  2006-05-22-14.02.58.804001
                    Last statistics update:
                    Number of columns:                       7
                    Number of rows:                          4
                    Width of rows:                           37
                    Number of buffer pool pages:             1
                    Distinct row values:                     No
                    Tablespace name:                         USERSPACE1
                    Tablespace overhead:                     12.670000
                    Tablespace transfer rate:                0.180000
                    Source for statistics:                   Single Node
                    Prefetch page count:                     32
                    Container extent page count:             32
                    Table overflow record count:             0
                    Table Active Blocks:                     -1
```

TOPIC 22.22 정적 SQL문에 대한 db2exfmt 출력

Point

PREP 또는 BIND 명령의 EXPLAIN 옵션을 이용하여 정적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

Tip

EXPLAIN 세부 옵션으로 사용되는 ALL과 REOPT 옵션은 Windows와 UNIX용 DB2에서만 지원됩니다.

Tip

헤더 부분을 제외하면, 정적 SQL문에 대한 익스플레인 예제와 결과가 동일합니다.

1. 정적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하려면, PREP 또는 BIND 명령어로 패키지를 생성할 때, EXPLAIN 옵션을 YES 또는 ALL 으로 설정합니다.

```
$ db2 prep <ESQL 소스파일명> EXPLAIN <YES 또는 ALL>
$ db2 bind <바인드 파일명> EXPLAIN <YES 또는 ALL>
```

2. PREP 또는 BIND 명령어의 EXPLAIN 옵션의 세부 옵션은 다음과 같습니다.

옵션	설명
NO	익스플레인 정보를 캡처하지 않습니다.
YES	정적 SQL문에 대한 익스플레인 정보를 저장합니다.
ALL	정적 SQL문과 동적 SQL문에 대한 익스플레인 정보를 저장합니다.
REOPT	실행 시간에 명령문 재최적화 되는 경우 (호스트 변수, 특수 레지스터, 매개 변수 표시 문자에 대한 실제 값이 사용 가능할 때)에 정적 또는 동적 SQL문에 대한 익스플레인 정보가 캡처되도록 합니다.

3. PREP 명령어에서 EXPLAIN 옵션을 YES 로 설정하여 패키지를 생성하면, 정적 SQL문에 대한 익스플레인 정보가 익스플레인 테이블에 저장됩니다.

```
$ db2 connect to sample
$ db2 prep select64.sqc EXPLAIN YES
```

4. 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
$ db2exfmt  -d sample -g TIC -w -1 -n % -s % -# 0 –t
DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM Corp.
    1991, 2002
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

********************* EXPLAIN INSTANCE *********************
DB2_VERSION:                    08.02.3
SOURCE_NAME:                    SELECT64
SOURCE_SCHEMA:                  KR001325
SOURCE_VERSION:
EXPLAIN_TIME:                   2006-05-22-17.13.15.731000
EXPLAIN_REQUESTER:              KR001325

Database Context:
----------
<생략>
```

UNIT 23

프로그래밍

이 장에서는 DB2가 제공하는 ESQL, CLI, API, JDBC에 대한 전반적인 내용에 대해 소개합니다.

DB2 9.7 개발자 가이드

Developer Edition

- ESQL
- CLI
- API
- JDBC

TOPIC 2301 ESQL

Point

SQL과 프로그래밍 언어가 혼합된 방식인 ESQL로 구현된 응용프로그램을 빌드하려면 프리컴파일이 필요합니다. 프리컴파일러와 헤더 파일, 라이브러리를 제공하려면 개발 머신에 DB2 Application Development Client 모듈 이상을 설치해야 합니다.

Tip

ESQL은 Embedded SQL 의 약자입니다.

Tip

prep 명령어에서 BINDFILE 옵션이 없으면, bnd 파일을 생성하지 않고 패키지를 생성하므로 bind 명령어를 실행할 필요가 없습니다.

Tip

ESQL/C의 소스 파일의 확장자는 sqc입니다. C 컴파일러를 위한 PATH, LIBPATH 등의 환경 변수가 적절히 구성되었는지 확인하십시오.

Tip

ESQL 방식의 프로그램의 최종 실행 모듈은 바이너리 코드로 된 C 실행 모듈과 데이터베이스 시스템 카탈로그에 저장된 팩키지입니다. C 실행 모듈은 SQL문에 대한 액세스 플랜을 가진 팩키지를 호출합니다.

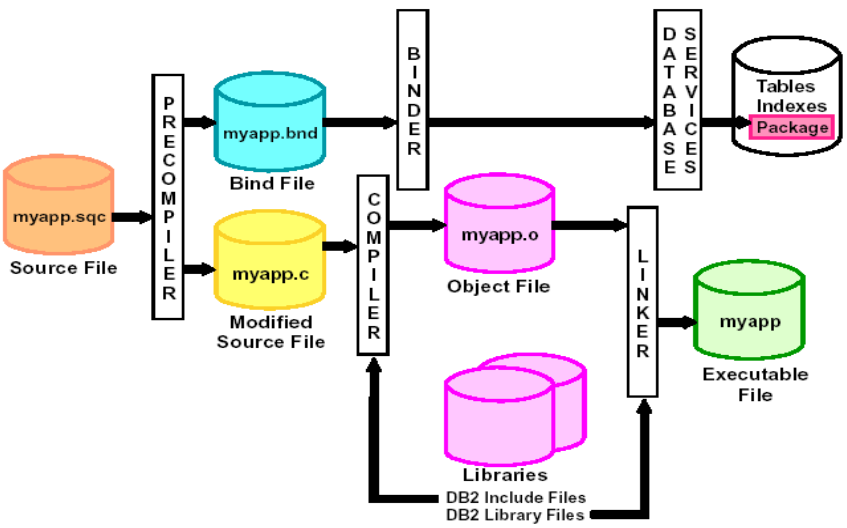
Tip

프로그램의 로직을 표현하는 프로그래밍 언어를 호스트 언어라고 합니다.

Tip

SQL문은 시작과 종료를 나타내는 특수한 구분자를 이용하여 표현합니다. C 언어를 사용하는 경우에는 EXEC SQL 로 시작하고, ; (세미콜론)으로 종료합니다.

1 ESQL은 데이터의 액세스는 SQL문으로 구현하고, 프로그램의 로직은 프로그래밍 언어로 구현하는 혼합 방식입니다. 프리컴파일과 SQL문에 대한 바인드, 프로그래밍 언어에 대한 컴파일과 링크 과정을 통해서 팩키지와 바이너리 실행 파일을 생성합니다.




```
$ db2 connect to sample
$ db2 prep myapp.sqc BINDFILE
$ db2 bind myapp.bnd blocking all grant public
$ xlc -c myapp.c –I~db2inst1/sqllib/include
$ xlc -o myapp myapp.o -ldb2 –L~db2inst1/sqllib/lib
$ myapp
```

Figure 2301A ESQL/C 프로그램 빌드 과정

2 DB2 개발자로 로그인하여 ESQL/C 소스 파일을 작성합니다. 확장자를 sqc 로 지정합니다.

```
#include <stdio.h>

EXEC SQL INCLUDE SQLCA;

int main()
{
   EXEC SQL BEGIN DECLARE SECTION;
      char fname[13];
      char lname[16];
      char newfname[13];
   EXEC SQL END DECLARE SECTION;

   EXEC SQL CONNECT TO sample;
```

TOPIC 23 01 ESQL

Point

SQL과 프로그래밍 언어가 혼합된 방식인 ESQL로 구현된 응용프로그램을 빌드하려면 프리컴파일이 필요합니다. 프리컴파일러와 헤더 파일, 라이브러리를 제공하려면 개발 머신에 DB2 Application Development Client 모듈 이상을 설치해야 합니다.

```
    printf("Enter the lastname : ");
    gets(lname);

    EXEC SQL SELECT FIRSTNME INTO :fname
            FROM employee
            WHERE LASTNAME = :lname;

    if (SQLCODE == 0)
      printf("First name = %s\n", fname);
    else if (SQLCODE == 100)
      printf("Data is not found\n");
    else {
      printf("Select Error : SQLCODE = %ld\n", SQLCODE);
      EXEC SQL ROLLBACK;
      EXEC SQL CONNECT RESET;
      return (-1);
    }

    EXEC SQL COMMIT;
    EXEC SQL CONNECT RESET;

    return(0) ;
}
```

Tip

프로그램을 실행할 때는 팩키지에 대한 EXECUTE 권한이 있어야 합니다.

Tip

~<인스턴스명>은 DB2 인스턴스 개발자의 홈디렉토리를 의미합니다.

3 prep 명령어로 프리컴파일하고, bind 명령어로 팩키지를 생성합니다.

```
$ db2 connect to <데이터베이스명>
$ db2 prep <소스파일명>.sqc bindfile
$ db2 bind <소스파일명>.bnd blocking all grant public
```

4 C 컴파일러를 이용하여 컴파일과 링크를 실행하십시오.

```
$ xlc -c <소스파일명>.c –I~<인스턴스명>/sqllib/include
$ xlc -o <실행파일명> <소스파일명>.o -ldb2 –L~<인스턴스명>/sqllib/lib
```

5 바이너리 형식의 실행 모듈을 실행하십시오.

```
$ <실행파일명>
Enter the lastname : LEE
First name = Newman
```

TOPIC 23 02 CLI

Point

CLI는 C 라이브러리 형식으로 된 API를 이용하여 SQL문을 표현하는 프로그램 방식으로 직접적인 SQL문이 표현되어 있지 않으므로 프리컴파일은 필요하지 않으며, 이종 DBMS 간의 호환성이 높습니다. SQL문에 대한 패키지는 별도로 생성되지 않습니다.

Tip

CLI는 Call Level Interface의 약자입니다.

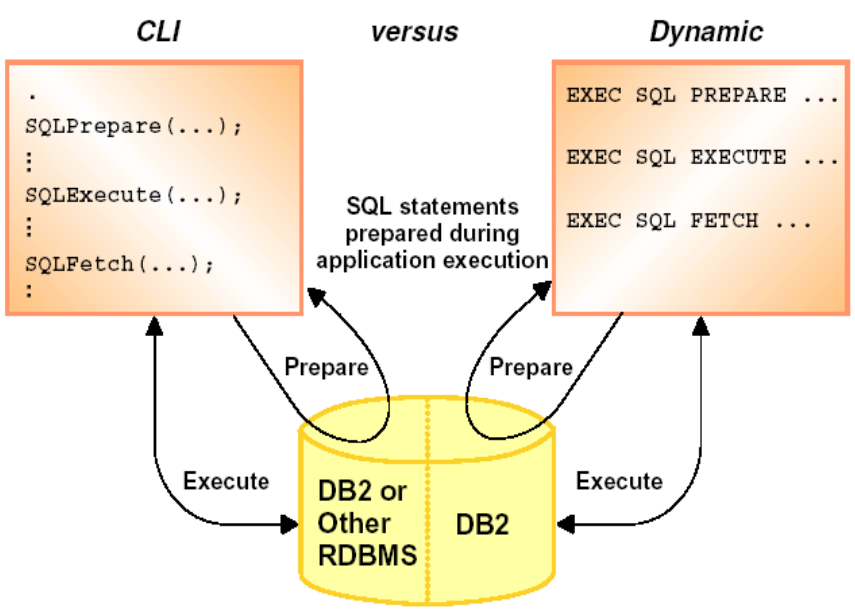
Tip

CLI의 소스 파일의 확장자는 c입니다. C 컴파일러를 위한 PATH, LIBPATH 등의 환경 변수가 적절히 구성되었는지 확인하십시오.

Tip

최종 실행 모듈은 바이너리 코드로 된 C 실행 모듈입니다. C 실행 모듈은 CLI를 위해 데이터베이스 시스템에 내장된 패키지를 호출합니다.

1 CLI는 데이터를 액세스하는 SQL문은 C 라이브러리 형식으로 된 API 로 구현하고, 프로그램의 로직은 C 언어로 구현하는 방식입니다. C 언어에 대한 컴파일과 링크 과정을 통해서 바이너리 실행 파일을 생성합니다. ESQL 에서 지원하는 동적 SQL문에 대응하는 API 가 모두 지원됩니다.




```
$ xlc -c mycli.c –I~db2inst1/sqllib/include
$ xlc -o mycli mycli.o -ldb2 –L~db2inst1/sqllib/lib
$ mycli
```

Figure 2302A CLI 와 ESQL

2 DB2 개발자로 로그인하여 CLI 소스 파일을 작성합니다. 확장자를 c 로 지정합니다.

```
#include <stdio.h>
#include <sqlcli1.h>

main() {

  SQLHANDLE   henv;
  SQLHANDLE   hdbc;
  SQLHANDLE   hstmt;

  SQLCHAR * stmt = ( SQLCHAR * ) "SELECT deptnumb, location
   FROM org";
  SQLRETURN   sqlrc = SQL_SUCCESS;
```

TOPIC 23 02 CLI

Point

CLI는 C 라이브러리 형식으로 된 API를 이용하여 SQL문을 표현하는 프로그램 방식으로 직접적인 SQL문이 표현되어 있지 않으므로 프리컴파일은 필요하지 않으며, 이종 DBMS 간의 호환성이 높습니다. SQL문에 대한 패키지는 별도로 생성되지 않습니다.

Tip

ESQL 방식의 프로그램에서 dynamic SQL로 구현했던 모든 로직은 대응되는 한 개 이상의 CLI로 변환될 수 있습니다. dynamic SQL이 지원하지 않는 기능을 구현하려면 CLI를 이용합니다.

```
struct
{  SQLINTEGER ind ;
   SQLSMALLINT val ;
} deptnumb ;

struct
{  SQLINTEGER ind ;
   SQLCHAR val[15] ;
} location ;

SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
SQLConnect(hdbc,  (SQLCHAR *)"SAMPLE", SQL_NTS, NULL,
 SQL_NTS, NULL, SQL_NTS);
SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);

SQLExecDirect( hstmt, stmt, SQL_NTS ) ;

printf("DEPTNUMB LOCATION     \n" ) ;
printf("-------- ------------\n" ) ;
sqlrc = SQLFetch( hstmt );

if (sqlrc == SQL_NO_DATA_FOUND)
     printf("\n    Data not found.\n");

while (sqlrc != SQL_NO_DATA_FOUND)
{
   sqlrc = SQLGetData( hstmt, 1, SQL_C_SHORT, &deptnumb.val,
                       0, &deptnumb.ind ) ;
   sqlrc = SQLGetData( hstmt, 2, SQL_C_CHAR, location.val,
                       15, &location.ind ) ;
   printf( "%-8d %-14.14s \n", deptnumb.val, location.val ) ;
   sqlrc = SQLFetch( hstmt );
}
```

Point

CLI는 C 라이브러리 형식으로 된 API를 이용하여 SQL문을 표현하는 프로그램 방식으로 직접적인 SQL문이 표현되어 있지 않으므로 프리컴파일은 필요하지 않으며, 이종 DBMS 간의 호환성이 높습니다. SQL문에 대한 팩키지는 별도로 생성되지 않습니다.

```
    SQLFreeStmt(hstmt, SQL_CLOSE);
    SQLEndTran(SQL_HANDLE_DBC, hdbc, SQL_COMMIT);
    SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
    SQLDisconnect(hdbc);
    SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
    SQLFreeHandle(SQL_HANDLE_ENV, henv);

    return(0);
}
```

Tip

~<인스턴스명>은 DB2 인스턴스 개발자의 홈디렉토리를 의미합니다.

3 C 컴파일러를 이용하여 컴파일과 링크를 실행하십시오.

```
$ xlc -c <소스파일명>.c –I~<인스턴스명>/sqllib/include
$ xlc -o <실행파일명> <소스파일명>.o -ldb2 –L~<인스턴스명>/sqllib/lib
```

4 바이너리 형식의 실행 모듈을 실행하십시오.

```
$ <실행파일명>
DEPTNUMB LOCATION
--------------------------------
      10 New York
      15 Boston
      20 Washington
      38 Atlanta
      42 Chicago
      51 Dallas
      66 San Francisco
      84 Denver
      10 New York
      15 Boston
      20 Washington
      38 Atlanta
      42 Chicago
      51 Dallas
      66 San Francisco
      84 Denver
```

TOPIC 2303 API

Point

API는 DB2 명령어에 대응하는 C 라이브러리 함수입니다. ESQL/C 소스에서 사용할 수 있으며, DB2가 제공하는 다양한 명령어에 대한 관리 API를 프로그램 로직과 함께 구현하는 방식입니다.

Tip

API는 Application Programming Interface의 약자입니다.

Tip

ESQL/C와 동일한 방식으로 빌드합니다.

1 API는 DB2 명령어를 C 라이브러리 형식으로 된 API 로 구현하는 방식입니다. 데이터 액세스에 SQL문을 사용하고, 프로그램 로직을 C 언어로 구현하는 것은 ESQL/C 와 동일합니다.

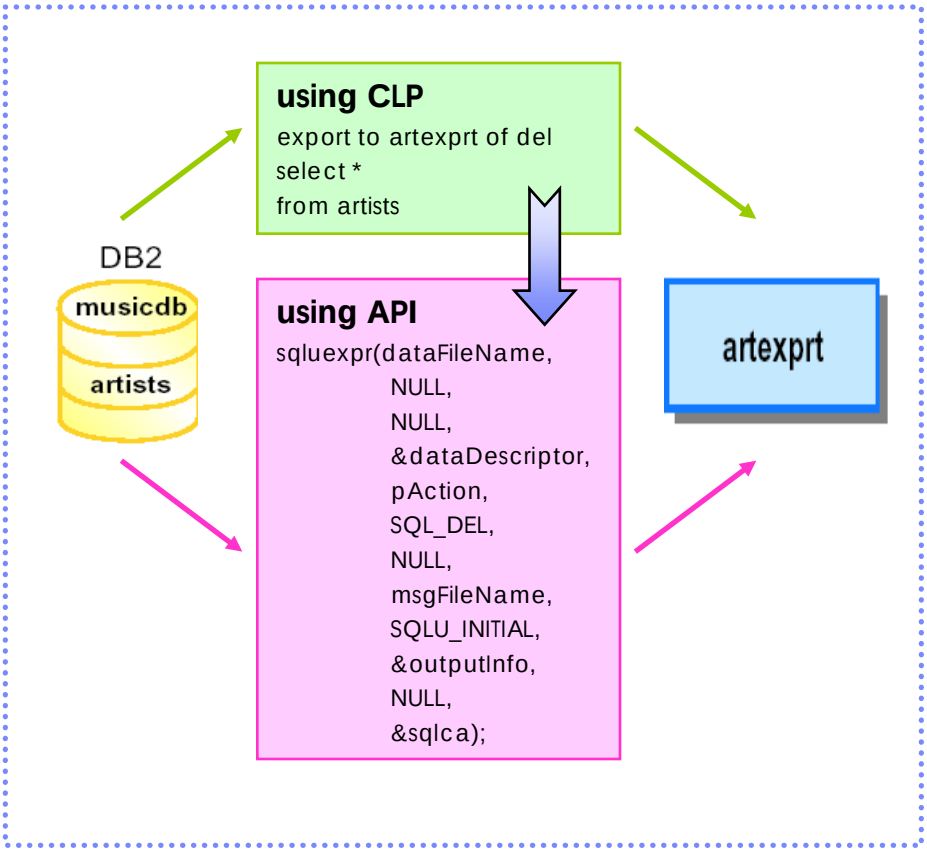


Figure 2303A • • • export 명령어와 sqluexpr 관리 API

2 DB2 개발자로 로그인하여 API 를 포함한 ESQL/C 소스 파일을 작성합니다. ESQL/C 소스이므로 확장자는 sqc 로 지정합니다.

```
$ login <DB2 개발자>
$ vi <소스파일명>.sqc
#include <stdio.h>
#include <stdlib.h>
#include <sqlutil.h>

int main()
{
 struct sqlca sqlca;

 char dataFileName[256];
```

TOPIC 23 03 API

Point

API는 DB2 명령어에 대응하는 C 라이브러리 함수입니다. ESQL/C 소스에서 사용할 수 있으며, DB2가 제공하는 다양한 명령어에 대한 관리 API를 프로그램 로직과 함께 구현하는 방식입니다.

Tip

소스에서 사용된 sqluexpr 은 DB2 명령어인 export 에 대응되는 API 입니다.

```
 struct sqldcol dataDescriptor;
 char actionString[256];
 struct sqlchar *pAction;
 char msgFileName[128];
 struct sqluexpt_out outputInfo;

 EXEC SQL connect to sample;

 strcpy(dataFileName, "ORG.DEL");
 dataDescriptor.dcolmeth = SQL_METH_D;
 strcpy(actionString, "SELECT deptnumb, deptname FROM org");
 pAction = (struct sqlchar *)malloc(sizeof(short) + sizeof(actionString)
   + 1);
 pAction->length = strlen(actionString);
 strcpy(pAction->data, actionString);
 strcpy(msgFileName, "tbexport.MSG");
 outputInfo.sizeOfStruct = SQLUEXPT_OUT_SIZE;

 printf("   client destination file name: %s\n", dataFileName);
 printf("   action                      : %s\n", actionString);
 printf("   client message file name    : %s\n", msgFileName);

 sqluexpr(dataFileName,
          NULL,
          NULL,
&dataDescriptor,
        pAction,
        SQL_DEL,
        NULL,
        msgFileName,
        SQLU_INITIAL,
        &outputInfo,
        NULL,
        &sqlca);
 free(pAction);
 EXEC SQL connect reset;
 return 0;
}
```

TOPIC 23 03 API

Point

API는 DB2 명령어에 대응하는 C 라이브러리 함수입니다. ESQL/C 소스에서 사용할 수 있으며, DB2가 제공하는 다양한 명령어에 대한 관리 API를 프로그램 로직과 함께 구현하는 방식입니다.

3 C 컴파일러를 이용하여 컴파일과 링크를 실행하십시오.

```
$ xlc -c <소스파일명>.c –I~<인스턴스명>/sqllib/include
$ xlc -o <실행파일명> <소스파일명>.o -ldb2 –L~<인스턴스명>/sqllib/lib
```

Tip

~<인스턴스명>은 DB2 인스턴스 개발자의 홈디렉토리를 의미합니다.

4 바이너리 형식의 실행 모듈을 실행하십시오.

```
$ <실행파일명>

$ cat ORG.DEL
10,"Head Office"
15,"New England"
20,"Mid Atlantic"
38,"South Atlantic"
42,"Great Lakes"
51,"Plains"
66,"Pacific"
84,"Mountain"
10,"Head Office"
15,"New England"
20,"Mid Atlantic"
38,"South Atlantic"
42,"Great Lakes"
51,"Plains"
66,"Pacific"
84,"Mountain"
```

TOPIC 2304 JDBC

Point

JDBC 드라이버는 DB2 클라이언트 제품에 포함되어 있습니다. JDBC type2, type4 를 지원하며, ESQL 방식인 SQLJ 도 지원합니다.

Tip

JDBC 드라이버는 DB2 클라이언트 제품에 포함되어 있습니다.

Tip

SQLJ는 다른 ESQL 프로그램과 유사한 방법으로 프리컴파일과 바인드 과정을 거쳐 팩키지가 생성됩니다. SQLJ용 Run-Time 클래스는 DB2 클라이언트 제품에 포함됩니다.

1 SQLJ 또는 JDBC 방식으로 작성합니다. 데이터 액세스는 JDBC에서 제공하는 데이터베이스와 관련된 메소드를 이용합니다.

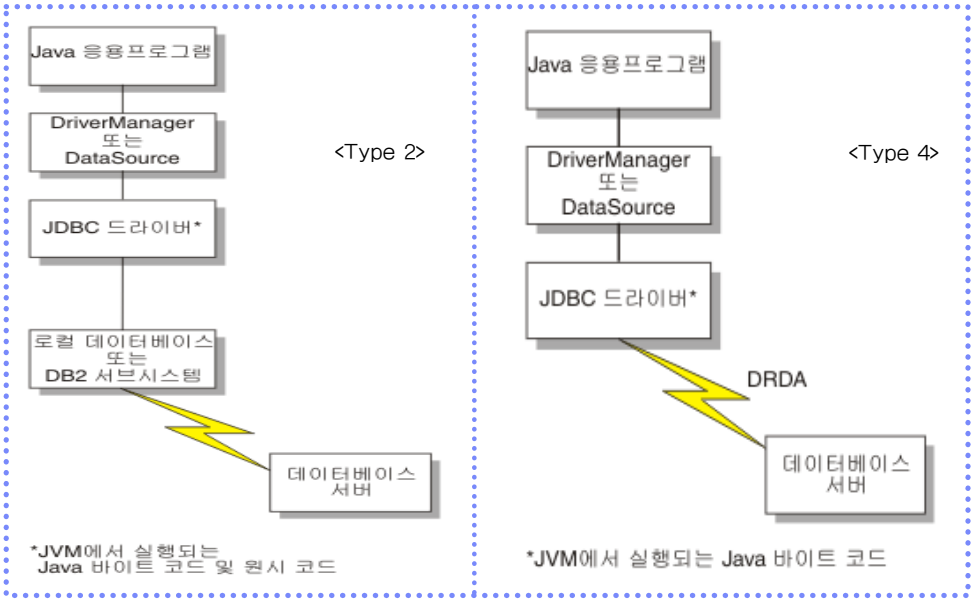


Figure 2304A JDBC 드라이버를 사용하는 자바 애플리케이션

2 DB2 개발자로 로그인하여 JAVA 소스 파일을 작성합니다. 확장자는 java 로 지정합니다.

```
$ login <DB2 개발자>
$ vi <소스파일명>.java
import java.sql.*;

class myjdbc {

  static {
    try {

    Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance
    ();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

  public static void main(String argv[]) {
    Connection con = null;
    String url = "jdbc:db2:sample";
```

TOPIC 23 04 JDBC

Point

JDBC 드라이버는 DB2 클라이언트 제품에 포함되어 있습니다. JDBC type2, type4 를 지원하며, ESQL 방식인 SQLJ 도 지원합니다.

Tip

JDBC 방식의 프로그램에서 구현하는 SQL문은 동적 SQL문으로 처리됩니다. SQLJ 방식을 이용하면, 정적 SQL문을 구현할 수 있습니다.

```
try {
        con = DriverManager.getConnection(url);
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * from employee");

        System.out.print("Received results:\n\n");
        while (rs.next()) {
          String a = rs.getString(1);
          String str = rs.getString(2);
          System.out.print(" empno= " + a + " firstname= " + str + "\n");
        }
        rs.close();
        stmt.close();
        con.close();
     } catch( Exception e ) {
        e.printStackTrace();
     }
  }
}
```

3 Java 버전을 확인하십시오.

```
$ java -version
java version "1.3.1“
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1)
```

4 Java 버전에 맞는 컴파일러가 실행되고 있는지 확인하십시오.

```
$ which javac
/usr/java13_64/bin/javac
```

5 PATH, CLASSPATH 환경 변수가 적절히 구성되었는지 확인하십시오.

```
$ echo $PATH
$ echo $CLASSPATH
```

6 Javac 명령어로 컴파일합니다.

```
$ javac <소스파일명>.java
```

Point

JDBC 드라이버는 DB2 클라이언트 제품에 포함되어 있습니다. JDBC type2, type4 를 지원하며, ESQL 방식인 SQLJ 도 지원합니다.

7 java 명령어로 자바 클래스 파일을 실행합니다. 실행 결과는 다음과 같습니다.

```
$ java <클래스파일명>

Received results:

empno=000010 firstname=CHRISTINE
empno=000020 firstname=MICHAEL
empno=000030 firstname=SALLY
empno=000050 firstname=JOHN
empno=000060 firstname=IRVING
empno=000070 firstname=EVA
empno=000090 firstname=EILEEN
empno=000100 firstname=THEODORE
empno=000110 firstname=VINCENZO
empno=000120 firstname=SEAN
empno=000130 firstname=DOLORES
empno=000140 firstname=HEATHER
empno=000150 firstname=BRUCE
empno=000160 firstname=ELIZABETH
empno=000170 firstname=MASATOSHI
empno=000180 firstname=MARILYN
empno=000190 firstname=JAMES
empno=000200 firstname=DAVID
empno=000210 firstname=WILLIAM
empno=000220 firstname=JENNIFER
empno=000230 firstname=JAMES
empno=000240 firstname=SALVATORE
empno=000250 firstname=DANIEL
empno=000260 firstname=SYBIL
empno=000270 firstname=MARIA
empno=000280 firstname=ETHEL
empno=000290 firstname=JOHN
empno=000300 firstname=PHILIP
empno=000310 firstname=MAUDE
empno=000320 firstname=RAMLAL
empno=000330 firstname=WING
empno=000340 firstname=JASON
```

UNIT 24

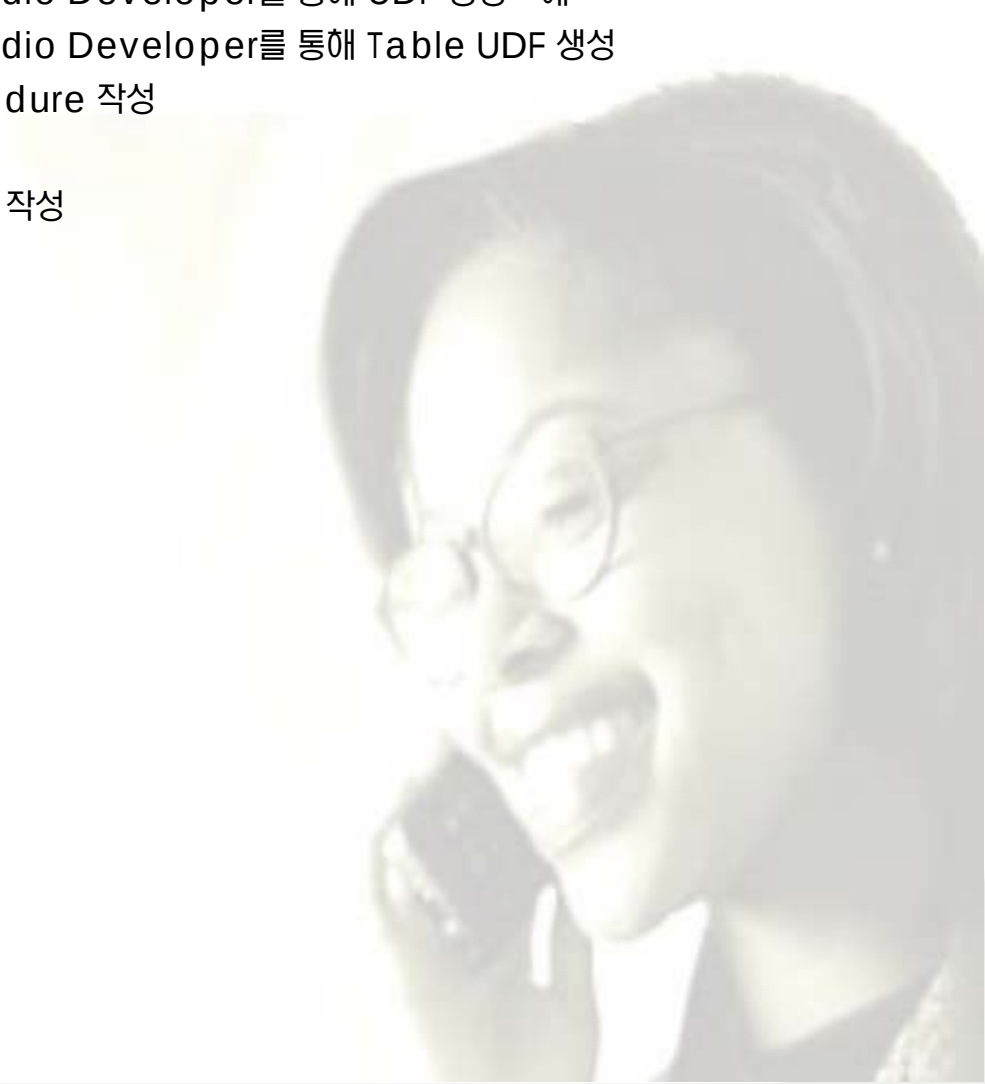
스토어드 프로그램

이 장에서는 DB2가 제공하는 Stored Function 및 Stored Procedure, Trigger 을 통한 Application Logic 작성에 대한 내용을 소개합니다.

DB2 9.7 개발자 가이드

Developer Edition

- Stored Program
- IBM Data Studio Developer를 통해 Stored Program 생성
- IBM Data Studio Developer를 통해 UDF 생성
- IBM Data Studio Developer를 통해 UDF 생성 - 예
- IBM Data Studio Developer를 통해 Table UDF 생성
- Stored Procedure 작성
- Trigger 작성
- 모듈(Module) 작성



TOPIC 03 01 Stored Program

Point

Stored Program은 UDF, Trigger, Stored Procedure등의 작성을 통하여 DBMS내에 응용프로그램 로직을 저장하고 동작하도록 하는 기능입니다.

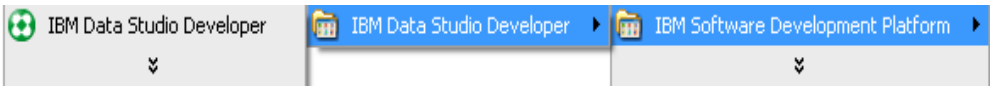
1. Stored Program은 DBMS Application Object 를 사용함으로써 응용프로그램 코드의 단순화, 성능개선, 코드의 재 사용 향상 및 이식성을 증대할 수 있습니다.

2. 일반적은 Editor를 이용하여 작성 및 생성 할수 있습니다.

   그러나 IBM Data Studio Developer를 사용하면 프로그램 디버깅이 가능하여, 작성 시 매우 편리합니다.

3. Data Studio Developer 시작 방법은 아래와 같습니다.

   Windows에서 "시작" → "프로그램" → "IBM Software Development Platform" → "IBM Data Studio Developer " → "IBM Data Studio Developer"를 수행합니다.



Tip

- Data Studio Developer는 9.7에서는 Optim Development Studio로 변경되었습니다.

TOPIC 03 02 Stored Program 생성

Point

IBM Data Studio Developer 실행하여 Stored Program을 생성하는 절차를 소개합니다.

Tip

각 프로젝트는 DB connecti on 이 필요합니다.

1 데이터베이스에 연결합니다.

IBM 에서 "연결" → "새연결" 를 실행합니다.
연결하려는 DB 이름과 호스트 및 포트번호를 그림과 기술합니다.

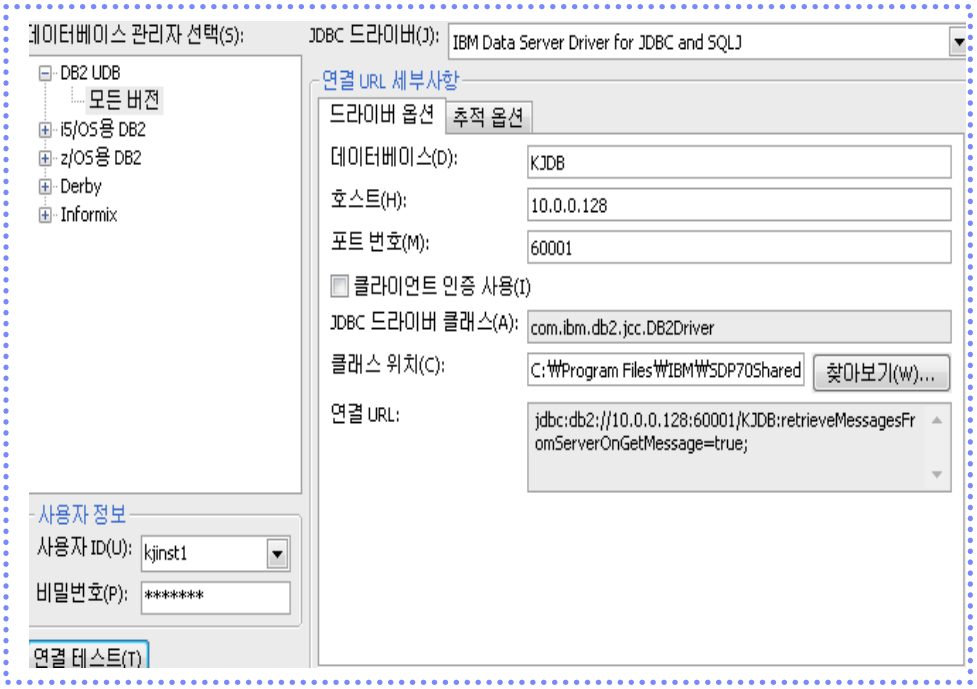


Figure 2402A ••• 데이터베이스 연결

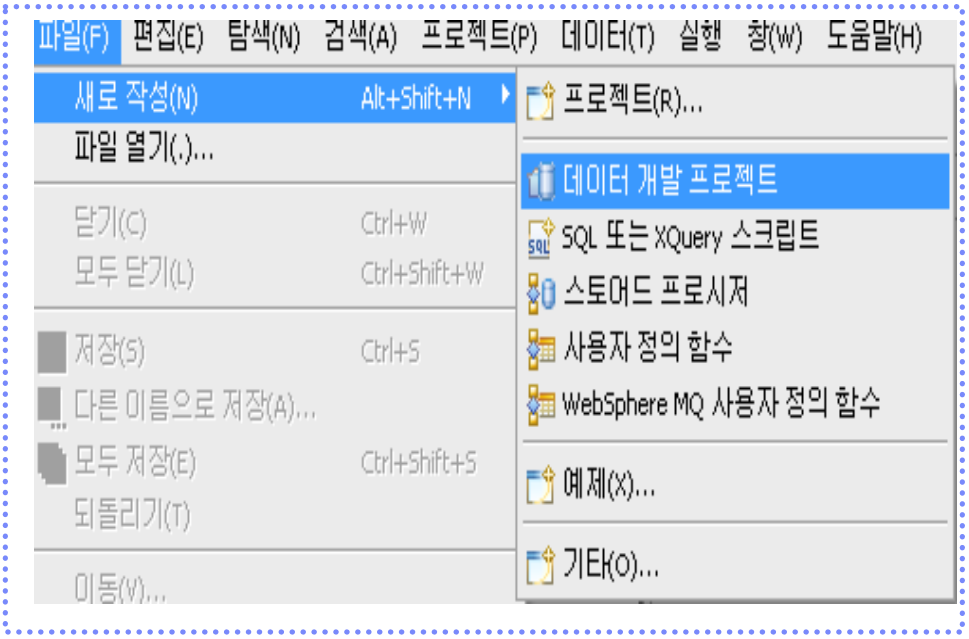


Figure 2402B ••• 데이터 개발 프로젝트를 실행

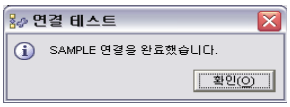
TOPIC 03 02 Stored Program 생성

Point

IBM Data Studio Developer 실행하여 Stored Program을 생성하는 절차를 소개합니다.

Tip

"연결테스트" 가 성공하면 다음과 같은 팝업이 뜹니다.



2 스토어드 프로시저 작성을 선택합니다.

데이터 개발 프로젝트에서 스토어드 프로시저에서 새로 작성을 실행합니다.
이름과 언어 및 개발하려는 상황에 맞게 설정 후 다음 버튼을 클릭합니다.

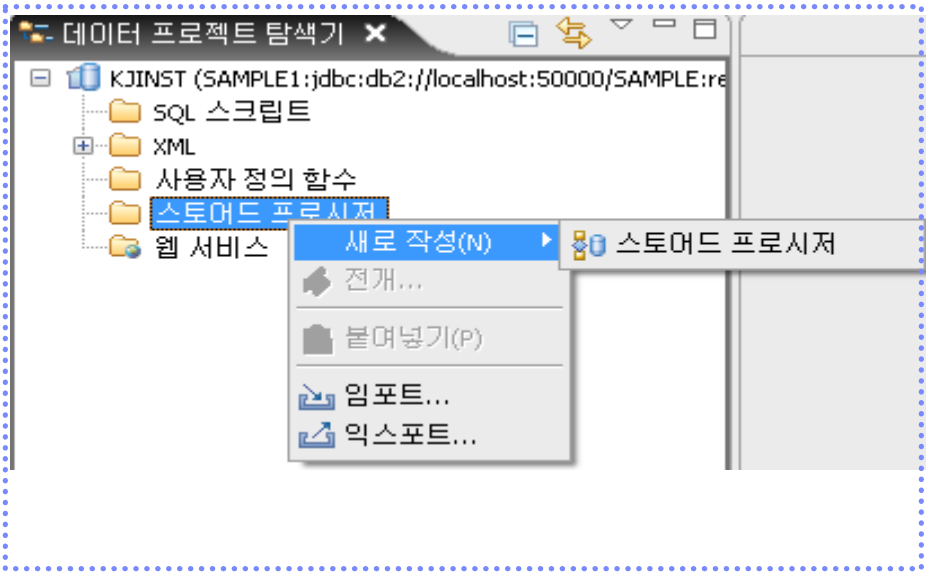


Figure 2402C ••• IBM Data Studio Developer– 스토어드 프로시저작성

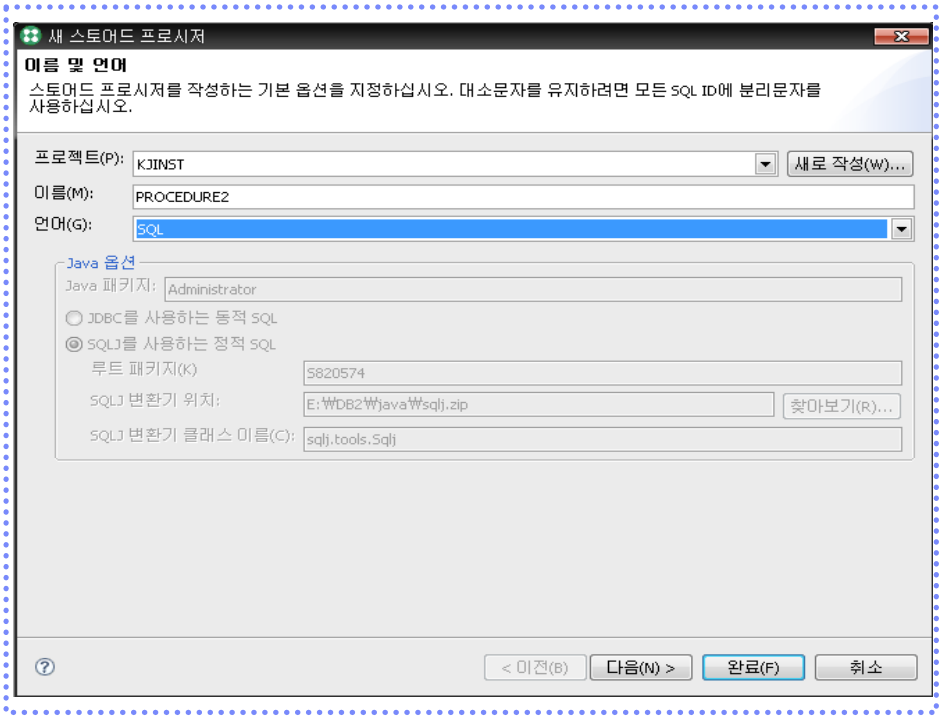


Figure 2402D ••• IBM Data Studio Developer – 스토어드 프로시저 작성

TOPIC 03 02 Stored Program 생성

Point

IBM Data Studio Developer 실행하여 Stored Program을 생성하는 절차를 소개합니다.

3 SQL 작성을 클릭하여 스토어드 프로시저를 작성합니다.

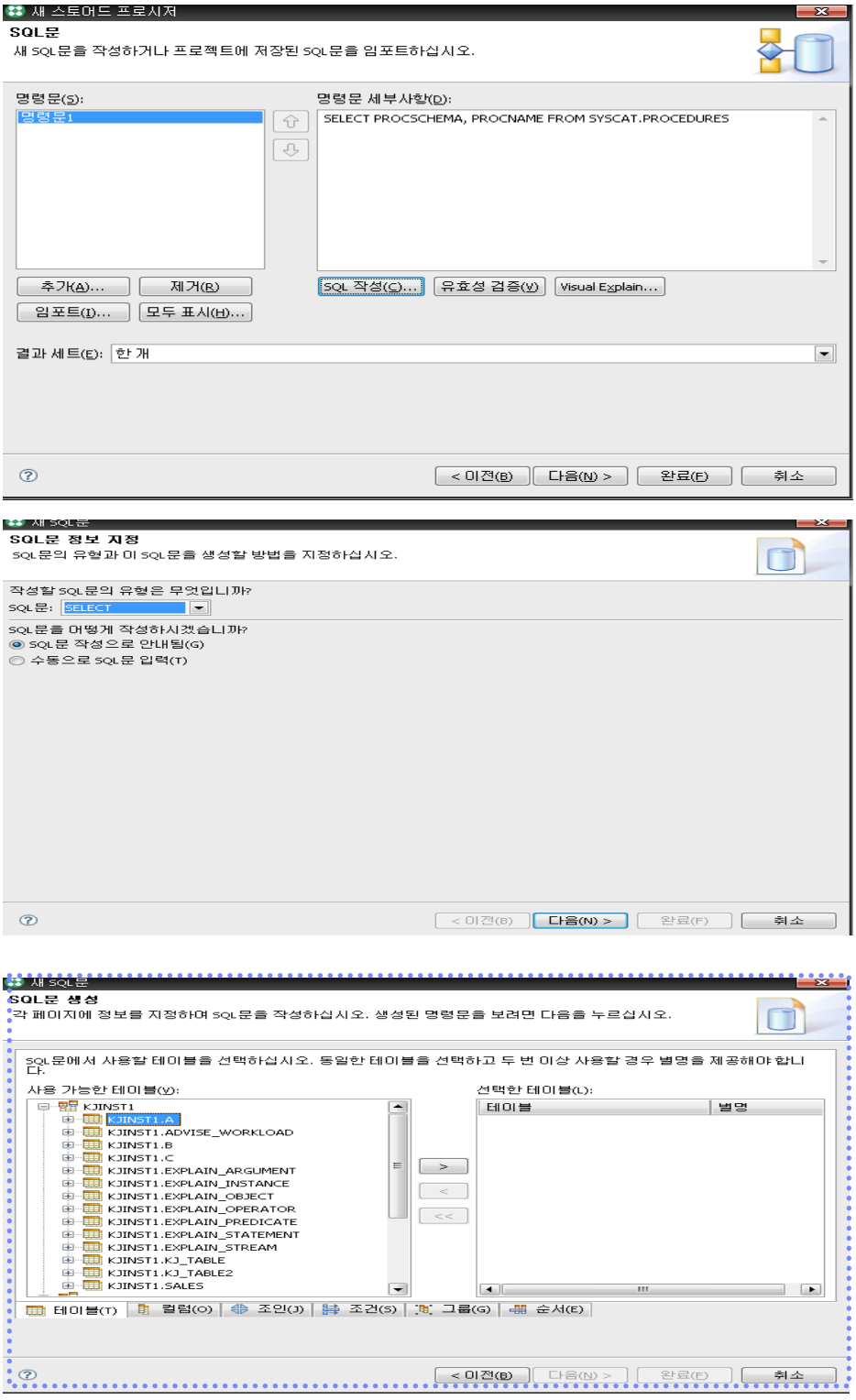


Figure 2402E IBM Data Studio Developer – 스토어드 프로시저 작성

TOPIC 03 03 UDF 생성

Point

IBM Data Studio Developer를 실행하여 User-Defined Function (UDF) 을 작성하는 방법입니다.

Tip

SQL UDF를 단순한 과정을 거쳐 생성합니다. 다음 예는 세금을 계산 하는 간단한 "TAX" function을 생성합니다.

1 UDF 생성하기

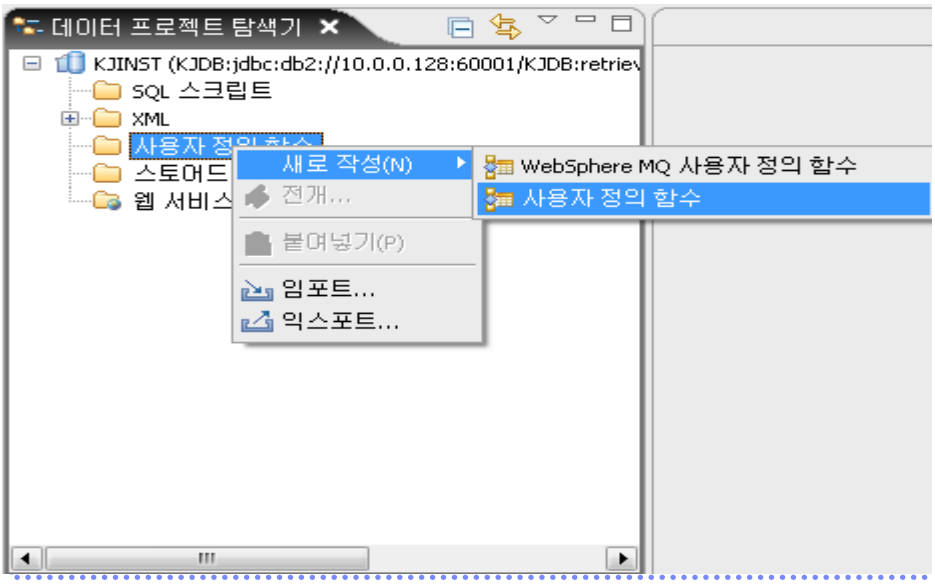
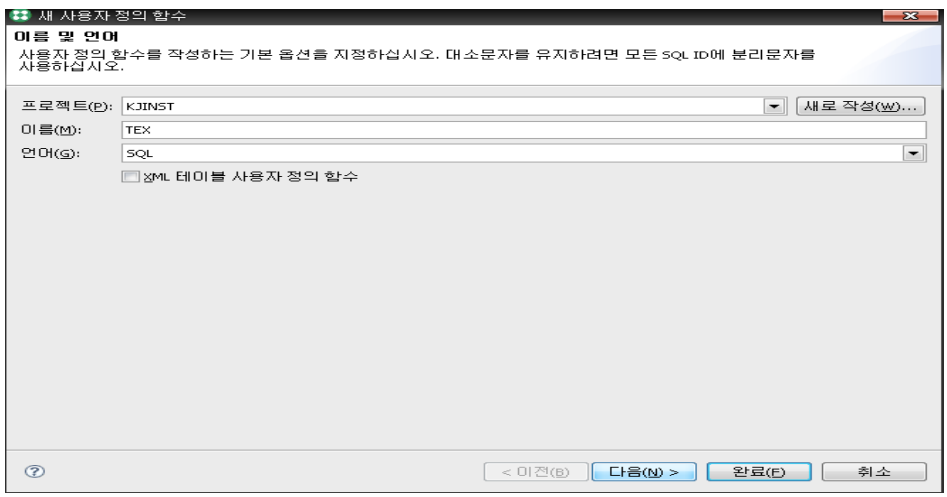


Figure 2403A IBM Data Studio Developer– UDF 새로 작성-1

➲ 다음 그림에서 완료 버튼을 클릭하고 아래와 같이 내용을 수정합니다.



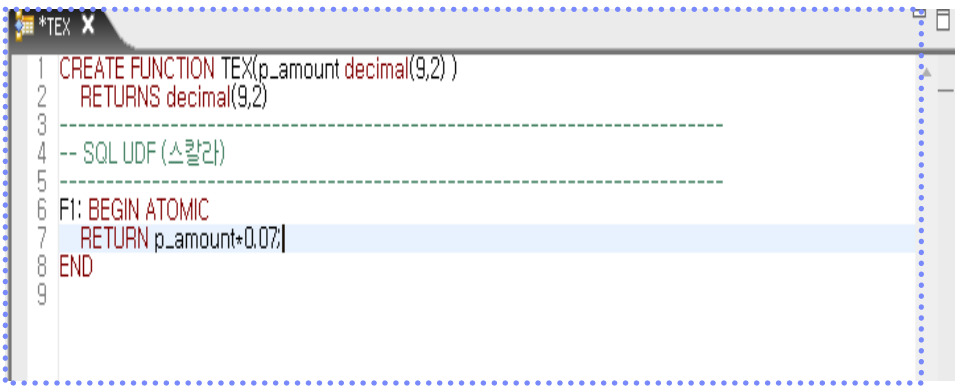


Figure 2403B IBM Data Studio Developer– UDF 새로 작성-2

TOPIC 03 03 UDF 생성

Point

IBM Data Studio Developer를 실행하여 User-Defined Function (UDF) 을 작성하는 방법입니다.

Tip

CREATE FUNCTION 다음에는 함수 이름이 반드시 기술되어야 하며, 괄호내부에 매개변수와 데이터 타입을 여러 개 지정할 수 있습니다.

2 UDF 의 필수구문 및 선택구문 구성

"CREATE FUNCTION"문장은 몇 가지 필수구문 및 선택 구문으로 구성됩니다.

예제 에서는 p_amount DECIMAL(9,2) 하나의 Parameter를 정의하였습니다. Parameter p_amount는 판매금액을 나타냅니다. Application에서 함수를 call하면, 함수는 판매금액에 대한 세금을 반환합니다. "RETURN" 구문은 DECIMAL(9,2) 반환을 기술합니다.

함수 본문은 "BEGIN ATOMIC ... END SQL" block 내부에 기술합니다.
함수는 다음 한 문장을 기술합니다.
IBM Data Studio Developer에서 "전개" 버튼을 눌러 실행합니다.

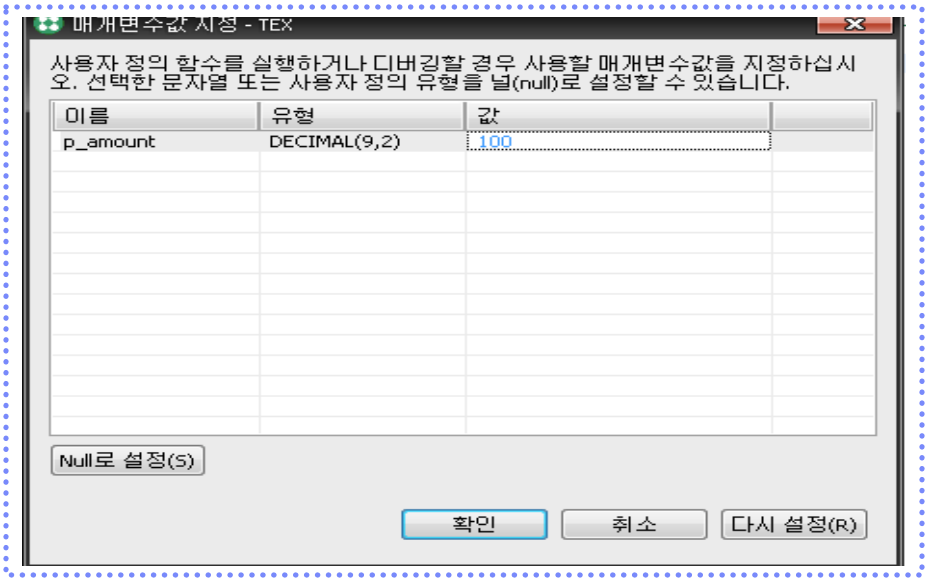


Figure 2403C IBM Data Studio Developer– 새로 작성-매개변수 지정

➲ 위와 같이. 입력변수 p_amount 에 대한 입력창이 팝업됩니다.
➲ 임의의 값을 입력하고 "확인" 버튼을 눌러 실행합니다.
➲ 함수가 성공적으로 수행되면, 다음과 같이 표시됩니다.

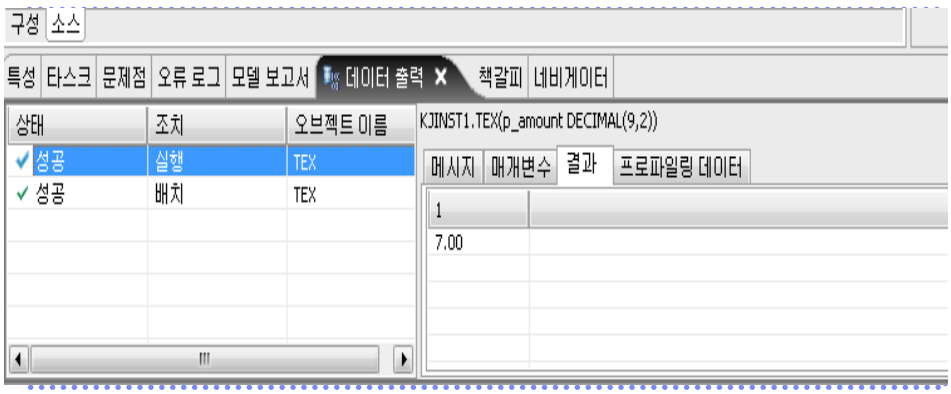


Figure 2403D IBM Data Studio Developer– UDF 새로 작성- 출력보기

TOPIC 03 03 UDF 생성

Point

IBM Data Studio Developer를 실행하여 User-Defined Function (UDF) 을 작성하는 방법입니다.

➲ 빠른 실행을 위해, 명령 창에서 다음과 같이 Query를 수행합니다.

```
SELECT   product_id, retail_price,
         KJINST1.TAX(retail_price) AS tax
   FROM KJINST1.product;
```

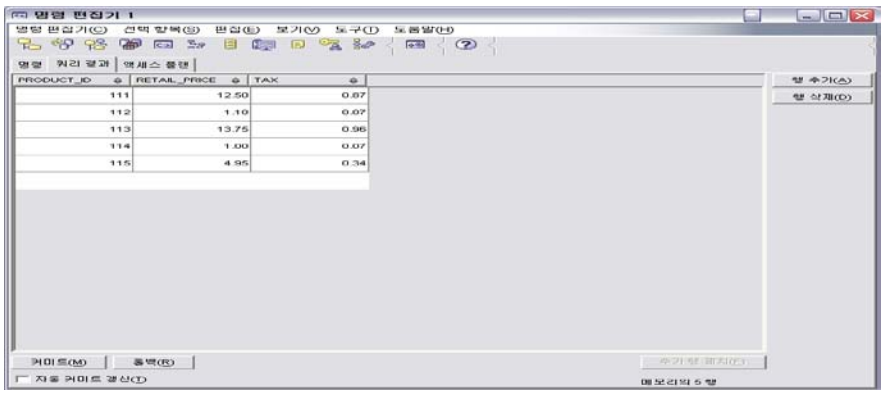


Figure 2403E IBM Data Studio Developer– UDF 새로 작성- 출력보기

Tip

TAX 함수의 Qualifier에 유의하세요.
함수 앞의 스키마를 제거하기 위해서는 "DB2 CURRENT PATH" 를 수정하세요. 이는 PATH와 같은 환경변수 역할을 합니다.

➲ DB2세션에서 함수에 대한 경로를 결정하는데 사용됩니다.
➲ 명령 창에서 다음 Query을 통해 "current path" 를 확인할 수 있습니다.

```
VALUES CURRENT PATH;
```

만일 "db2admin" 계정을 DB에 Connect하면, 다음과 같이 결과가 표시됩니다.

```
"SYSIBM","SYSFUN","SYSPROC","DB2ADMIN"
```

결과에서 보듯이, CURRENT PATH는 스키마 리스트 입니다. 뒤이어, 접속 사용자 스키마가 붙습니다.
➲ 함수이름 앞에 스키마가 정의되어 있지 않으면, CURRENT PATH에 따라 함수를 결정합니다. 스키마 순서에 따라, 실행할 함수를 검색하게 됩니다.

➲ CURRENT PATH를 변경하기 위해, "SET CURRENT PATH" 를 사용합니다.

```
SET CURRENT PATH = CURRENT PATH,"KJINST1"
VALUES CURRENT PATH

------------------------------------------------------
"SYSIBM","SYSFUN","SYSPROC","DB2ADMIN","KJINST1"

SET CURRENT PATH = SYSTEM PATH,"KJINST1"                    1
VALUES CURRENT PATH

------------------------------------------------------
"SYSIBM","SYSFUN","SYSPROC","KJINST1"                       2
```

TOPIC 03 03 UDF 생성

Point

IBM Data Studio Developer를 실행하여 User-Defined Function (UDF) 을 작성하는 방법입니다.

Tip

디폴트 SQL 경로(또는 사용자 스키마 앞에 SYSIBM이 있는 SQL 경로)를 사용하고 스키마에 새 SYSIBM 함수와 이름이 같은 기존 함수가 있는 경우, SYSIBM 함수가 대신 사용됩니다. 이 상황은 일반적으로 성능을 향상시키지만 예기치 않은 동작의 원인이 될 수 있습니다.

```
SET CURRENT PATH = USER,"KJINST1"
VALUES CURRENT PATH

------------------------------------------------------------
"DB2ADMIN","KJINST1"                                          3

SET CURRENT PATH =
"DB2ADMIN","SYSIBM","SYSFUN","SYSPROC","KJINST1"
VALUES CURRENT PATH

------------------------------------------------------------  4
"DB2ADMIN","SYSIBM","SYSFUN","SYSPROC","KJINST1"
```

➲ CURRENT PATH에 스키마 “KJINST1”를 추가합니다.
➲ CURRENT PATH를 SYSTEM PATH와 “KJINST1”로 변경합니다.
➲ 현재의 USER ID와 “KJINST1”를 포함하여 CURRENT PATH를 변경합니다.
➲ CURRENT PATH에 모든 SCHMA를 기술하여 적용합니다.

TOPIC 03 04 UDF 생성 - 예

Point

IBM Data Studio Developer 를 실행하여 UDF 를 작성하는 예제입니다.

1 복잡한 Query 실행하기

다음은 특정 기간 동안에 판매금액에 대한 이익률을 계산하는 UDF를 작성합니다.
해당 정보가 다음과 같이 세 개의 Table에 분산 되어 있습니다.

1) PRODUCT_PURCHASE : 각 제품별 판매 가격 정보
2) PRODUCT : 제품별 비용
3) SALES : 판매에 대한 일자/시간 관리 정보

특정 기간 동안의 이익률을 계산 하려면, 매번 세 개의 Table을 Join하여야 합니다.
이를 단순화 하기 위하여, PRODUCT ID, 시작일자 및 종료일자를 입력 파라미터를 갖는 UDF를 작성합니다.

➲ Table Layout

```
CREATE TABLE KJINST1.PRODUCT (
      PRODUCT_ID        INT         NOT NULL,
      DESCRIPTION       VARCHAR(40)  NOT NULL,
      COST              DECIMAL(7,2) NOT NULL,
      RETAIL_PRICE      DECIMAL(7,2) NOT NULL,
      INVENTORY         INT         NOT NULL,
      MINIMUM_INVENTORY  INT         NOT NULL
                        WITH DEFAULT 0,
      PRIMARY KEY (PRODUCT_ID) );

CREATE TABLE KJINST1.CUSTOMER (
     CUSTOMER_ID        INT         NOT NULL
       GENERATED ALWAYS AS IDENTITY
       (START WITH 0 INCREMENT BY 1),
     CREDIT_CARD         CHAR(16),
     EXPIRY_DATE         CHAR(4),
     LASTNAME            VARCHAR(28),
     FIRSTNAME           VARCHAR(28),
     ADDRESS             VARCHAR(300),
     ZIP_CODE            CHAR(6),
     PHONE               CHAR(10),
     PRIMARY KEY (CUSTOMER_ID) );

CREATE TABLE KJINST1.SALES (
     SALES_TRANSACTION_ID  INT
       GENERATED ALWAYS AS IDENTITY
       (START WITH 0 INCREMENT BY 1),
     CUSTOMER_ID           INT
       REFERENCES KJINST1.CUSTOMER(CUSTOMER_ID),
     SUB_TOTAL             DECIMAL(7,2),
     TAX                   DECIMAL(7,2),
     TYPE                  INT NOT NULL,
     TRANSACTION_TIMESTAMP TIMESTAMP,
     PRIMARY KEY (SALES_TRANSACTION_ID) );
```

TOPIC 03 04 UDF 생성 - 예

Point

IBM Data Studio Developer 를 실행하여 UDF 를 작성하는 예제입니다.

```
CREATE TABLE KJINST1.PRODUCT_PURCHASES (
    SALES_TRANSACTION_ID  INT
      REFERENCES KJINST1.SALES(SALES_TRANSACTION_ID)
            ON DELETE CASCADE,
    PRODUCT_ID          INT
      REFERENCES KJINST1.PRODUCT(PRODUCT_ID),
    PRICE               DECIMAL(7,2) NOT NULL,
    QTY                 INT NOT NULL );

CREATE TABLE KJINST1.AUDIT_STOCKJINST1HK (
    STAFF               VARCHAR(50),
    CHECKTIME           TIMESTAMP );
```

➲ PROD_PROFIT UDF 생성하기

```
CREATE FUNCTION KJINST1.PROD_PROFIT                                  1
  ( p_pid INTEGER, p_sdate DATE, p_edate DATE )                      2
  RETURNS DECIMAL(9,2)
----------------------                                               3
-- SQL UDF (Scalar)
----------------------
F1: BEGIN ATOMIC
  DECLARE v_retail_price DECIMAL(9,2);                               4
  DECLARE v_cost DECIMAL(9,2);                                       5
  DECLARE v_err VARCHAR(70);

  SET (v_retail_price, v_cost) =                                     6
    ( SELECT SUM(retail_price)
           , SUM(cost)
       FROM KJINST1.product p
         , KJINST1.product_purchases pp
         , KJINST1.sales s
      WHERE p.product_id = pp.product_id
        AND pp.sales_transaction_id
          = s.sales_transaction_id
        AND p.product_id = p_pid
        AND DATE(s.transaction_timestamp)
          BETWEEN p_sdate AND p_edate );                             7
                    SET v_err = 'Error: product ID '                 8
                              || CHAR(p_pid)
                             || ' was not found.';

                    IF ( v_retail_price IS NULL                      9
                      OR v_cost         IS NULL )
                          THEN
                      SIGNAL SQLSTATE '80000'
                      SET MESSAGE_TEXT = v_err;
                          END IF;

                                                                     10
                          RETURN
                ( v_retail_price - v_cost ) / v_cost * 100;
                          END                                        11
```

TOPIC 03 04 UDF 생성 - 예

Point

IBM Data Studio Developer 를 실행하여 UDF 를 작성하는 예제입니다.

1) UDF KJINST1.PROD_PROFIT를 생성합니다.
2) PROD_PROFIT에 세개의 INPUT Parameter를 지정합니다.
   - p_pid : Product ID
   - p_sdate : 시작일자 조건
   - p_edate : 종료일자 조건
3) DECIMAL(9,2)형태의 이익률을 반환합니다.
4) UDF "KJINST1.TAX"는 단문으로 이루어진 함수 입니다. PROD_PROFIT처럼 복문으로 구성된 UDF는 BEGIN ATOMIC(4) 와 END(11)사이에 함수 Logic을 기술 합니다.
5) 함수 내에서 사용될 변수를 기술합니다.
   - v_retail_price : 판매 금액
   - v_cost : 판매 비용
   - v_err : PRODUCT ID가 없을 경우의 Error Message
6) PRODUCT, PRODUCT_PURCHASES와 SALES를 Join하여 판매금액과 비용을 산정한다. "SET"문장을 통해 값을 변수에 Assign한다.
7) SALES Table의 transaction_timestamp가 TIMESTAMP형태로 저장되어 있습니다. 이를 p_sdate, p_edate와 비교하기 위하여 DATE()함수를 사용하여 DATE 값으로 변환한다.
8) PRODUCT ID가 없을 경우 Error Message를 생성합니다.
9) 만일, PRODUCT ID가 없을 경우, UDF는 Error를 발생한다. 이후, Function 수행을 종료하고, 호출한 Application에 Error를 반환한다. v_err를 VARCHAR(70)으로 정의하였습니다. "SIGNAL SQLSTATE"의 error text의 한계가 70자 입니다. 만일, Message가 한계를 초과하면, 경고 없이 절삭됩니다.
10) 판매금액과 비용을 통해 이익률을 계산하여 반환합니다.

IBM Data Studio Developer를 통해 빌드 후, 실행합니다.

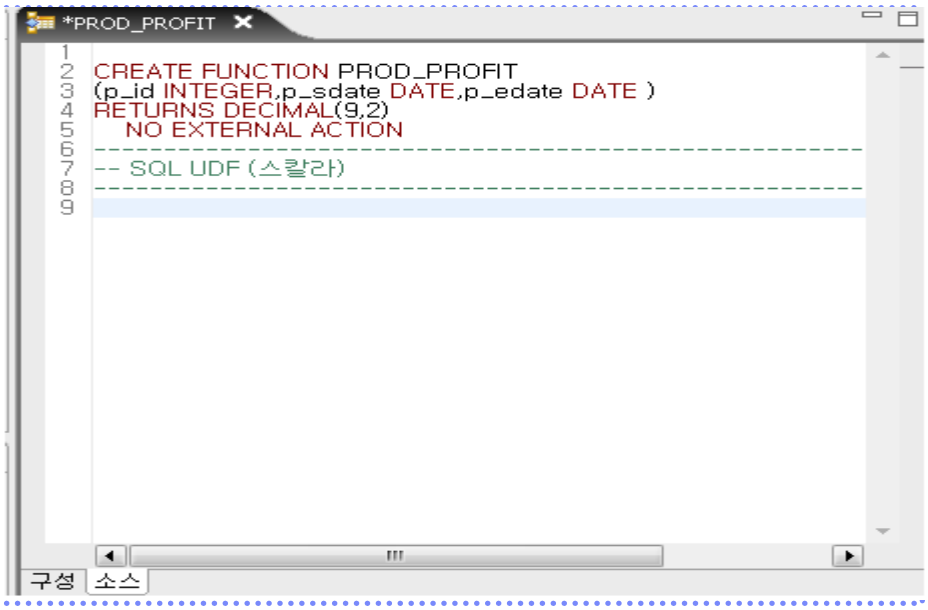


Figure 2404A••• IBM Data Studio Developer– UDF 실행하기 -1

TOPIC 03 04 UDF 생성 - 예

Point

IBM Data Studio Developer 를 실행하여 UDF 를 작성하는 예제입니다.

Tip

SP를 명령 창에서 실행하고자 할 때는 아래와 같은 방법으로 사용합니다.

>db2
"call KJINST1.PROD_PROFIT (111, 2009-07-01, 2009-07-31)

➲ PROD_PROFIT를 실행

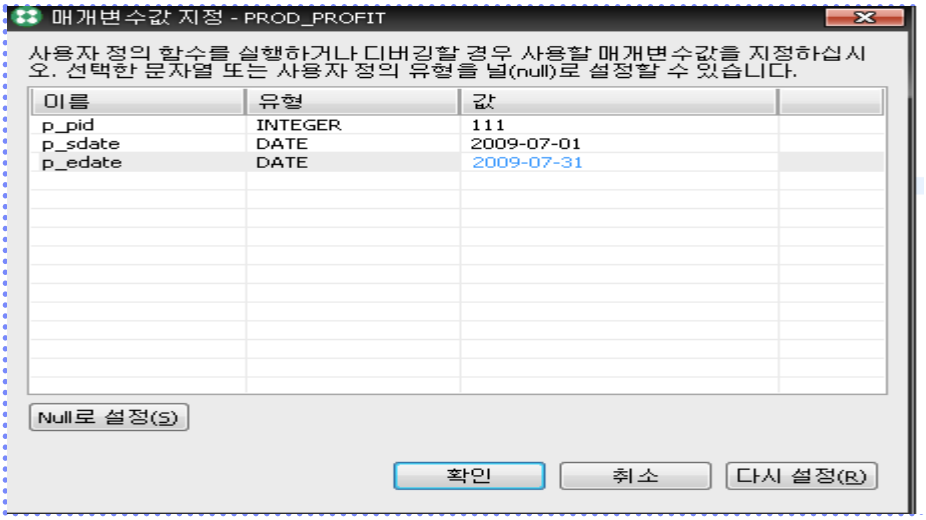


Figure 2404B ••• IBM Data Studio Developer– UDF 실행하기 -2

➲ 실행결과 화면 & 잘못된 Product ID를 입력했을 때의 화면

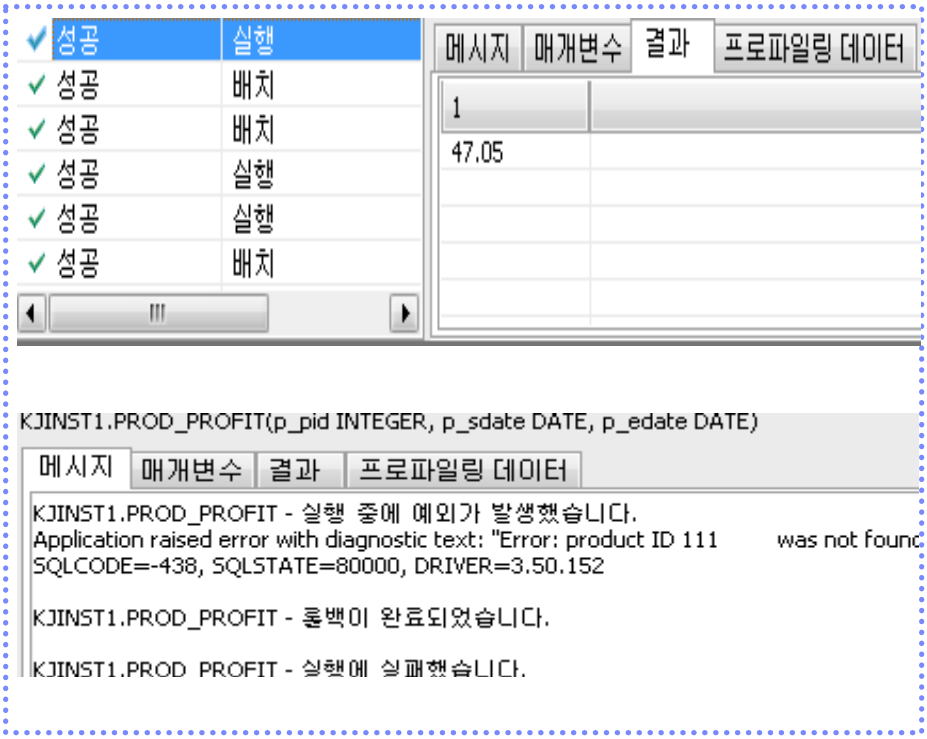


Figure 2404C ••• IBM Data Studio Developer– UDF 실행하기 -2

TOPIC 03 05 Table UDF 작성

Point

IBM Data Studio Developer 를 실행하여 Table UDF 를 작성하는 예제입니다.

1 Table UDF

Query의 FROM절에서 사용되며, Table형xo의 Row를 반환합니다. 다음은 입고가 필요한 모든 PRODUCT를 반환하는 Table 함수입니다. 누가, 언제 STOCK를 점검했는지 LOG을 위해 AUDIT_STOCKCHK Table을 사용합니다.

➲ STOCKCHK UDF 생성하기

```
CREATE FUNCTION KJINST1..STOCKCHK()                          1

   RETURNS TABLE ( PRODUCT_ID       INTEGER                  2
                 , DESCRIPTION      VARCHAR(40)
                 , INVENTORY        INTEGER
                 , MINIMUM_INVENTORY INTEGER )

   MODIFIES SQL DATA                                         3
-----------------------------------------------
-- SQL UDF (TABLE)
-----------------------------------------------
F1: BEGIN ATOMIC

   INSERT INTO KJINST1.AUDIT_STOCKCHK                        4
   VALUES (USER, CURRENT TIMESTAMP);

   RETURN
      SELECT PRODUCT_ID
           , DESCRIPTION
           , INVENTORY
           , MINIMUM_INVENTORY
        FROM KJINST1.PRODUCT
       WHERE INVENTORY < MINIMUM_INVENTORY;

END
```

1) UDF KJINST1.STOCCHK를 입력 Parameter없이 생성합니다.
2) RETURN절에 Table function이 반환할 Column과 Type을 정의합니다.
3) Table function은 기본적으로 실행 위주로 수행됩니다. “MODIFIES SQL DATA”를 기술하여, Function내에서 INSERT, UPDATE 및 DLETE를 할 수 있도록 합니다.
4) 누가 Call을 했는지를 관리하기 위해, AUDIT_STOCKCHK Table에 데이터를 입력합니다.
   - Special Register
   - USER : DB에 접속한 current user ID
   - CURRENT TIMESTAMP : 현재 시간

TOPIC 03 05 Table UDF 작성

Point

IBM Data Studio Developer 를 실행하여 Table UDF 를 작성하는 예제입니다.

```
1  CREATE FUNCTION KJINST1.STOCKCHK( )
2     RETURNS TABLE( product_id INTEGER
3                  , descriiption VARCHAR(40)
4                  , inventory INTEGER
5                  , minimum_inventory INTEGER)
6      MODIFIES SQL DATA
7     NO EXTERNAL ACTION
8  ------------------------------------------------
9  -- SQL UDF (스칼라)
10 ------------------------------------------------
11 F1: BEGIN ATOMIC
12    INSERT INTO KJINST1.AUDIT_STOCKCHK VALUES(USER, CU
13    RETURN
14         SELECT product_id, description,inventory,minimum_inve
15         FROM KJINTS1.PRODUCT
16         WHERE inventory < minimum_inventory;
17 END
18
```

Figure 2405A • • • IBM Data Studio Developer– Table UDF

➲ STOCKCHK UDF 수행

```
SELECT * FROM TABLE (KJINST1.STOCKCHK())  AS STOCKCHK;
```

STOCKCHK.sql

메시지 | 매개변수 | 결과 | 프로파일링 데이터

PRODUCT_ID	DESCRIIPT...	INVENTORY	MINIMUM_I...

Figure 2405B • • • IBM Data Studio Developer– Table UDF

TOPIC 03 06 Stored Procedure 작성

Point

IBM Data Studio Developer 를 실행하여 Stored Procedure를 생성하는 예입니다.

1 Stored Procedure 이해하기

SP는 DB내의 데이터를 Access하거나 수정하는 하나 이상의 SQL문장으로 구성된 DB Object입니다. SP는 DB2의 제어 하에 수행되고 관리 됩니다. SP는 SQL PL, C/C++, Java, COBOL, CLR 및 OLE를 사용하여 작성됩니다. SQL Procedure가 간단하기 때문에 주로 사용됩니다.

➲ SQL Stored Procedure의 장점

- ➢ Code의 재 사용을 통한 Business Logic 통합
- ➢ 보안 Level 강화
- ➢ 성능 개선

➲ SP는 DB에 저장되며, SQL 문장과 Business Logic을 encapsulate합니다. 적절한 권한을 가진 모든 Application Client에서 SP를 호출할 수 있습니다. 또한 Code의 재 사용률을 증대합니다. 또한, SP내의 Business Logic 변경이 관련된 모든 Application또는 Client와 독립적이므로 관리 비용을 절감할 수 있습니다.

➲ 사용자는 SP를 통해 Access하는 Table 또는 View에 대해 권한이 필요치 않습니다. 다만 SP를 호출할 수 있는 권한만 있으면 됩니다. 이를 통해, 사용자의 예상치 않은 접근에 대한 통제를 할 수 있습니다.

SP는 DBMS내에 SQL과 Business Logic을 가지고 수행합니다. 즉, Application과 DB 사이의 N/W Traffic을 줄일 수 있습니다. 또한 성능을 위해 SQL이 컴파일 됩니다.

2 Stored Procedure 개발하기

Tip

Table에 Column을 추가 하기 위해, “ALTER TABLE” 문장을 사용하거나 제어센터 (Control Center)를 사용할 수 있습니다.

다음은 SP를 통해 할 수 있는 몇 가지 예를 소개합니다.
이를 위해 DB내의 table을 변경합니다. 예제에서는, Internet On-Line 주문을위한 Web site를 SP를 소개합니다.
On-Line 주문을 위해, SALES 와 PRODUCT_PURCHASES table을 POS (Point-of-Sale) 시스템과 같이 변경합니다. 주문 상태를 관리하기 위해, Column을 추가 합니다.

SALES Table의 ORDER_STATUS column :
Column 에 Check Constraints (N, C, P & I)를 추가합니다.
- N : New Order
- C : 주문 완료
- P : 부분적 주문 완료
- I : 고객정보 부족

PRODCUT_PURCHASES Table의 STATUS column :
Column에 Check Constraints (N, C & O)를 추가합니다.
- N : New Order
- C : 주문 완료
- O : 재고 부족

TOPIC 03 06 Stored Procedure 작성

Point

IBM Data Studio Developer 를 실행하여 Stored Procedure를 생성하는 예입니다.

➲ Stored procedure (SP) 생성

다음은 새로운 계약에 관해 정보를 조회하는 SP를 IBM Data Studio Developer를 통해 생성하는 과정을 보여줍니다.

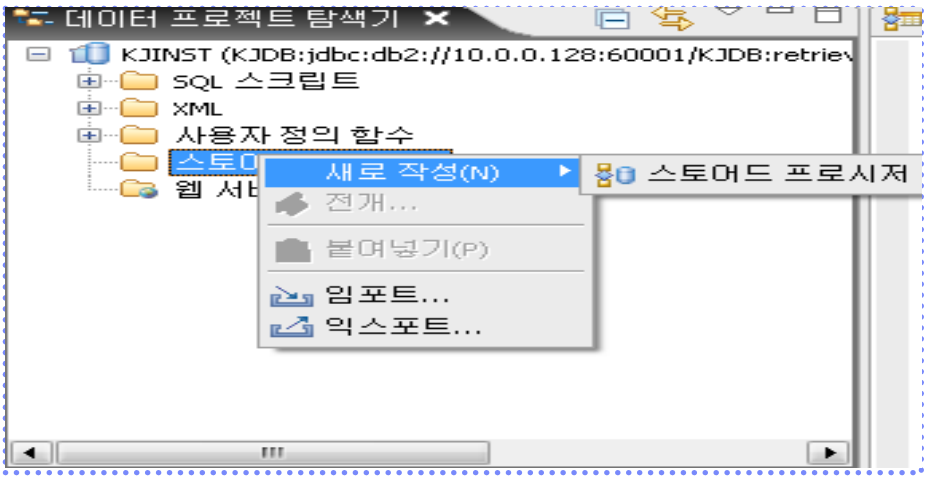


Figure 2406A IBM Data Studio Developer 에서 SP구현하기 -1

Procedure PROCEDD_NEWORDER를 Parameter없이 정의 합니다.
Procedure 본문은 "BEGIN"과 "END" 블록 사이에 기술합니다. 다음의 Query 결과를 처리하기 위해 FOR loop를 정의합니다. FOR Loop내에, 새로운 주문에 대한 처리를 위해 Business Logic을 기술합니다.

```
SELECT SALES_TRANSACTION_ID, CUSTOMER_ID
  FROM SALES
 WHERE ORDER_STATUS = 'N'
```

➲ Stored Procedure PROCESS_NEW 생성

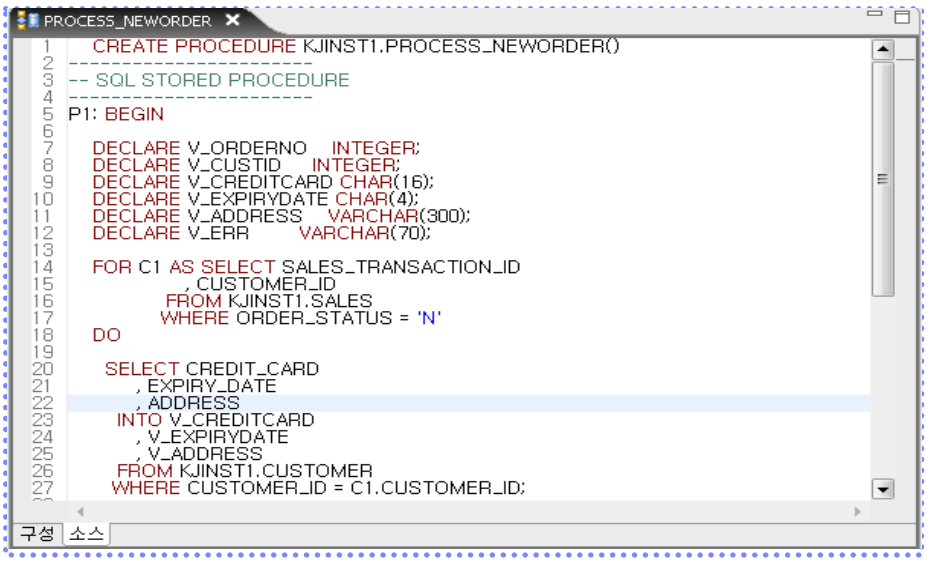


Figure 2406B IBM Data Studio Developer 에서 SP구현하기 -2

IBM Data Studio Developer 에서 PROCESS_NEW를 Build합니다.

TOPIC 03 06 Stored Procedure 작성

Point

IBM Data Studio Developer 를 실행하여 Stored Procedure를 생성하는 예입니다.

이제까지, 단순 SP를 생성하는 과정을 살펴 보았습니다.
예제에서는 단순한 SP를 작성하였습니다. 다음은 stored procedure의 강력한 기능을 통해 새로운 SP를 전개합니다.
PROCESS_NEWORDER에서 Web application에 입력한 새로운 주문을 처리합니다.
이제까지는 새로운 주문을 검색하는 Logic을 작성하였습니다. FOR loop내의 각 주문에 다음과 같은 처리를 합니다.

- 제품을 주문한 고객 정보를 점검합니다.
- 만일 고객의 주소나 신용카드 정도가 불충분하면 처리하지 않습니다.
- 검증된 주문은 처리를 계속합니다.

➲ 수정된 KJINST1.PROCESS_NEWORDER

```
CREATE PROCEDURE KJINST1.PROCESS_NEWORDER()
-----------------------
-- SQL STORED PROCEDURE
-----------------------
P1: BEGIN

    DECLARE V_ORDERNO    INTEGER;
    DECLARE V_CUSTID     INTEGER;
    DECLARE V_CREDITCARD CHAR(16);
    DECLARE V_EXPIRYDATE CHAR(4);
    DECLARE V_ADDRESS    VARCHAR(300);
    DECLARE V_ERR        VARCHAR(70);

    FOR C1 AS SELECT SALES_TRANSACTION_ID                          (1)
                   , CUSTOMER_ID
                FROM KJINST1.SALES
               WHERE ORDER_STATUS = 'N'
    DO

      SELECT CREDIT_CARD                                           (2)
           , EXPIRY_DATE
           , ADDRESS
        INTO V_CREDITCARD
           , V_EXPIRYDATE
           , V_ADDRESS
        FROM KJINST1.CUSTOMER
       WHERE CUSTOMER_ID = C1.CUSTOMER_ID;

      IF    (V_CREDITCARD IS NULL AND V_EXPIRYDATE IS NULL)        (3)
         OR (V_ADDRESS IS NULL)
      THEN

          SET V_ERR = 'THE CUSTOMER INFORMATION IS'
                    || ' NOT COMPLETE TO PROCESS THE ORDER NO '
                    || CHAR(V_ORDERNO);

          SIGNAL SQLSTATE'80000' SET MESSAGE_TEXT = V_ERR;         (4)

      END IF;

      --CALL ANOTHER SP TO PROCESS THE VALID ORDERS                (5)
      SET V_ORDERNO = C1.SALES_TRANSACTION_ID;
      CALL KJINST1.FILLORDER(V_ORDERNO, V_CUSTID);

    END FOR;

END P1
```

TOPIC 03 06 Stored Procedure 작성

Point

IBM Data Studio Developer 를 실행하여 Stored Procedure를 생성하는 예입니다.

1) Query 결과를 처리하기 위한 FOR Loop 정의
2) 고객 정보에 해당하는 카드 정보 및 주소 확인 하는 Process
3) 고객정보의 정합성 점검
4) 필요한 정보가 부적하면 Error 처리 : SQLSTATE 80000
5) 주문 번호 및 고객정보를 입력변수로 SP FILLORDER를 CALL힙니다.

➲ 수정된 KJINST1.PROCESS_NEWORDER

```
CREATE PROCEDURE KJINST1.FILLORDER
            ( IN V_SALESTXNID INTEGER
            , IN V_CUSTID INTEGER )

-----------------------
-- SQL STORED PROCEDURE
-----------------------

P1: BEGIN

   -- DECLARE VARIABLES
   DECLARE V_PRODID         INTEGER;
   DECLARE V_QTY            INTEGER;
   DECLARE V_INVENTORY      INTEGER;
   DECLARE V_PRICE          DECIMAL(5,2);
   DECLARE V_TOTALCOUNT     INTEGER DEFAULT 0;
   DECLARE V_COUNT          INTEGER DEFAULT 0;
   DECLARE V_LASTPRODUCT    INTEGER DEFAULT 0;
   DECLARE V_ERR            VARCHAR(70);

   -- DECLARE CURSORS
   DECLARE C1 CURSOR WITH HOLD FOR
      SELECT PP.PRODUCT_ID
          , RETAIL_PRICE, QTY
        FROM KJINST1.PRODUCT_PURCHASES PP
          , KJINST1.PRODUCT P
       WHERE PP.PRODUCT_ID      =P.PRODUCT_ID
         AND SALES_TRANSACTION_ID=V_SALESTXNID;

   -- DECLARE EXCEPTION HANDLER
   DECLARE CONTINUE HANDLER FOR NOT FOUND
      SET V_LASTPRODUCT = 1;
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
      RESIGNAL;
```

Stored Procedure 작성

Point

IBM Data Studio Developer 를 실행하여 Stored Procedure를 생성하는 예입니다.

```
IF V_SALESTXNID IS NULL THEN

   --CREATION A NEW SALES TXN
   INSERT INTO KJINST1.SALES
            ( SALES_TRANSACTION_ID
            , CUSTOMER_ID
            , SUB_TOTAL
            , TYPE
            , TRANSACTION_TIMESTAMP)
        VALUES ( DEFAULT
            , V_CUSTID
            , 0
            , 5
            , CURRENT TIMESTAMP);

   VALUES IDENTITY_VAL_LOCAL()
    INTO V_SALESTXNID;

END IF;

OPEN C1;
FETCH C1 INTO V_PRODID, V_PRICE, V_QTY;

WHILE (V_LASTPRODUCT = 0) DO

   BEGIN

      DECLARE C_NO_STOCK CONDITION
        FOR SQLSTATE '80000';

          DECLARE EXIT HANDLER FOR C_NO_STOCK
                                  BEGIN
                UPDATE KJINST1.PRODUCT_PURCHASES
                              SET STATUS = 'O'
                    WHERE SALES_TRANSACTION_ID
                              = V_SALESTXNID
                            AND PRODUCT_ID
                              = V_PRODID;
                                  END;
SET V_TOTALCOUNT = V_TOTALCOUNT + 1;
       SELECT INVENTORY
         INTO V_INVENTORY
         FROM PRODUCT
        WHERE PRODUCT_ID = V_PRODID;
```

TOPIC 03 06 Stored Procedure 작성

Point

IBM Data Studio Developer 를 실행하여 Stored Procedure를 생성하는 예입니다.

```
        --CHECK IF INVENTORY SATISFY THE QTY
        IF V_INVENTORY >= V_QTY THEN
          BEGIN ATOMIC
            INSERT INTO KJINST1.PRODUCT_PURCHASES
                  (SALES_TRANSACTION_ID
                  , PRODUCT_ID
                  , PRICE
                  , QTY
                  , STATUS)
            VALUES (V_SALESTXNID
                  , V_PRODID
                  , V_PRICE
                  , V_QTY
                  , 'C');

            UPDATE KJINST1.PRODUCT
              SET INVENTORY=INVENTORY-V_QTY
              WHERE PRODUCT_ID = V_PRODID;

            SET V_COUNT = V_COUNT + 1;
          END;
      ELSE
          --NOT ENOUGH STOCK TO FILL ORDER
          SET V_ERR = 'THERE IS NOT ENOUGH '
                || 'STOCK FOR PRODUCT ID  '
                || CHAR(V_PRODID)
                ||' TO FILL THE ORDER '
                || CHAR(V_SALESTXNID);
          SIGNAL SQLSTATE '80000'
            SET MESSAGE_TEXT = V_ERR;
        END IF;

      END;

      FETCH C1 INTO V_PRODID, V_PRICE, V_QTY;

    END WHILE;

    IF V_COUNT = V_TOTALCOUNT THEN
      UPDATE KJINST1.SALES
        SET ORDER_STATUS = 'C'
       WHERE SALES_TRANSACTION_ID
          = V_SALESTXNID;
    ELSE
       UPDATE KJINST1.SALES
         SET ORDER_STATUS = 'P'
        WHERE SALES_TRANSACTION_ID
           = V_SALESTXNID;
    END IF;

    UPDATE KJINST1.SALES
      SET SUB_TOTAL
        = (SELECT SUM(PRICE)
           FROM PRODUCT_PURCHASES
           WHERE SALES_TRANSACTION_ID
             = V_SALESTXNID)
     WHERE SALES_TRANSACTION_ID = V_SALESTXNID;

  END P1
```

TOPIC 03 07 Trigger 작성

Point

Trigger 에 대해 알아봅니다.

1 Trigger 이해하기

➲ Trigger는 Table에 INSERT/UPDATE/DELETE 처리 전□ 후에 필요한 일련의 작업을 자동으로 수행하기 위한 Table과 관련 있는 Database Object입니다.
➲ Trigger를 발생 시키는 문장을 "Triggering SQL" 문장 이라 합니다. Trigger 를 Triggering SQL 문장 전 또는 후에 실행 하게 할 것인지를 선택할 수 있습니다.

➲ 세 가지 Trigger Type

Trigger Type	설명
BEFORE	Table의 Data가 Triggering SQL문장에 의해 변경되기 전에 수행됩니다.
AFTER	Triggering SQL문장이 성공적으로 완료되면 Trigger가 수행됩니다. Trigger에 따라, AFTER Trigger는 다른 Trigger를 수행할 수 있습니다. DB2는 최대 16 Level까지 다른 Trigger를 연쇄적으로 수행할 수 있습니다.
INSTRADE OF	Trigger가 View를 대상으로 정의됩니다. SQL 문장이 복잡한 View에 대해 INSERT/UPDATE/DELETE를 할 때 유용합니다. View에 대해 허용되지 않는 INSERT/UPDATE/DELETE에 대해 사용됩니다. View의 Column은 해당 Table에 Column과 자동으로 Mapping되지 않으므로 INSERT/UPDATE/DELETE할 수 없다. Business Logic과 해당 Table과 View의 column에 대한 Mapping정보를 알고 있다면, SQL의 제약 사항을 우회적으로 INSTEAD Trigger 본문에 해당 Business Logic에 정의할 수 있습니다. INSTEAD OF Trigger에 SQL문장을 정의하여 Application Interface를 단순화 할 수 있습니다.

Trigger는 Application 전반에 걸쳐 수행되어야 될 Business Rule을 항상 수행하게 될 때 유용하게 사용됩니다. 특정 Table의 Data가 다른 Table의 Data와 관련 있는 Business Rule이 있을 수 있다. 만일, Business Rule이 변경되면, DB에 있는 Trigger 정의만 변경하면 되며, 모든 Application은 추가적인 변경 없이 새로운 Business Rule을 따르게 됩니다.

TOPIC 03 07 Trigger 작성

Point

Trigger 에 대해 알아봅니다.

2 CLP/제어센터를 통한 Trigger 생성

Trigger를 생성하기 위한 많은 Tool들이 있습니다. “CREATE TRIGGER” 명령도 그 중 하나 입니다.

➲ CLP에서 수행

```
CREATE TRIGGER KJINST1.UPD_PRODINV_TRIG
   AFTER INSERT ON KJINST1.PRODUCT_PURCHASES
   REFERENCING NEW AS NEWROW
   FOR EACH ROW  MODE DB2SQL
BEGIN ATOMIC
   IF ( NEWROW.QTY > 0 ) THEN
      UPDATE KJINST1.PRODUCT
        SET INVENTORY  = INVENTORY - NEWROW.QTY
       WHERE PRODUCT_ID = NEWROW.PRODUCT_ID;
   ELSEIF ( NEWROW.QTY < 0 ) THEN
      UPDATE KJINST1.PRODUCT
        SET INVENTORY  = INVENTORY + NEWROW.QTY
       WHERE PRODUCT_ID = NEWROW.PRODUCT_ID;
   END IF;
END@

db2 –td@ -vf <fie_name>
```

PRODUCT_PURCHASES Table에 대해 INSERT가 수행되면, UPD_PRODINV_TRIG Trigger가 수행됩니다. 수량이 0보다 크면(판매) 재고가 감소됩니다. 수량이 0보다 작으면(반품) 재고가 증가됩니다.
제어센터를 통해 Trigger를 생성하는 방법을 소개합니다. 제어센터에서 Database를 선택한 후, PRODUCT_PURCHASES Table에서 마우스 오른쪽을 클릭합니다.

➲ 제어센터를 통한 Trigger 생성

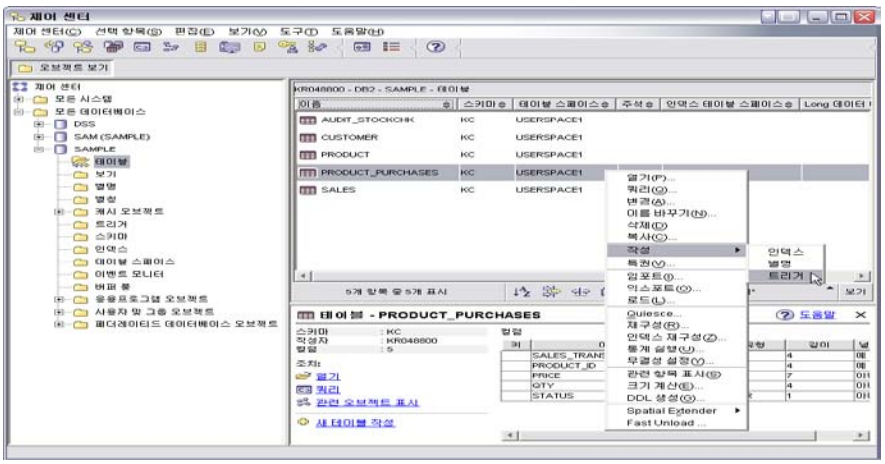


Figure 2407A 제어센터에서 Trigger 작성하기 -1

TOPIC 03 07 Trigger 작성

Point

Trigger 에 대해 알아봅니다.

"트리거 작성" 다이얼로그에서 Trigger 정의를 기술합니다.
화면이 "트리거" 와 "트리거 조치" 탭으로 나뉘어 집니다. "주석" 을 제외한 모든 항목을 정의합니다.

- 트리거 스키마 : KC
- 트리거 이름 : UPT_PRODINV_TRIG
- 테이블 또는 뷰 스키마 : KC
- 테이블 또는 뷰 이름 : PRODUCT_PURCHASE
- 트리거 조치 시간 : AFTER
- 트리거가 실행되도록 하는 조작 : 삽입

➲ 트리거 작성 : 트리거 탭

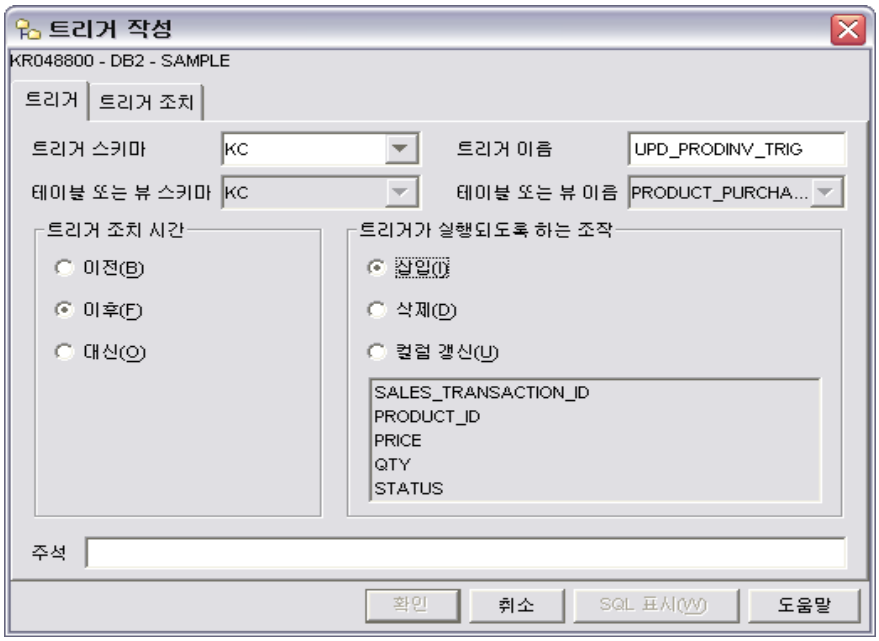


Figure 2407B ••• 제어센터에서 트리거 작성하기 -2

INSERT/UPDATE/DELETE된 이전 행 과 새 행에 대한 참조 명을 정의 할 수 있습니다. 이 전이(Transition) 변수는 Trigger 본문에서 사용됩니다. "OLD Transition 변수" 는 UPDATE/DELETE가 수행되면 생성됩니다.
한편, "NEW Transition 변수" 는 UPDATE /INSERT가 수행되면 생성됩니다.

Triggering SQL	OLD Transition	NEW Transition
DELETE	OLD	
INSERT		NEW
UPDATE	OLD	NEW

TOPIC 03 07 Trigger 작성

Point

Trigger 에 대해 알아봅니다.

➲ 트리거 작성 : 트리거 조치 탭

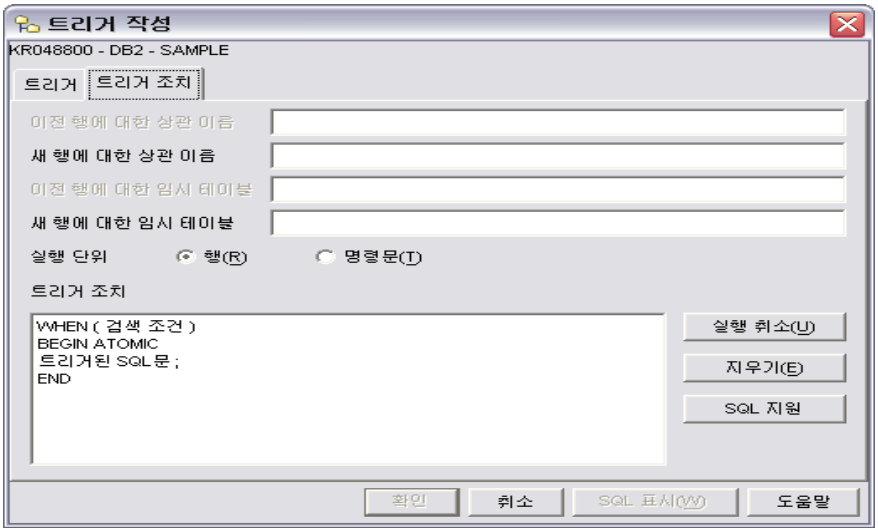


Figure 2407C 제어센터에서 트리거 작성하기 -3

INSERT Trigger를 작성하므로,
새 행에 관련된 참조만 활성화 됩니다. "새 행에 대한 상관 이름"에 "NEWROW"를 입력합니다. 다음은 Triggering SQL문장에 의해 영향을 받는 각 Row 단위로 Trigger를 실행 할 것인지, 또는 Row 수에 관계 없이 각 문장 별로 실행 할 것인지를 선택합니다.
UPT_PRODINV_TRIG Trigger에서는 PRODUCT_PURCHASE에 입력되는 각 Row 단위로 Trigger를 수행하게 합니다. 그러므로 실행단위로서 "행'을 선택합니다.
마지막으로, Trigger가 실행될 때의 SQL 문장을 "트리거 조치"에 기술합니다. 트리거 조치에는 기본적인 템플릿이 제공됩니다. "WHEN"절에는 Trigger 수행을 위한 조건을 정의합니다. 예를 들어, Trigger의 Base Table에 조건을 만족하는 경우에만 Trigger를 수행하도록 정의할 수 있습니다.

➲ WHEN 절

Ex) "가격 x 수량"이 100보다 큰 경우만 수행
WHEN (newrow.price * newrow.qty > 100)

WHEN절 다음에는 "BEGIN ATOMIC" ~ "END"에 해당하는 SQL문장이 정의됩니다.
SQL 문장 중, 하나라도 실패하면 Trigger 조치가 실패 하도록 반드시 "ATOMIC"으로 기술하여야 합니다.
Trigger 문장을 기술할 때는 몇 가지 규칙이 있습니다. 그 중 하나가, BEFORE Trigger에서는 INSERT/UPDATE/DELETE를 기술할 수 없습니다.
만일, Data 수정을 원한다면, AFTER Trigger를 정의하세요. 자세한 내용은 "DB2 SQL Reference Guide"를 참조 하세요!.
계속해서, Trigger 조치를 작성합니다. PRODUCT_PURCHASES Table에 대해 INSERT가 수행되면, UPD_PRODINV_TRIG Trigger가 수행됩니다. 수량이 0보다 크면(판매) 재고가 감소됩니다. 수량이 0보다 작으면(반품) 재고가 증가됩니다.
각 Trigger문장 뒤에는 ";"으로 끝을 맺습니다.

Tip

UPT_PRODINV_TRIG Trigger에서는 PRODUCT_PURCHASE Table에 입력되는 모든 Operation 에 대해 Trigger를 수행하므로, WHEN절을 정의하지 않습니다.

TOPIC 03 07 Trigger 작성

Point

Trigger 에 대해 알아봅니다.

Figure 2407D ••• 제어센터에서 트리거 작성하기 -4

Tool에 의해 생성된 "CREATE TRIGGER" 에 대한 문장을 보려면 "SQL 표시" 를 Click합니다.

"트리거 작성" 창에서 바로 생성하기 위하여 "확인" 을 Click합니다.
제어센터에서 트리거 Object를 선택하여 생성된 Trigger를 확인합니다.

Tip

해당 창에서 SQL문장을 복사 하거나, 다른 파일로 저장 할 수 있습니다.

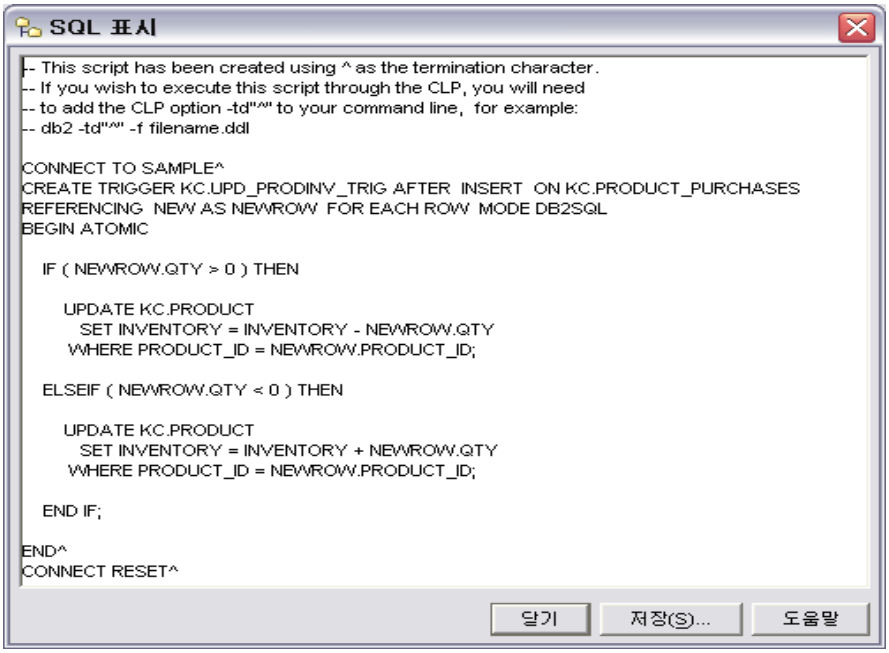


Figure 2407E ••• 제어센터에서 트리거 작성하기 -5

TOPIC 03 07 Trigger 작성

Point

Trigger 에 대해 알아봅니다.

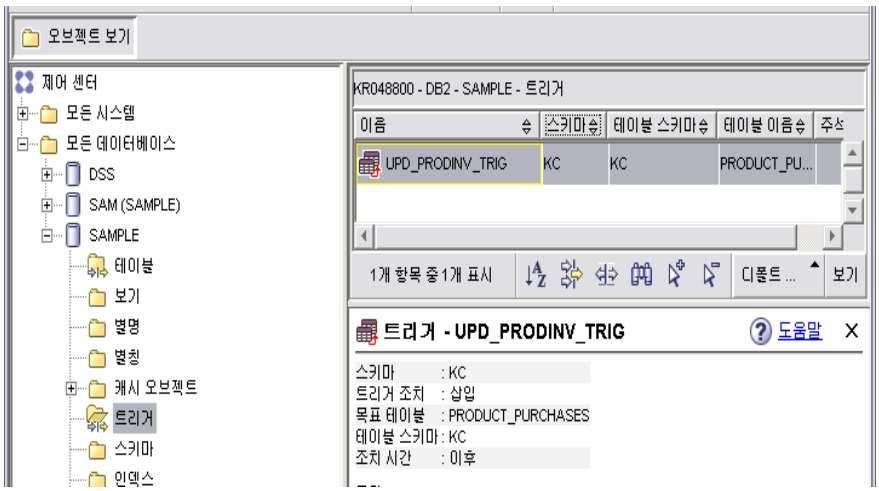


Figure 2407F 제어센터에서 트리거 작성하기 -6

3 Create Trigger 문

```
>>-CREATE TRIGGER--trigger-name--+-NO CASCADE BEFORE-+---------->
                                 +-AFTER-------------+
                                 '-INSTEAD OF--------'
>--+-INSERT-------------------------+--ON--+-table-name-+------->
   +-DELETE-------------------------+      '-view-name--'
   '-UPDATE--+--------------------+-'
             |    .-,-----------. |
             |    V             | |
             '-OF---column-name-+-'
>--+-------------------------------------------------------------+-->
   |              .-------------------------------------------. |
   |              V                      .-AS-.                 | |
   '-REFERENCING -----------------+-OLD--+----+--correlation-name-+-+-'
                                  |      .-AS-.                  |
                                  +-NEW--+----+--correlation-name-+
                                  |            .-AS-.            |
                                  +-OLD_TABLE--+----+--identifier-+
                                  |            .-AS-.            |
                                  '-NEW_TABLE--+----+--identifier-'
>--+-FOR EACH ROW--------------+--MODE DB2SQL------------------->
   '-FOR EACH STATEMENT--------'
>--| triggered-action |------------------------------------><

triggered-action:
|--+-------------------------------------+--------------------->
   '--------WHEN--(--search-condition--)-'
>--+---------+----SQL-procedure-statement----------------------|
   '--label--'
```

TOPIC 03 07 Trigger 작성

Point

Trigger 에 대해 알아봅니다.

4 Trigger 사용 예제

➲ BEFORE INSERT TRIGGER

```
CREATE TRIGGER KJINST1.default_class_end
   NO CASCADE
   BEFORE INSERT ON KJINST1.cl_sched
   REFERENCING NEW AS n
   FOR EACH ROW
   MODE DB2SQL
WHEN (n.ending IS NULL)
SET n.ending = n.starting + 1 HOUR
```

➲ AFTER UPDATE TRIGGER

```
CREATE TRIGGER KJINST1.audit_emp_sal
   AFTER UPDATE OF salary ON KJINST1.employee
   REFERENCING OLD AS o
               NEW AS n
   FOR EACH ROW
   MODE DB2SQL
INSERT INTO KJINST1.audit
VALUES (CURRENT TIMESTAMP
     ,  ' Employee '
        | | o.empno
        | | ' salary changed from '
        | | CHAR(o.salary)
        | | ' to '
        | | CHAR(n.salary)
        | | ' by '
        | | USER)
```

➲ SQL PL을 이용한 BEFORE INSERT TRIGGER

```
CREATE TRIGGER KJINST1.validate_sched
   NO CASCADE BEFORE INSERT ON KJINST1.cl_sched
   REFERENCING NEW AS n
   FOR EACH ROW
   MODE DB2SQL
vs: BEGIN ATOMIC

  -- supply default value for ending time if null
  IF (n.ending IS NULL) THEN
     SET n.ending = n.starting + 1 HOUR;
  END IF;

  -- ensure that class does not end beyond 9PM
  IF (n.ending > '21:00') THEN

     SIGNAL SQLSTATE '80000'
       SET MESSAGE_TEXT='class ending time is'
                  | | ' beyond 9pm';

  ELSEIF (n.DAY=1 or n.DAY=7) THEN

     SIGNAL SQLSTATE '80001'
       SET MESSAGE_TEXT='class cannot be'
                  | | 'scheduled on a weekend';

  END IF;
END vs
```

TOPIC 03 07 Trigger 작성

Point

Trigger 에 대해 알아봅니다.

➲ INSTEAD OF TRIGGER

```
CREATE VIEW KJINST1.org_by_division
 (division, number_of_dept)
AS
 SELECT division, count(*)
 FROM KJINST1.org
 GROUP BY division


CREATE TRIGGER KJINST1.upd_org
   INSTEAD OF UPDATE
   ON KJINST1.org_by_division
   REFERENCING OLD AS o
               NEW AS n
   FOR EACH ROW
   MODE DB2SQL
BEGIN ATOMIC

 IF (o.number_of_dept != n.number_of_dept) THEN

   SIGNAL SQLSTATE '80001'
   SET MESSAGE_TEXT
   ='The number of department is not updatable.';

 END IF;

 UPDATE KJINST1.org
   SET division = n.division
  WHERE division = o.division;

END
```

View에 대한 Update후의 처리 결과를 확인할 수 있습니다.

```
select * from KJINST1.org_by_division;

DIVISION   NUMBER_OF_DEPT
---------- --------------
Corporate               1
Eastern                 5
Midwest                 2
Western                 2

UPDATE KJINST1.org_by_division
  SET division='Eastern_1'
 WHERE division='Eastern';

select * from KJINST1.org_by_division;

DIVISION   NUMBER_OF_DEPT
---------- --------------
Corporate               1
Eastern_1               5
Midwest                 2
Western                 2
```

TOPIC 03 07 Trigger 작성

Point

Trigger 에 대해 알아봅니다.

➲ SP를 Call하는 Trigger

```
CREATE TRIGGER KJINST1.tr_autoproc_order
    AFTER INSERT ON KJINST1.sales
    FOR EACH ROW MODE DB2SQL
    BEGIN ATOMIC

      CALL KJINST1.process_neworder();

    END
```

5 View Trigger 와 Table Trigger 비교

다음 예를 통해 View Trigger와 Table Trigger의 차이점을 살펴 봅니다.
이를 위해 Table 과 View, Table Insert Trigger와 View Insert Trigger를 사용합니다.

➲ Table, View 및 Trigger 생성

```
CREATE TABLE KJINST1.T_AIRPORT (
  AIRPORT_CODE CHAR( 3) NOT NULL,
  AIRPORT_NAME CHAR(50)
);

CREATE VIEW KJINST1.V_AIRPORT
AS
  SELECT *
    FROM KJINST1.T_AIRPORT
;

CREATE TRIGGER KJINST1.INSERT_T_AIRPORT
    AFTER INSERT
    ON KJINST1.T_AIRPORT
    FOR EACH ROW
    MODE DB2SQL
BEGIN ATOMIC
END;

CREATE TRIGGER KJINST1.INSERT_V_AIRPORT
    INSTEAD OF INSERT
    ON KJINST1.V_AIRPORT
    FOR EACH ROW
    MODE DB2SQL
BEGIN ATOMIC
END;
```

TOPIC 03 07 Trigger 작성

Point

Trigger 에 대해 알아봅니다.

➲ Table, View 및 Trigger 생성

```
INSERT INTO KJINST1.t_airport
      VALUES ('KOR', 'SEOUL_T');
INSERT INTO KJINST1.v_airport
      VALUES ('KOR', 'SEOUL_V');

select * from KJINST1.t_airport;

AIRPORT_CODE AIRPORT_NAME
------------ -------------------------
KOR          SEOUL_T

select * from KJINST1.v_airport;

AIRPORT_CODE AIRPORT_NAME
------------ -------------------------
KOR          SEOUL_T
```

➲ INSTEAD OF Trigger 수정

```
CREATE TRIGGER KJINST1.insert_v_airport
   INSTEAD OF INSERT
   ON KJINST1.v_airport
   REFERENCING NEW AS n
   FOR EACH ROW
   MODE DB2SQL
BEGIN ATOMIC
   INSERT INTO KJINST1.t_airport
   VALUES (n.airport_code, n.airport_name);
END;
```

TOPIC 03 08 모듈 작성

Point

DB2 모듈은 저장 프로시저, 함수, 변수, 커서드의 오브젝트를 하나의 묶음으로 관리합니다. 오라클의 패키지(Package)와 유사한 기능을 수행합니다.

1 모듈(Module)

➲ DB2 9.7에서 새롭게 제공하는 모듈을 통해서 일련의 DBMS의 오브젝트를 그룹으로 관리합니다.
➲ 오브젝트 리스트: 저장프로시저, 사용자 함수, 배열,레코드, 사용자 정의 커서등 서브 모듈로 지정할 수 있습니다.

2 모듈 작성 실습 : 모듈에 프로시저 포함하기

Tip

모듈에 등록된 프로시저나 함수는 독립적으로도 실행이 가능합니다. 하지만 모듈을 사용하면 모듈간의 의존성 관리가 쉽습니다.

1. 간단 테스트 테이블 하나 작성

```
create table  TEST_TEST (name varchar(20) , entrydate
timestamp)
```

4. 모듈 선언

```
CREATE MODULE mod_test1
```

5. 선언한 모듈에 프로시져 추가 (고정변수)

```
ALTER MODULE mod_test1
  PUBLISH PROCEDURE proc1_test2  (name  varchar(20))
        BEGIN
         call  proc1_test  ('kasung');
        END
```

6. 모듈 실행

```
call mod_test1. proc1_test3 ('HONG')
```

7. 데이타 이상유무 확인

```
select * from test_test

NAME         ENTRYDATE
------------  ----------------------
kasung       2009. 4. 3 오전 3:37:36
```

8. 스키마.모듈명.프로시져명으로 도 실행 예시

```
call  db2inst1.mod_test1. proc1_test3 ('KOREA')
```

TOPIC 03 08 모듈 작성

Point

DB2 모듈은 저장 프로시저, 함수, 변수, 커서등의 DB2 오브젝트를 하나의 묶음으로 관리합니다. 오라클의 패키지(Package)와 유사한 기능을 수행합니다.

3 모듈 작성 실습: 모듈에 함수 포함하기

```
1.        기존의 모듈에 appending 하기

          ALTER MODULE mod_test1
            PUBLISH function  fn_addsum123  (p1 int)
                              returns int
                    BEGIN
                          return ( select fn_addsum(p1) from
          sysibm.sysdummy1 );
                    END

2.모듈의 함수 실행

select  mod_test1.fn_addsum123 (10)  from sysibm.sysdummy1

        1
          ----
          1000
```

4 모듈 작성 실습: 모듈에 사용자 정의 타입 포함하기

```
1. CREATE MODULE INVENTORY

2. ALTER MODULE INVENTORY ADD
  TYPE ITEMLIST AS INTEGER ARRAY[VARCHAR(100)]

3. ALTER MODULE INVENTORY ADD VARIABLE ITEMS ITEMLIST
```

TOPIC 03 08 모듈 작성

Point

사용자 정의 모듈과 서브 모듈 정보를 조회해 봅니다.

➲ 모듈 정보 조회 (1) : DB2 9.7에 추가된 관리자 뷰 이용

```
SQL> SELECT MODULENAME, DIALECT , MODULETYPE ,
REMARKS   FROM SYSCAT.MODULES

 MODULENAME     DIALECT     MODULETYPE     REMARKS
 -------------  ----------  -------------  --------------------
EMP_ADMIN       PL/SQL      P              PL/SQL Package
Body
 MOD_TEST1      DB2 SQL PL  M              (null)

SQL>SELECT OBJECTMODULENAME , OBJECTNAME  ,
OBJECTTYPE  FROM SYSCAT.MODULEOBJECTS

 OBJECTMODULENAME     OBJECTNAME      OBJECTTYPE
 -------------------  --------------  -------------
 EMP_ADMIN            GET_DEPT_NAME   FUNCTION
 EMP_ADMIN            UPDATE_EMP_SAL  FUNCTION
...
```

➲ 모듈 정보 조회 (2) : DB2 9.7 이전 버전에서 지원하는 관리자 뷰 예시

```
SQL> SELECT  ROUTINEMODULENAME ,   ROUTINENAME   ,
ROUTINETYPE FROM SYSCAT.ROUTINES

ROUTINEMODULENAME     ROUTINENAME              ROUTI
NETYPE
 --------------------  ----------------------  -------------
 DBMS_OUTPUT           ENABLE                    P
 DBMS_OUTPUT           GET_LINES                 P
 DBMS_ALERT            INIT                    P
```

UNIT 25

객체 관계형 특성

DB2가 지원하는 객체 관계형 (Object-Relational) 특성인 사용자 정의 개별 유형과 사용자 정의 구조화 유형 (UDT), 사용자 정의 함수(UDF), 사용자 정의 메소드(UDM), 사용자 정의 구조화 유형을 이용한 테이블 (Typed Table) 등에 대해 설명합니다.

DB2 9.7 개발자 가이드

Administrator Edition

- 객체 관계형 특성의 개요
- 사용자 정의 개별 유형
- 사용자 정의 구조화 유형

TOPIC 0401 객체 관계형 특성의 개요

Point

DB2에서의 Object-Relational 의 개념정의를 알아봅니다.

Tip

DB2의 확장된 기능으로 RDBMS로서의 Object Oriented의 개념과 Methodology를 함께 사용할 수 있습니다.

1 DB2 Object Extensions

➲ 컴퓨터 프로그램 언어 기술에 있어, 최근의 중요한 개발 기술의 하나는 Object-Orientation(객체 지향)입니다. 객체지향이라 함은 응용프로그램의 Object가 classification에 의해 서로 연관이 있는 독립된 Object로서 모델로 만들어 질 수 있는 개념입니다.

➲ 내부의 구체적인 구현사항은 숨겨진 채로, 외부의 function과 attribute는 구체화 되어 공개됩니다.

➲ DB2의 Object 기술은 RDBMS기술을 뿐만 아니라 포함하여 많은 Object 기술과 관련하여 많은 장점을 제공합니다.

2 DB2 Object-Relational 특성

➲ Data Type for very large objects
- TEXT, AUDIO, ENGINEERING DATA, VIDEO
- Binary Large Objects (BLOB)
- Character Large Objects (CLOB)
- Double-Byte Character Large Objects (DBCLOB)

➲ User Defined data Type
- Distinct Type
- Structured Type

➲ User Defined behaviors
- User Defined functions (UDF)
- User Defined methods : encapsulated with Structured type.

➲ Index extensions
- External Table function정의를 통해 structure type 및 distinct type에 대한 Index Key값 변경 및 성능최적화를 위한 검색 기능 제공

➲ Constraints
- Unique
- Referential integrity
- Table check
- Triggers

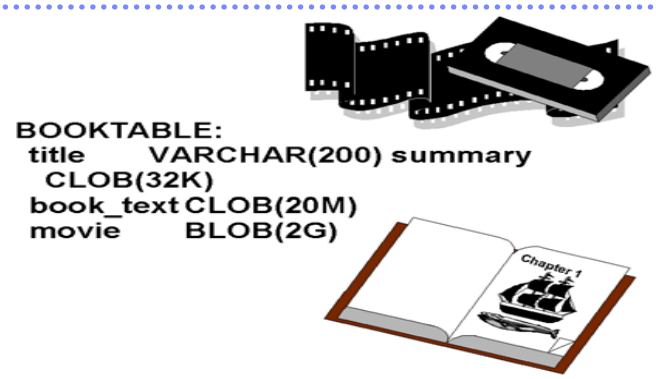


Figure 2501A DB2 Object-Relational 특성

TOPIC 04-02 사용자 정의 개별 유형

Point

User Defined Distinct Type에 대해 알아봅니다.

Tip

User Defined Distinct type을 통해 확장성, 유연성, 일관성 및 성능을 제공합니다.

1 User Defined Distinct Type

➲ **Extensibility**

새로운 Type을 정의 함으로써, Application에서 사용되는 데이터 Type을 다양하게 쓸 수 있습니다.

➲ **Flexibility**

User Defined Function에 사용하여 새로운 데이터 타입을 위한 별도의 Function을 기술함으로써 시스템의 다양성을 제공합니다.

➲ **Consistency**

Distinct Type을 통해 일관된 Process를 합니다. Distinct Type에 정의된 해당 Function을 수행함으로써 일관성을 유지합니다.

➲ **Encapsulation**

Distinct Type에 대한 Function과 연산자를 정의할 수 있습니다. 이를 통해, 수행중인 Application은 distinct type 내의 정의된 Function 내부의 Logic과 독립적으로 수행가능 합니다.

➲ **Performance**

Distinct type은 DBMS에 통합됩니다. 내부적으로는 내장 데이터 타입과 동일한 형태로 표현되며, 내장함수, 비교연산자 및 인덱스 등을 사용함에 있어 동일한 효율성을 제공합니다.

다른 통화(Currency)를 다룰 필요가 있고, 이 통화가 서로 비교되거나 Query에 의해 직접적으로 다뤄지는 것을 허락하지 않는 것을 보증하기 위한 업무를 가정하고 이를 위한 Type을 정의합니다.

즉, 서로 다른 화폐 단위와 가치 비교를 할 경우, 전환이 필요합니다.이를 위해, 필요한 각종 통화를 정의합니다.

```
CREATE DISTINCT TYPE US_DOLLAR
                    AS DECIMAL (9,2) WITH COMPARISONS
CREATE DISTINCT TYPE CANADIAN_DOLLAR
                    AS DECIMAL (9,2) WITH COMPARISONS
CREATE DISTINCT TYPE EURO
                    AS DECIMAL (9,2) WITH COMPARISONS
```

TOPIC 04-02 사용자 정의 개별 유형

Point

User Defined Distinct Type을 통한 Table 정의를 살펴 봅니다.

입사 지원자들의 양식이 Tabled에 저장되었다면, 이 양식으로부터 정보를 얻어내기 위해 function을 사용할 것입니다. 이를 위해 별도의 Type을 정의합니다.

```
CREATE DISTINCT TYPE PERSONAL.APPLICATION_FORM
                                    AS CLOB(32K)
```

LOB Type은 Function을 지원하지 않으므로, 이를 위한 별도의 Type을 통해 Function을 작성합니다.

2 User Defined Distinct Type 을 통한 Table 정의

Sales :

➲ CREATE TABLE US_SALES (
PRODUCT_ITEM INTEGER,
MONTH INTEGER CHECK (MONTH BETWEEN 1 AND 12),
YEAR INTEGER CHECK (YEAR > 1985),
TOTAL US_DOLLAR)

➲ CREATE TABLE CANADIAN_SALES (
PRODUCT_ITEM INTEGER,
MONTH INTEGER CHECK (MONTH BETWEEN 1 AND 12),
YEAR INTEGER CHECK (YEAR > 1985),
TOTAL CANADIAN_DOLLAR)

➲ CREATE TABLE GERMAN_SALES (
PRODUCT_ITEM INTEGER,
MONTH INTEGER CHECK (MONTH BETWEEN 1 AND 12),
YEAR INTEGER CHECK (YEAR > 1985),
TOTAL EURO)

TOPIC 04-02 사용자 정의 개별 유형

Point

Distinct Type에 대한 예를 통해 살펴 봅니다.

3 Distinct Type 에 대한 사용 예

➲ Distinct Type 과 상수의 비교

```
SELECT PRODUCT_ITEM
  FROM US_SALES
 WHERE TOTAL > US_DOLLAR (100000)
   AND month = 7
   AND year = 1999;

SELECT PRODUCT_ITEM
  FROM US_SALES
 WHERE TOTAL > CAST (100000 AS us_dollar)
   AND MONTH = 7
   AND YEAR = 1999;
```

➲ 다른 Type 간의 Cast

```
CREATE FUNCTION EURO_TO_US_DOUBL ( EURO DOUBLE )
RETURNS DOUBLE SPECIFIC EURO_TO_US_DOUBL_DO
LANGUAGE SQL CONTAINS SQL
NO EXTERNAL ACTION
NOT DETERMINISTIC
RETURN EURO * 1.2;

CREATE FUNCTION CDN_TO_US_DOUBL ( CDN DOUBLE )
RETURNS DOUBLE SPECIFIC CDN_TO_US_DOUBL
LANGUAGE SQL CONTAINS SQL
NO EXTERNAL ACTION
NOT DETERMINISTIC
RETURN CDN * 0.87;

CREATE FUNCTION EURO_TO_US_DEC (DECIMAL(9,2))
RETURNS DECIMAL(9,2)
SOURCE EURO_TO_US_DOUBL (DOUBLE);

CREATE FUNCTION US_DOLLAR (EURO)
RETURNS US_DOLLAR
SOURCE EURO_TO_US_DEC (DECIMAL());

CREATE FUNCTION CDN_TO_US_DEC (DECIMAL(9,2))
RETURNS DECIMAL(9,2)
SOURCE CDN_TO_US_DOUBL (DOUBLE);

CREATE FUNCTION US_DOLLAR (CANADIAN_DOLLAR)
RETURNS US_DOLLAR
SOURCE CDN_TO_US_DEC (DECIMAL());
```

TOPIC 04-02 사용자 정의 개별 유형

Point

Distinct Type에 대한 예를 통해 살펴 봅니다.

➲ 다른 Type 간의 비교

```
SELECT US.PRODUCT_ITEM, US.TOTAL
  FROM US_SALES AS US
     ,CANADIAN_SALES AS CDN
     ,GERMAN_SALES AS GERMAN
 WHERE US.PRODUCT_ITEM = CDN.PRODUCT_ITEM
   AND US.PRODUCT_ITEM = GERMAN.PRODUCT_ITEM
   AND US.TOTAL > US_DOLLAR (CDN.TOTAL)
   AND US.TOTAL > US_DOLLAR (GERMAN.TOTAL)
   AND US.MONTH = 7
   AND US.YEAR = 1999
   AND CDN.MONTH = 7
   AND CDN.YEAR = 1999
   AND GERMAN.MONTH = 7
   AND GERMAN.YEAR = 1999;
```

➲ Sourced UDF

```
CREATE FUNCTION SUM (EURO)
RETURNS EURO
SOURCE SYSIBM.SUM (DECIMAL());

CREATE FUNCTION SUM (CANADIAN_DOLLAR)
RETURNS CANADIAN_DOLLAR
SOURCE SYSIBM.SUM (DECIMAL());

CREATE FUNCTION SUM (US_DOLLAR)
RETURNS US_DOLLAR
SOURCE SYSIBM.SUM (DECIMAL());

SELECT PRODUCT_ITEM, US_DOLLAR (SUM (TOTAL))
FROM CANADIAN_SALES
WHERE YEAR = 1994
GROUP BY PRODUCT_ITEM;
```

TOPIC 04-02 사용자 정의 개별 유형

Point

Distinct Type에 대한 예를 통해 살펴 봅니다.

➲ 다른 Type 간의 Assignment

```
CREATE TABLE US_SALES_2006
(PRODUCT_ITEM INTEGER,
 TOTAL        US_DOLLAR);

CREATE TABLE GERMAN_SALES_2006
(PRODUCT_ITEM INTEGER,
 TOTAL        US_DOLLAR);

CREATE TABLE CANADIAN_SALES_2006
(PRODUCT_ITEM INTEGER,
 TOTAL        US_DOLLAR);
```

```
INSERT INTO US_SALES_2006
SELECT PRODUCT_ITEM, SUM (TOTAL)
  FROM US_SALES
 WHERE YEAR = 1994
 GROUP BY PRODUCT_ITEM;

INSERT INTO GERMAN_SALES_2006
SELECT PRODUCT_ITEM, US_DOLLAR (SUM (TOTAL))
  FROM GERMAN_SALES
 WHERE YEAR = 1994
 GROUP BY PRODUCT_ITEM;

INSERT INTO CANADIAN_SALES_2006
SELECT PRODUCT_ITEM, US_DOLLAR (SUM (TOTAL))
  FROM CANADIAN_SALES
 WHERE YEAR = 1994
 GROUP BY PRODUCT_ITEM;
```

➲ 다른 Type 간의 UNION

```
CREATE VIEW ALL_SALES AS
SELECT PRODUCT_ITEM, MONTH, YEAR, TOTAL
  FROM US_SALES
 UNION ALL
SELECT PRODUCT_ITEM, MONTH, YEAR
     , US_DOLLAR (TOTAL) TOTAL
  FROM CANADIAN_SALES
 UNION ALL
SELECT PRODUCT_ITEM, MONTH, YEAR
     , US_DOLLAR (TOTAL) TOTAL
  FROM GERMAN_SALES;
```

TOPIC 0403 사용자 정의 구조화 유형

Point

User Defined Structured Type에 대해 살펴 봅니다.

Tip

"AS"절 이하에 Type과 관련된 속성을 정의합니다. BusinessUnit_T는 Name과 HeadCount 속성을 가집니다. Structured Type을 생성하기 위해서는 "MODE DB2SQL"을 지정해야 합니다.

1 User Defined Structured Type

➲ Structured Type은 잘 Attribute로 구성된 잘 정의된 구조체 Modeling에 유용합니다. 각 Attribute는 Type의 속성들입니다.

➲ Type을 생성하기 위해, Type의 이름, attribute의 이름 및 Data type를 기술 합니다. 또한 선택적으로 Type에 대한 reference 방법을 정의합니다.

➲ 예제 : BusinessUnit_T 정의

```
CREATE TYPE BusinessUnit_t AS (
      Name       VARCHAR(20)
     ,Headcount  INT
) REF  USING INT MODE DB2SQL;

CREATE TYPE ADDRESS_T AS (
      STREET VARCHAR(30)
     ,NUMBER VARCHAR(15)
     ,CITY   VARCHAR(30)
     ,STATE  VARCHAR(10)
) REF USING INTEGER MODE DB2SQL;
```

➲ Structured Type의 특징

1. Inheritance : Structured Type을 정의 할 때 subtype을 포함할 수 있습니다. 즉, subtype에 포함된 attribute를 재 사용할 수 있습니다.

2. Storing instance of structured type :
   → As a row in a table
   Create table Person of Person_t ...
   → As a value in a column
   Create table Properties
   (ParcelNum Int,
   Photo BLOB(2k),
   Address Address_t)...

TOPIC 04-03 사용자 정의 구조화 유형

Point

User Defined Structured Type에 대한 예를 통해 살펴 봅니다.

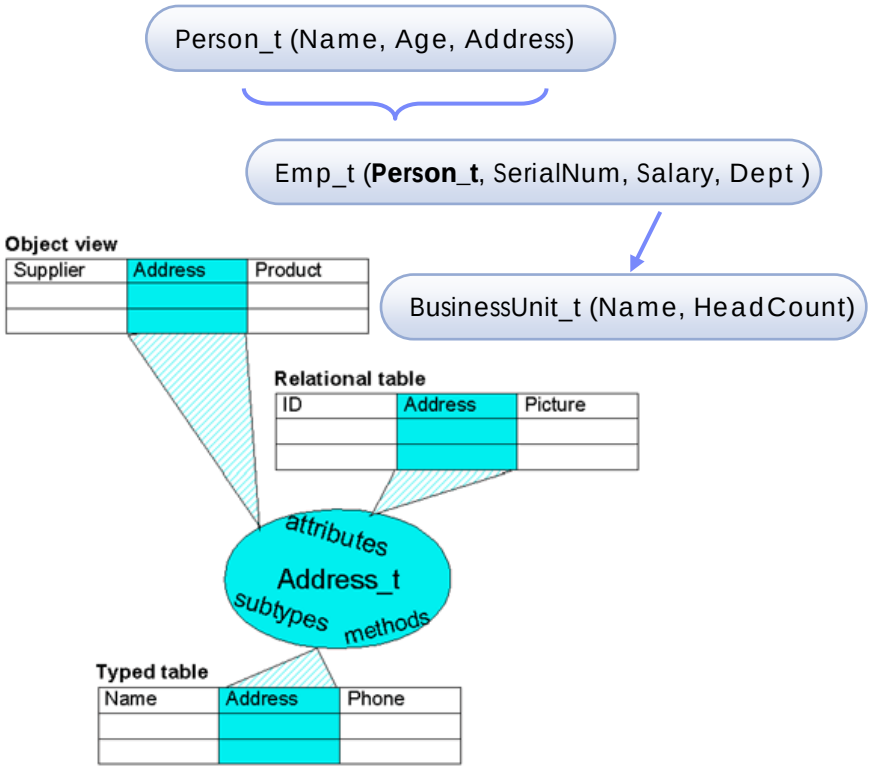


Figure 2503A • • • User Defined Structured Type 예제

2 Structured Type 생성

➲ Structured Type은 다른 Structured Type을 포함하여 생성될 수 있습니다.

Tip

Original Structured Type을 "Super Type"이라 하고, 새로 생성된 Structured Type을 "Sub Type" 이라고 합니다.

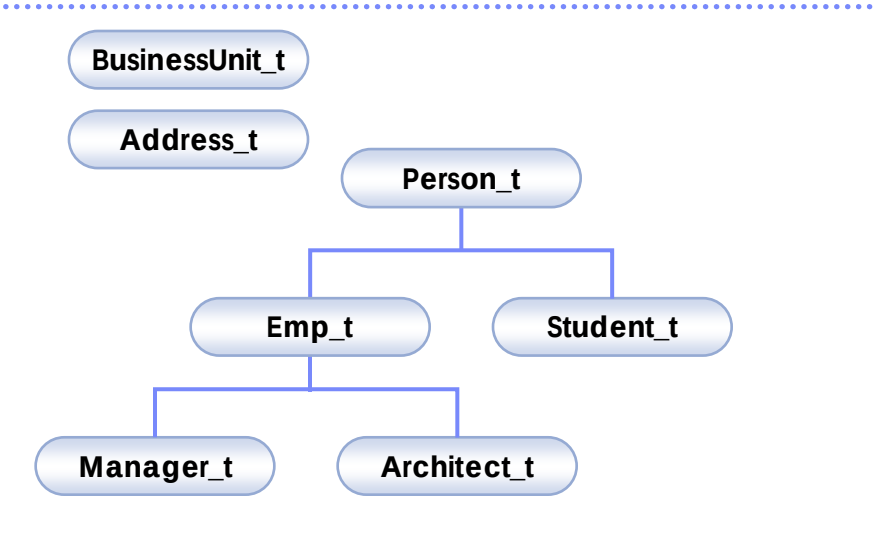


Figure 2503B • • • Structured Type예제

TOPIC 04-03 사용자 정의 구조화 유형

Point

User Defined Structured Type에 대한 예를 통해 살펴 봅니다.

➲ 예제 : BusinessUnit_T 정의

```
CREATE TYPE PERSON_T AS (
      NAME VARCHAR(20)
     ,AGE INT
     ,ADDRESS ADDRESS_T
)
INSTANTIABLE
REF USING VARCHAR(13) FOR BIT DATA MODE DB2SQL;


CREATE TYPE EMP_T UNDER PERSON_T AS (
      SERIALNUM INT
     ,SALARY    DECIMAL (9,2)
     ,DEPT      REF(BUSINESSUNIT_T)
) MODE DB2SQL;


CREATE TYPE STUDENT_T   UNDER PERSON_T AS (
      SERIALNUM    VARCHAR(6)
     ,GPA DOUBLE ) MODE DB2SQL;


CREATE TYPE MANAGER_T   UNDER EMP_T    AS (
      BONUS       DECIMAL(7,2)
) MODE DB2SQL;


CREATE TYPE ARCHITECT_T UNDER EMP_T    AS (
      STOCKOPTION INTEGER
) MODE DB2SQL;
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

Reference type과 Representation Type 에 대해 살펴 봅니다.

Tip

모든 Structured Type에 대해 DB2는 자동적으로 Companion Type을 생성합니다.

Tip

Reference Type을 Typed Table의 Reference를 저장하기 위해 사용합니다. 또한, Table의 각각의 Row를 참조하기 위해 사용합니다.

Tip

REF USING 절로 그 type을 명시하지 않으면, DB2는 VARCHAR FOR BIT DATA (16)의 Default Type을 사용합니다.

3 Reference Type

➲ Companion Type은 Reference Type이라 불리고, 그것이 참조하는 Structured Type은 Referenced Type이라고 불립니다.

➲ SQL 문장에서 Reference Type을 사용하기 위해서는, REF (type-name)을 사용하며, type-name은 Referenced type을 나타냅니다.

➲ DB2는 Type된 Table에서 Reference Type을 Object Identifier Column으로 사용합니다. Object Identifier는 Typed Table Hierarchy에서 Unique Identifier로 사용됩니다.

4 Representation Type

➲ Type Hierarchy의 Root Type을 생성할 때, "Create Type ... REF USING ..."을 사용하여 참조를 위한 Base Type을 명시합니다.
이때, Base Type을Representation Type이라고 합니다.

➲ Root Type의 Representation Type은 모든 Subtype에 상속됩니다.
"REF USING ..." 은 Type Hierarchy에서 Root type에서만 정의할 수 있습니다.

In the examples used throughout this section, the representation type for the BusinessUnit_t type is INTEGER, while the representation type for Person_t is VARCHAR(13).

BusinessUnit_t Representation Type : Integer
Person_t Representation Type : VARCHAR(13)

TOPIC 04-03 사용자 정의 구조화 유형

Point

Casting 과 Reference Type 에 대해 살펴 봅니다.

Tip

기본값은 Structured Type과 representation Type의 이름을 사용합니다.

5 Casting 과 Reference Type 비교

➲ DB2는 Reference Type과 Representation Type간의 Cast를 위해 자동으로 Casting function을 생성합니다.

➲ Create Type 문장의 "CAST WITH ..."절을 사용하여 Cast function의 이름을 지정할 수 있습니다.

➲ Ex : Create Type Person_t ...

다음은 "Create Type Person_t..."에 의해 자동으로 생성되는 Function 입니다.

```
CREATE FUNCTION VARCHAR (REF (Person_t) )
RETURNS VARCHAR;
```

또한, 반대의 Operation을 위한 function을 생성합니다.

```
CREATE FUNCTION Person_t(VARCHAR(13))
RETURNS REF (Person_t);
```

6 기타 System-Generated Routine

➲ DB2는 Structured Type을 생성할 때 마다, 내재적으로 Function과 method를 생성합니다 – Constructor, Observer, Modify

➲ Constructor

```
Constructor function :
CREATE FUNCTION Person_t  ( ) RETURNS Person_t
CREATE FUNCTION Manager_t ( ) RETURNS Manager_t
```

➲ Mutator Method

Mutator Method는 Object의 개별 Attribute별로 존재합니다. Mutator Method가 호출될 때, attribute의 새로운 값을 반환 받습니다.

예로, Person_t는 다음의 각각의 속성에 대해 Mutator Method를 만듭니다. - name, age, address.

```
ALTER TYPE Person_t
    ADD METHOD AGE(int)
  RETURNS Person_t;
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

Type에 대한 Method 정의에 대해 살펴 봅니다.

➲ Observer Method

Observer method는 Object의 개별 Attribute별로 존재합니다.
Observer Method는 Object의 값을 Return합니다.
예로, Person_t는 다음의 Observer method를 만듭니다.

```
ALTER TYPE Person_t
    ADD  METHOD AGE()
 RETURNS INTEGER;
```

7 Type 에 대한 Method 정의

Structured Type에 대해 Method를 정의 할 수 있습니다. Create Method 생성 이전에 Method Specification 정의 되어야 합니다.

➲ Method 생성

```
  ALTER TYPE EMP_T
    ADD METHOD CALC_BONUS (RATE DOUBLE)
 RETURNS DECIMAL(9,2)
LANGUAGE SQL
CONTAINS SQL
     NO EXTERNAL ACTION DETERMINISTIC;

  CREATE METHOD CALC_BONUS (RATE DOUBLE)
 RETURNS DECIMAL(9,2)
    FOR EMP_T
  RETURN SELF..SALARY * RATE;
-----
  ALTER TYPE EMP_T
    ADD METHOD CALC_BONUS (RATE INT)
 RETURNS DECIMAL(9,2)
LANGUAGE SQL
CONTAINS SQL
     NO EXTERNAL ACTION DETERMINISTIC;

  CREATE METHOD CALC_BONUS (RATE INT)
 RETURNS DECIMAL(9,2)
    FOR EMP_T
  RETURN SELF..SALARY * RATE;
```

TOPIC 0403 사용자 정의 구조화 유형

Point

Typed Table에 대한 Object 저장에 대해 살펴 봅니다.

Tip

Typed Table은 다른 Table이 참조할 수 있는 Identity Attribute를 가지고 있습니다.

8 Typed Table에 대한 Object 저장

➲ Typed Table에 Row 형태로 저장하거나, Structured type의 속성을 가진 Column에 저장 할 수 있습니다.

➲ Pserson_t를 위한 Create Table 문장

```
CREATE TABLE Person OF Person_t
(REF IS Oid USER GENERATED);
```

➲ Person Table에 Data Insert

```
INSERT INTO Person (Oid, Name, Age)
VALUES (Person_t('a'), 'Andrew', 29);
```

OID	NAME	AGE	Address
a	Adnrew	29	NULL

➲ Person Table에 Data Update

```
UPDATE Person SET Age=30 WHERE Name='Andrew';
```

OID	NAME	AGE	Address
a	Adnrew	30	NULL

➲ Person_t Table 의 subtable생성 → Emp_t를 위한 Table

Person table속성을 가지 Emp Table을 subtable 이라고 합니다.

```
CREATE TABLE Emp OF Emp_t UNDER Person
INHERIT SELECT PRIVILEGES
( SerialNum WITH OPTIONS NOT NULL,
  Dept      WITH OPTIONS SCOPE BusinessUnit);
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

Typed Table에 대한 Object 저장에 대해 살펴 봅니다.

➲ Employee Table에 Data Insert

```
INSERT INTO Emp
  (Oid, Name, Age, SerialNum, Salary)
VALUES
  (Emp_t('s'), 'Susan', 39, 24001, 37000.48);
```

OID	NAME	AGE	ADDRESS	SerialNum	Salary	Dept
s	Susan	39		24001	37000.48	

➲ Table 조회

```
SELECT oid, name, age, salary FROM emp;
OID     NAME    AGE         SALARY
------- ------- ----------- -----------
x'73'   Susan           39    37000.48

SELECT oid, name, age FROM person;
OID     NAME    AGE
------- ------- -----------
x'61'   Andrew          30
x'73'   Susan           39
```

TOPIC 0403 사용자 정의 구조화 유형

Point

Typed Table 간의 Relationship 정의에 대해 살펴 봅니다.

9 Typed Table간의 Relationship 정의

Typed Table과 Typed Table간의 관계를 정의할 수 있으며, 동일한 Typed Table에 대해 관계를 정의할 수 있습니다.

➲ Emp Table과 Business Table간의 관계 정의

```
update emp set dept = BusinessUnit_t(1)
  where oid = emp_t('s');
update emp set dept = NULL
  where oid = emp_t('s');
```

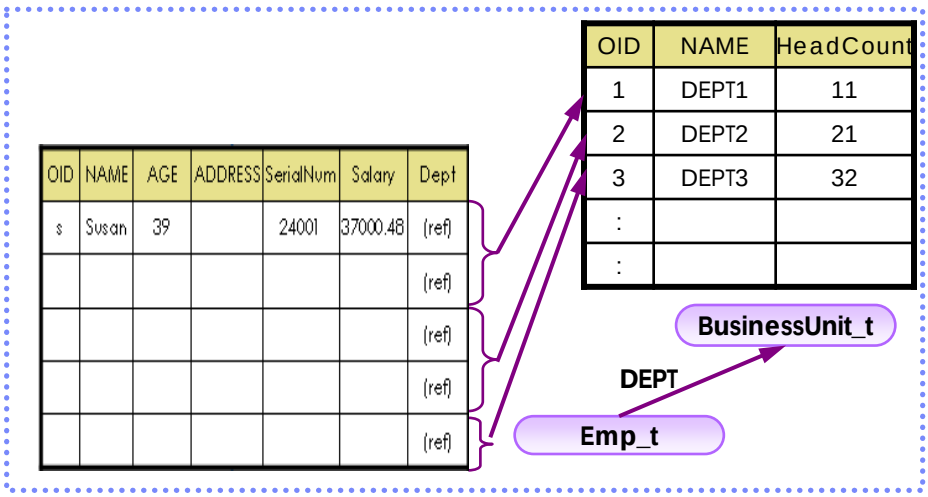


Figure 2503C Emp table과 Business Table간의 관계 정의

➲ Emp Table 조회

```
select oid, name
     ,dept->name
     ,dept->headcount
 from emp
where oid = emp_t('s')"

OID    NAME   NAME   HEADCOUNT
------ ------ ------ -----------
x'73'  Susan  DEPT1         11


INSERT INTO BusinessUnit (Oid, Name, Headcount)
  VALUES (BusinessUnit_t(1), 'DEPT1',11)
       ,(BusinessUnit_t(2), 'DEPT2',21)
       ,(BusinessUnit_t(3), 'DEPT3',32);

select * from businessunit;
OID         NAME    HEADCOUNT
----------- ------- -----------
          1 DEPT1           11
```

TOPIC 0403 사용자 정의 구조화 유형

Point

Column에 Object를 저장하는 것에 대해 살펴 봅니다.

10 Column에 Object 저장

DBMS의 built-in data type으로 Data를 저장하기에 적절하지 않을 경우, Object를 column에 저장할 수 있다.

➲ Person Table

OID	NAME (VARCHAR)	AGE (INT)	Address (Address_t)			
			Street	Number	City	State

➲ Person table에 address 저장

```
UPDATE EMP
  SET ADDRESS=ADDRESS..NUMBER('4869')..STREET('APPLETREE')
 WHERE NAME='FRANKY'
  AND ADDRESS..STATE='CA';

UPDATE EMP
  SET ADDRESS..NUMBER = '4869',
      ADDRESS..STREET = 'APPLETREE'
 WHERE NAME='FRANKY'
  AND ADDRESS..STATE='CA' ;

INSERT INTO Person (Oid, Name, Age, Address)
VALUES ( Person_t('B')
    , 'Billy'
    , 29
    , ADDRESS_T('Gang Name Dogok','123','Seoul','Korea'));
```

➲ Address_t의 Table function

```
CREATE FUNCTION ADDRESS_T
   (street VARCHAR(30),
    number VARCHAR(15),
    city   VARCHAR(30),
    state  VARCHAR(20) )
  RETURNS ADDRESS_T
 LANGUAGE SQL
  RETURN ADDRESS_T()..street(street)
                ..number(number)
                ..city(city)
                ..state(state);
```

TOPIC 0403 사용자 정의 구조화 유형

Point

Typed table에서 Structured Type 사용하는 것에 대해 살펴봅니다.

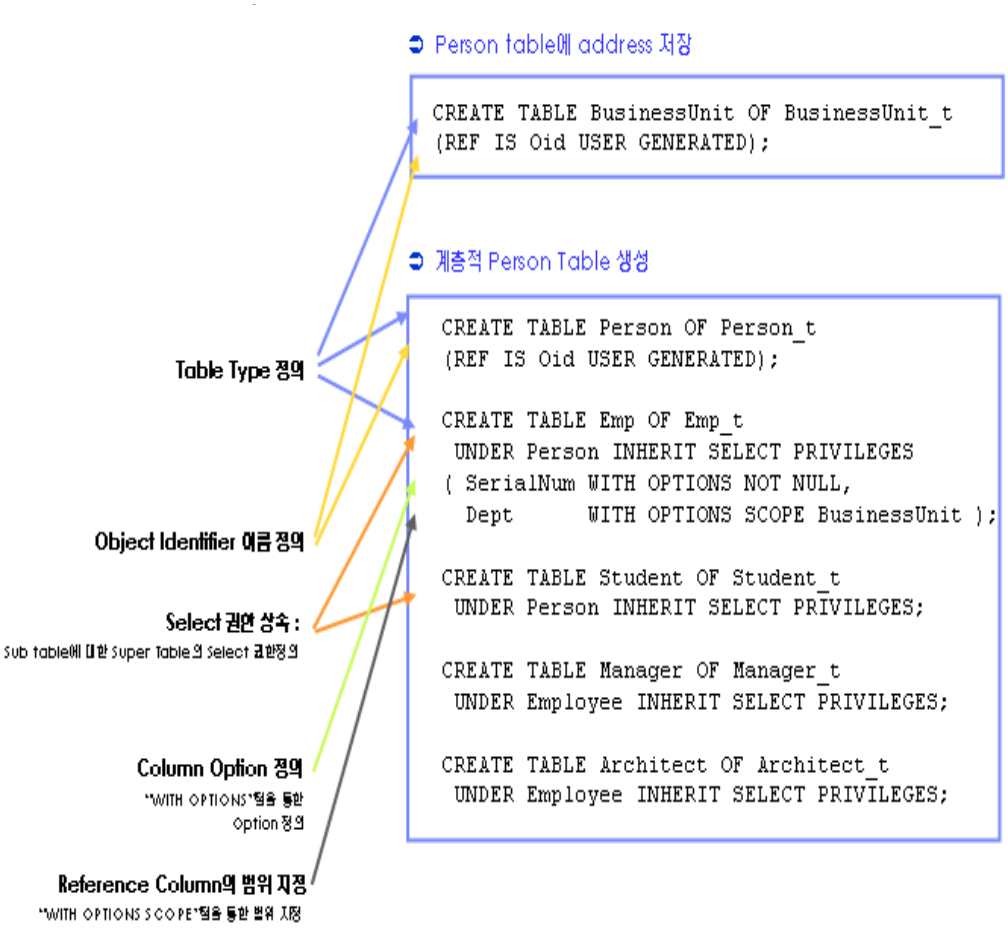
Tip

Typed Table은 "Create Type"문장을 통해 기술된 Object를 실제로 저장하는데 사용할 수 있습니다.

11 Typed Table에서 Structured Type 사용

"Create Table"문장을 사용하여 다양한 Typed Table을 생성 할 수 있습니다. 또한 Structured Type 구조를 가진 Type을 통해 계층적 Table구조를 생성 할 수 있습니다.

➲ Typed Table 생성



TOPIC 04-03 사용자 정의 구조화 유형

Point

Typed table Record 생성에 대해 살펴봅니다.

12 Typed Table Record 생성

➲ BusinessUnit Table

```
INSERT INTO BusinessUnit (Oid, Name, Headcount)
VALUES(BusinessUnit_t(1), 'Toy', 15);

INSERT INTO BusinessUnit (Oid, Name, Headcount)
VALUES(BusinessUnit_t(2), 'Shoe', 10);
```

➲ Person Table

```
INSERT INTO Person (Oid, Name, Age)
VALUES(Person_t('a'), 'Andrew', 20);

INSERT INTO Person (Oid, Name, Age)
VALUES(Person_t('b'), 'Bob', 30);

INSERT INTO Person (Oid, Name, Age)
VALUES(Person_t('c'), 'Cathy', 25);
```

➲ Person Table

```
INSERT INTO Emp (Oid, Name, Age, SerialNum, Salary, Dept)
VALUES(Employee_t('d'), 'Dennis', 26, 105, 30000,
BusinessUnit_t(1));

INSERT INTO Emp (Oid, Name, Age, SerialNum, Salary, Dept)
VALUES(Employee_t('e'), 'Eva', 31, 83, 45000,
BusinessUnit_t(2));

INSERT INTO Emp (Oid, Name, Age, SerialNum, Salary, Dept)
VALUES(Employee_t('f'), 'Franky', 28, 214, 39000,
BusinessUnit_t(2));
```

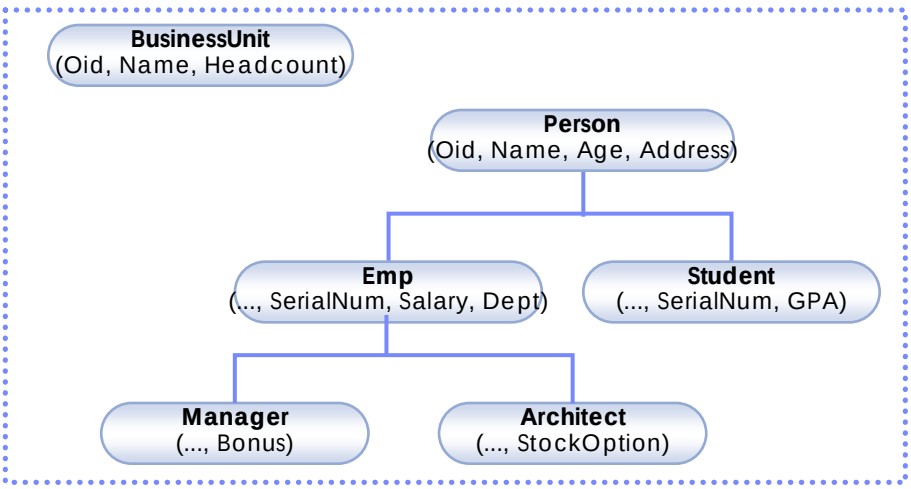


Figure 2503C Typed Table Record 생성 구조

TOPIC 04-03 사용자 정의 구조화 유형

Point

Typed table에서 Structured Type 사용하는 것에 대해 살펴봅니다.

➲ Student Table

```
INSERT INTO Student (Oid, Name, Age, SerialNum, GPA)
VALUES(Student_t('g'), 'Gordon', 19, '10245', 4.7);

INSERT INTO Student (Oid, Name, Age, SerialNum, GPA)
VALUES(Student_t('h'), 'Helen', 20, '10357', 3.5);
```

➲ Manager Table

```
INSERT INTO Manager (Oid, Name, Age, SerialNum,
   Salary, Dept, Bonus)
VALUES(Manager_t('i'), 'Iris', 35, 251, 55000,
   BusinessUnit_t(1), 12000);

INSERT INTO Manager (Oid, Name, Age, SerialNum,
   Salary, Dept, Bonus)
VALUES(Manager_t('j'), 'Christina', 10, 317, 85000,
   BusinessUnit_t(1), 25000);

INSERT INTO Manager (Oid, Name, Age, SerialNum,
   Salary, Dept, Bonus)
VALUES(Manager_t('k'), 'Ken', 55, 482, 105000,
   BusinessUnit_t(2), 48000);
```

➲ Architecture Table

```
INSERT INTO Architect (Oid, Name, Age, SerialNum,
   Salary, Dept, StockOption)
VALUES(Architect_t('l'), 'Leo', 35, 661, 92000,
   BusinessUnit_t(2), 20000);

INSERT INTO Architect (Oid, Name, Age, SerialNum,
   Salary, Dept, StockOption)
VALUES( Architect_t('m'), 'Brian', 7, 882, 112000,
    (SELECT Oid FROM BusinessUnit WHERE name = 'Toy'),
      30000);
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

Reference Type 생성에 대해 살펴봅니다.

13 Reference Type 생성

➲ 각각의 Structured type에 대해, DB2는 이에 상응하는 reference type을 제공합니다. 예를 들어, Person_t type을 생성할 때, REF(Person_t) type을 생성합니다.

➲ Create Table문장의 "With Options"절을 사용하여 Column과 Object간의 연관관계를 정의할 수 있습니다.

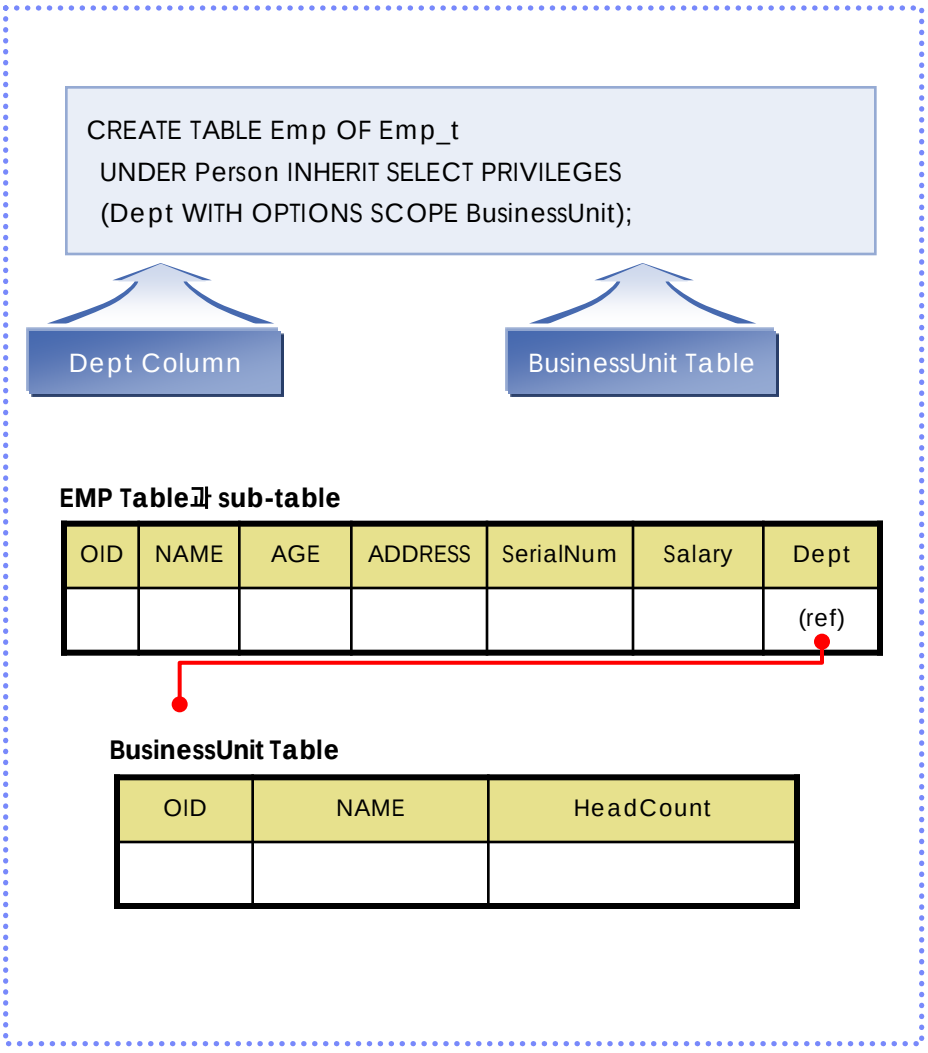


Figure 2503D Reference Type 생성

TOPIC 04-03 사용자 정의 구조화 유형

Point

Self-Reference Relationship 정의에 대해 살펴봅니다.

14 Self-Reference Relationship 정의

➲ Type 생성

```
CREATE TYPE COMPANY_T AS (
  NAME     VARCHAR(30),
  LOCATION VARCHAR(30)
) MODE DB2SQL ;

CREATE TYPE PART_T AS (
  DESCRIPT    VARCHAR(20),
  SUPPLIED_BY REF(COMPANY_T),
  USED_IN     REF(PART_T)
) MODE DB2SQL;
```

➲ Table 생성

```
CREATE TABLE SUPPLIERS OF COMPANY_T (
   REF         IS SUPPNO USER GENERATED);

CREATE TABLE PARTS OF PART_T (
   REF         IS PARTNO USER GENERATED,
   SUPPLIED_BY WITH OPTIONS SCOPE SUPPLIERS,
   USED_IN     WITH OPTIONS SCOPE PARTS);
```

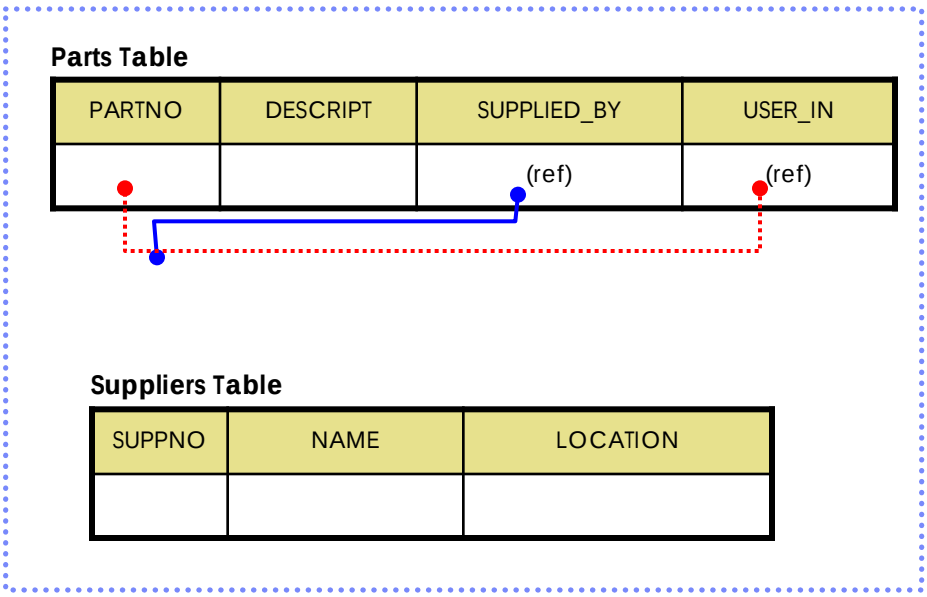


Figure 2503E ••• Self-Reference Relationship 정의

TOPIC 0403 사용자 정의 구조화 유형

Point

RI 와 Scoped Reference 의 차이에 대해 살펴봅니다.

Tip

RI관계를 강제적으로 정의 하려면 Referential Integrity를 정의하여야 한다.

15 RI (Referential Integrity) 와 Scoped Reference 의 차이

➲ Scoped Reference는 Table Object간 관계를 정의하지만 RI와는 다르다. Scope는 Target table에 대한 정보만을 제공한다.

➲ Scoped Reference는 다른 Object에 해당 값이 필요하거나, 강제적으로 제약하지는 않는다.

➲ Type 정의 및 Table 정의

```
CREATE TYPE  Empl_t AS (
  Name VARCHAR(10)
, Mgr REF(Empl_t)
) ref using integer MODE DB2SQL;

CREATE TABLE Empl   OF Empl_t (
  REF IS Oid USER GENERATED);
```

➲ RI 정의

```
ALTER TABLE Empl
  ADD CONSTRAINT pk1 UNIQUE(Oid);

ALTER TABLE Empl
  ADD CONSTRAINT fk1
     FOREIGN KEY(Mgr) REFERENCES Empl (Oid);
```

➲ Record 생성

```
insert into empl (oid,name)
values ( empl_t(1  ), '홍길동'              );

insert into empl (oid,name,mgr)
values ( empl_t(11 ), '김서방'   ,empl_t(1  ) );

insert into empl (oid,name,mgr)
values ( empl_t(111), '김서방111',empl_t(11 ) );

insert into empl (oid,name,mgr)
values ( empl_t(112), '김서방112',empl_t(11 ) );

insert into empl (oid,name,mgr)
values ( empl_t(12 ), '이서방'   ,empl_t(1  ) );

insert into empl (oid,name,mgr)
values ( empl_t(121), '이서방121',empl_t(12 ) );

insert into empl (oid,name,mgr)
values ( empl_t(122), '이서방122',empl_t(12 ) );
```

Empl Table

OID	NAME	MGR
		(ref)

Figure 2503F RI 와 Scoped Reference 차이

TOPIC 0403 사용자 정의 구조화 유형

Point

Typed View에 대해 살펴봅니다.

16 Typed View 정의

"Create View"문장을 사용하여 Typed View를 생성할 수 있습니다. 원하는 Attribute를 만들기 위해서는 Type을 생성할 수 있으며, 이를 바탕으로 View를 정의 할 수 있습니다.

➲ View를 위한 Type정의

```
CREATE TYPE VBUSINESSUNIT_T AS (
      NAME VARCHAR(20)
) REF USING INTEGER MODE DB2SQL;
```

➲ View를 정의

```
CREATE VIEW VBUSINESSUNIT OF VBUSINESSUNIT_T
MODE DB2SQL (REF IS VOID USER GENERATED)
AS SELECT VBUSINESSUNIT_T(INTEGER(OID))
       , NAME
    FROM BUSINESSUNIT;
```

➲ View 조회

```
SELECT * FROM VBUSINESSUNIT;

VOID        NAME
----------- ---------
          1 DEPT1
          2 DEPT2
          3 DEPT3
          4 DEPT4
          5 DEPT5
```

➲ View Hierarchy 생성

```
CREATE TYPE VPERSON_T
AS (NAME VARCHAR(20))
MODE DB2SQL;

CREATE TYPE VEMP_T UNDER VPERSON_T
AS (SALARY INT, DEPT REF(VBUSINESSUNIT_T))
MODE DB2SQL;
```

```
CREATE VIEW VPERSON OF VPERSON_T MODE DB2SQL
     (REF IS VOID USER GENERATED)
AS SELECT VPERSON_T (VARCHAR(OID))
       , NAME
    FROM ONLY(PERSON);

CREATE VIEW VEMP OF VEMP_T MODE DB2SQL
      UNDER VPERSON INHERIT SELECT PRIVILEGES
     (DEPT WITH OPTIONS SCOPE VBUSINESSUNIT)
AS SELECT VEMP_T(VARCHAR(OID))
       , NAME
       , SALARY
       , VBUSINESSUNIT_T(INTEGER(DEPT))
    FROM EMP;
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

조회에 대해 자세히 살펴 봅니다.

Tip

조회 권한이 있으면 해당 Table을 일반 Table조회 하듯이 조회 할 수 있습니다.

17 Typed Table 조회

➲ Person table 조회

```
select name, age from person;
```

18 Dereference Reference 에 대한 조회

Scoped Reference를 조회 하고자 할 때는 "Dereference Operation" 사용합니다.

Dereference Operator : ->
Scoped-reference-expression->column-in-target-typed-table

➲ Table 조회

```
select name
    , salary
    , dept->name
 from emp;
```

```
SELECT P.Descript
    , P.Supplied_by -> Location
 FROM Parts P
 WHERE P.Used_in -> Descript='Wing';
```

19 Built-in Function

```
DEREF           : Scope reference Column명을 통해 Object 확보
TYPE_NAME
TYPE_ID
TYPE_SCHEMA
```

➲ DEREF

```
DEREF ( Scoped-reference-expression )
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

조회에 대해 자세히 살펴 봅니다.

➲ TYPE_* function

```
CREATE TYPE PROJECT_T
AS (PROJID INT, RESPONSIBLE REF(EMP_T))
MODE DB2SQL;

CREATE TABLE PROJECT
OF PROJECT_T (REF IS OID USER GENERATED,
RESPONSIBLE WITH OPTIONS SCOPE EMP);


Insert into project
values ( project_t('A'),10, emp_t('s') );

SELECT OID
    , PROJID
    , RESPONSIBLE->NAME
    , TYPE_NAME  (DEREF(RESPONSIBLE))
    , TYPE_ID    (DEREF(RESPONSIBLE))
    , TYPE_SCHEMA(DEREF(RESPONSIBLE))
FROM PROJECT;
```

20 Query Specification

```
ONLY                    : 특정 type의 Object만 Return
IS OF                   : Type에 대한 유형 제어
OUTER                   : Object가 가질 수 있는 모든 Attribute 표시
GENERATE_UNIQUE()       : 유일한 값을 생성하는 변수
```

➲ ONLY

```
SELECT NAME
  FROM ONLY ( EMP );
```

➲ IS OF

```
 <expression> IS OF ( TYPE_NAME[, ... ] )

SELECT NAME
  FROM EMP E
 WHERE E.AGE > 15
   AND DEREF(E.OID) IS OF
     ( EMP_T, MANAGER_T, ARCHITECT_T);
```

TOPIC 04-03 사용자 정의 구조화 유형

Point

조회에 대해 자세히 살펴 봅니다.

➲ OUTER

```
SELECT TYPE_NAME(DEREF(Oid)) TYPE_NAME
    , AGE
    , NAME
    , SALARY
    , STOCKOPTION
 FROM OUTER(EMP);
```

```
TYPE_NAME   AGE  NAME      SALARY STOCKOPTION
----------- ---- -------- --------- -----------
EMP_T        26 Dennis   30000.00         -
EMP_T        36 Charlie  34000.00         -
EMP_T        46 Bill     44000.00         -
EMP_T        36 Charlie  34000.00         -
EMP_T        46 Bill     44000.00         -
EMP_T        39 Susan    37000.48         -
ARCHITECT_T   7 Brian   112000.00      30000
EMP_T        35 Marie    55000.00         -
```

➲ GENERATE_UNIQUE()

```
INSERT INTO EMP
 (Oid, Name, Age, SerialNum, Salary, Dept)
VALUES
 (emp_t(generate_unique())
  , 'Dennis', 26, 105, 30000,BusinessUnit_t(1));
```

```
select oid, name, age from emp where serialnum=105

OID                         NAME   AGE
--------------------------- ------ ---
x'20060404092229377998000000' Dennis  26
```

DB2 사용자 가이드

for Linux, UNIX and Windows

IBM Software Group
Information Management DB2 FTSS
Information Management Marketing

IBM

한국아이비엠주식회사
서울시 강남구 도곡동 467-12 군인공제회관
고객만족센터 TEL:(02)3781-7114 www.ibm.com/kr