SQL DB2 pour Administrateurs IBMi

Modernisation, développement d'applications et DB2 sous IBM i

Technologies, outils et nouveautés 2013-2014

13 et 14 mai 2014 – IBM Client Center Paris, Bois-Colombes

S23 – SQL pour les administrateurs IBMi

Grégory Jarrige
gjarrige@six-axe.fr

Sommaire

1 Introduction

La base de données DB2 for i a pour réputation (justifiée) de nécessiter une surveillance restreinte.

Mais avec la montée en puissance du "Big Data", et la consommation d'espace disque en augmentation constante qui en résulte, les administrateurs système et bases de données ont de plus en plus besoin de disposer d'informations en temps réel sur l'état des systèmes et des bases qu'ils supervisent.

Au travers du catalogue système, DB2 for i recèle de nombreuses pépites qui répondent aux besoins des administrateurs (accès aux valeurs système, aux PTFs, à la consommation disque, etc.)

Cette session a pour objectif de présenter certaines de ces fonctionnalités, et la manière dont vous pouvez les utiliser au quotidien pour administrer vos bases DB2 for i.

2 Présentation de Six-Axe

Depuis plus de 20 ans, Six-Axe Consultants accompagne ses clients en apportant son expertise métier et sa maîtrise des systèmes d'information pour des missions de conseil, d'assistance à maitrise d'ouvrage, de prestation technique et d'intégration de logiciels spécialisés. L'exercice de ce métier s'articule autour de plusieurs activités :

Expertise technique sur plateforme IBMi :

- Audit et Optimisation des performances systèmes,
- Expertise sur les programmes pour améliorer les performances,
- Installation/Configuration des environnements Zend sur System i,
- Formations au SQL DB2 et au langage RPG en partenariat avec IBM France
- Formation PHP dédiée aux développeurs IBMi, en partenariat avec Zend France et IBM France

Plusieurs activités :

- La délégation de personnel,
- Le développement sous cahier des charges,
- Tierce Maintenance Applicative (TMA)
- Webisation et modernisation d'applications

Site officiel : www.six-axe.fr

3 Présentation de l'auteur

Grégory Jarrige

Consultant open-source chez Six-Axe Consultants depuis janvier 2011.

Développe sur plateforme IBM i (ex. AS/400) depuis 1991

Expert PHP5 certifié par Zend depuis février 2010

Domaines d'expertise :

- PHP 5
- Javascript et HTML5
- SQL DB2 (incluant procédures stockées et SQLRPGLE)
- RPG Free

Assure régulièrement des sessions de formations, et de transfert de compétences, sur les technologies précitées (surtout les 3 premières).

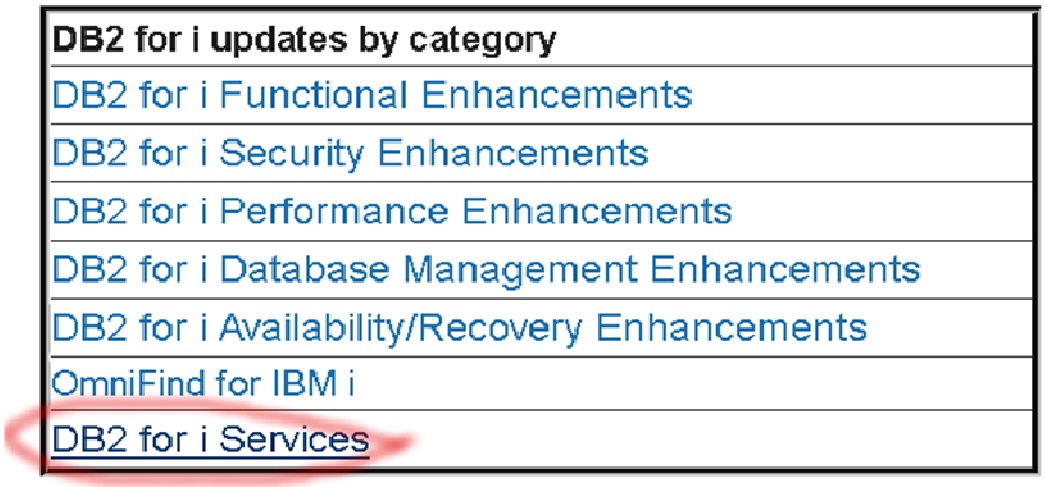
Rédacteur d'un blog dédié à la diffusion de bonnes pratiques autour de PHP et de DB2 : http://www.gregphplab.com

Contributeur régulier du site internet XDocs400 : http://xdocs400.com

4 Préambule

Avec les Technology Refresh (TR6, TR7 et TR8), DB2 for i se positionne comme un auxiliaire des administrateurs systèmes et bases de données, en simplifiant un certain nombre de tâches d'administration.

- Accès en SQL à des fonctions système
- Solution alternative aux commandes CL et APIs
- Nouvelle rubrique dans les Technology Updates :



La documentation officielle pour ces nouveaux services est accessible dans le "IBM Knowledge Center":

http://www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_72/rzajq/rzajqservicessys.htm

Lien vers le Wiki :

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/DB2%20for%20i%20-%20Services

5 DB2 for i Services

5.1 Application Services

5.1.1 QSYS2.QCMDEXC

La procédure stockée QSYS2.QCMDEXC() peut être utilisée pour exécuter différentes commandes systèmes IBMi.

Ce n'est pas vraiment une nouveauté, mais ce qui est nouveau, c'est que - depuis la TR7 - on n'est plus obligé de préciser la longueur de la commande système à exécuter, car la procédure est en mesure de le déterminer d'elle-même.

Deux exemples d'utilisation :

- Ajout d'une bibliothèque dans la "library list" :

```
CALL QSYS2.QCMDEXC('ADDLIBLE PRODLIB2');
```

- La même chose mais, via une "expression" concaténée à la volée :

```
DECLARE V_LIBRARY_NAME VARCHAR(10);
...
SET V_LIBRARY_NAME = 'PRODLIB2';
...
CALL QSYS2/QCMDEXC('ADDLIBLE ' CONCAT V_LIBRARY_NAME);
```

5.1.2 QSYS2.DELIMIT_NAME

Annoncé sur la TR8, mais en réalité déjà disponible sur la TR7, la fonction DELIMIT_NAME renvoie une valeur avec des délimiteurs (guillemets et/ou apostrophes) répondant à différentes problématiques rencontrées par les développeurs SQL.

Le schéma de la fonction est QSYS2 (il est implicite et on n'a pas besoin de le préciser à chaque utilisation).

Le paramètre d'entrée est une chaîne de 128 caractères maximum (en cas de dépassement, la valeur renvoyée est tronquée à cette longueur). La valeur renvoyée est un VARCHAR contenant une chaîne correctement délimitée.

Exemple :

```
SELECT
DELIMIT_NAME('ABC'), -- ABC
        DELIMIT_NAME('ABC') , -- "ABC"
        DELIMIT_NAME('TEST"NAME'), -- "TEST""NAME"
        DELIMIT_NAME('TEST''NAME2'), -- "TEST'NAME2"
        DELIMIT_NAME('NEW') -- "NEW"
FROM SYSIBM.SYSDUMMY1;
```

5.1.3 QYS2.OVERRIDE_TABLE

Il est parfois nécessaire, pour des raisons de performance, sur des applications critiques, d'agir sur le taux de transfert des données d'une table, en jouant sur la taille du buffer utilisé par DB2 pour les transferts de données. On peut réaliser ce type de manipulation en demandant au système d'utiliser un buffer intermédiaire de 256 K, comme dans l'exemple suivant :

```
CALL QCMDEXC ( 'OVRDBF FILE(PRODUCT) TOFILE(GJABASE/PRODUCT) SEQONLY(*YES
*BUF256KB)' ) ;
```

Mais avec l'arrivée de la TR7, on n'est plus obligé de recourir à la commande système OVRDBF, on peut recourir à la procédure stockée DB2 QSYS2/OVERRIDE_TABLE, comme dans les exemples suivants :

```
-- Override sur la table Product de la bibliothèque, en utilisant un buffer
bloqué à 256K
CALL QSYS2.OVERRIDE_TABLE('GJABASE', 'PRODUCT', '*BUF256KB');

-- Suppression de l'override
CALL QSYS2.OVERRIDE_TABLE('GJABASE', 'PRODUCT', 0);
```

A noter : pour l'override, un nombre d'octets spécifique peut être fourni, ou on peut recourir aux valeurs spéciales prédéfinies suivantes : *BUF32KB, *BUF64KB, *BUF128KB, *BUF256KB.

5.2 TCP/IP Services

5.2.1 SYSIBMADM.ENV_SYS_INFO

La vue DB2 ENV_SYS_INFO permet de récupérer via une simple requête SQL différentes informations qui peuvent intéresser tout le monde.

```
SELECT * FROM SYSIBMADM.ENV_SYS_INFO ;
```

OS_NAME	OS_VERSION	OS_RELEASE	HOST_NAME	TOTAL_CPUS	CONFIGURED_CPUS	TOTAL_MEMORY
IBM i	7	1	XXX six-axe fr	1	1	4096

On peut par exemple se servir des valeurs de OS_VERSION et OS_RELEASE pour savoir si le code s'exécute sur un serveur en V7R1, ce qui autorise à utiliser certaines instructions SQL comme par exemple MERGE (qui est franchement géniale pour les opérations de mise à jour).

```
MERGE INTO My_LIBRARY.testmerge A
 USING (SELECT * FROM SYSIBM.SYSDUMMY1) B
 ON A.macle = 'CLE1'
 WHEN MATCHED THEN
 UPDATE SET
  a.codea = 'A1' ,
  a.coden = a.coden + 1
 WHEN NOT MATCHED THEN
 INSERT ( a.macle , a.codea , a.coden )
 VALUES( 'CLE1' , 'A1' , 1 )
;
```

Si on détecte que l'on est dans une version antérieure à la V7R1, alors il faut utiliser une solution de rechange, plus laborieuse à écrire certes, mais qui permettra à votre application de fonctionner sur différentes versions d'OS de manière optimale.

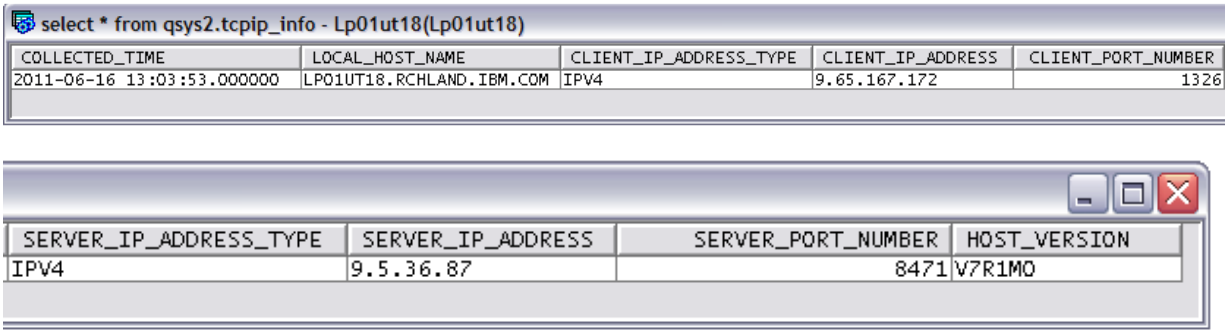
A noter que dans le MERGE que j'ai utilisé, la requête déclarée dans le paramètre USING est une requête sur la table pivot SYSDUMMY1. Ce n'est pas la manière la plus courante d'utiliser MERGE (elle est d'ailleurs rarement présentée dans les

documentations), mais elle est très pratique quand les données à mettre à jour proviennent de variables programmes (variables de programme RPG, ou variables de procédure stockée DB2).

5.2.2 QSYS2.TCPIP_INFO

La vue TCPIP_INFO fournit différentes informations intéressantes sur la connexion au serveur courant.

```
select * from QSYS2.TCPIP_INFO;
```



select * from qsys2.tcpip_info - Lp01ut18(Lp01ut18)

COLLECTED_TIME	LOCAL_HOST_NAME	CLIENT_IP_ADDRESS_TYPE	CLIENT_IP_ADDRESS	CLIENT_PORT_NUMBER
2011-06-16 13:03:53.000000	LP01UT18.RCHLAND.IBM.COM	IPV4	9.65.167.172	1326

SERVER_IP_ADDRESS_TYPE	SERVER_IP_ADDRESS	SERVER_PORT_NUMBER	HOST_VERSION
IPV4	9.5.36.87	8471	V7R1M0

5.3 PTF Services

5.3.1 QSYS2.PTF_INFO

La vue QSYS2.PTF_INFO fournit de précieuses informations aux administrateurs systèmes :

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	PTF_PRODUCT_ID	LICPGM	VARCHAR	7
2	PTF_PRODUCT_OPTION	PRODOPT	VARCHAR	6
3	PTF_PRODUCT_RELEASE_LEVEL	PRODRLS	VARCHAR	6
4	PTF_PRODUCT_DESCRIPTION	PRODDESC	VARCHAR	132
5	PTF_IDENTIFIER	PTFID	VARCHAR	7
6	PTF_RELEASE_LEVEL	PTFRLS	VARCHAR	6
7	PTF_PRODUCT_LOAD	PRODLOAD	VARCHAR	4
8	PTF_LOADED_STATUS	LOADSTAT	VARCHAR	19
9	PTF_SAVE_FILE	SAVF	VARCHAR	3
10	PTF_COVER_LETTER	COVER	VARCHAR	3
11	PTF_ON_ORDER	ONORD	VARCHAR	3
12	PTF_IPL_ACTION	IPLACT	VARCHAR	19
13	PTF_ACTION_PENDING	ACTPEND	VARCHAR	3
14	PTF_ACTION_REQUIRED	ACTREQ	VARCHAR	12
15	PTF_IPL_REQUIRED	IPLREQ	VARCHAR	9
16	PTF_IS_RELEASED	RELEASED	VARCHAR	3
17	PTF_MINIMUM_LEVEL	MINLVL	VARCHAR	2
18	PTF_MAXIMUM_LEVEL	MAXLVL	VARCHAR	2
19	PTF_STATUS_TIMESTAMP	STATTIME	TIMESTAMP	10
20	PTF_SUPERCEDED_BY_PTF	SUPERCEDE	VARCHAR	7
21	PTF_CREATION_TIMESTAMP	CRTTIME	TIMESTAMP	10
22	PTF_TECHNOLOGY_REFRESH_PTF	TRPTF	VARCHAR	3

```
SELECT * FROM QSYS2.PTF_INFO ;
```

PTF_PRODUC...	PTF_PRO...	PTF_PRO...	PTF_PRODUCT_DESCRIPTION	PTF_IDENTIFIER	PTF_RELEASE_LEVEL	PTF_PRODUCT_LOAD
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF06003	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47854	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47855	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47856	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47857	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47858	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47869	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47870	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47871	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47872	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47873	V7R1M0	5050
5770999	*BASE	V7R1M0	Microcode sous licence ...	MF47874	V7R1M0	5050

Exemples :

- trouver toutes les PTF qui seront impactées par le prochain IPL :

```
SELECT PTF_IDENTIFIER, PTF_IPL_ACTION, A.*
  FROM QSYS2.PTF_INFO A
  WHERE PTF_IPL_ACTION <> 'NONE'
```

- trouver les PTF chargées mais non encore appliquées :

```
SELECT PTF_IDENTIFIER, PTF_IPL_REQUIRED, A.*
  FROM QSYS2.PTF_INFO A
  WHERE PTF_LOADED_STATUS = 'LOADED'
  ORDER BY PTF_PRODUCT_ID
```

5.3.2 QSYS2.GROUP_PTF_INFO

La vue GROUP_PTF_INFO contient des informations sur les PTF de groupe pour le serveur.
L'API des PTF de Groupes (QpzListPtfGroups) API est utilisée pour récupérer ces informations.

L'information retournée est similaire à celle disponible sur la commande WRKPTFGRP.
Le tableau suivant décrit les colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	COLLECTED_TIME	COLLE00001	TIMESTAMP	10
2	PTF_GROUP_NAME	PTF_G00001	VARCHAR	60
3	PTF_GROUP_DESCRIPTION	PTF_G00002	VARCHAR	100
4	PTF_GROUP_LEVEL	PTF_G00003	INTEGER	9
5	PTF_GROUP_TARGET_RELEASE	PTF_G00004	VARCHAR	6
6	PTF_GROUP_STATUS	PTF_G00005	VARCHAR	20

```
SELECT * FROM QSYS2.GROUP_PTF_INFO ;
```

COLLECTED_TIME	PTF_GROUP_N...	PTF_GROUP_DESCRIPTION	PTF_GROUP_LE...	PTF_GRO...	PTF_GROUP_ST
2014-05-09 12:28:23.570065	SF99145 ...	PERFORMANCE TOOLS ...	6	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99145 ...	PERFORMANCE TOOLS ...	5	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99363 ...	WEBSPHERE APP SERVER V7.0 ...	14	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99363 ...	WEBSPHERE APP SERVER V7.0 ...	13	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99366 ...	PRINT PTFS ...	9	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99366 ...	PRINT PTFS ...	8	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99367 ...	TCP/IP PTF ...	8	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99368 ...	IBM HTTP SERVER FOR I ...	27	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99368 ...	IBM HTTP SERVER FOR I ...	26	V7R1M0	INSTALLED
2014-05-09 12:28:23.570065	SF99572 ...	JAVA ...	16	V7R1M0	INSTALLED

Exemple : déterminer le niveau de la dernière cumulative de PTF installée sur le système

```
SELECT MAX(PTF_GROUP_LEVEL) AS CUM_LEVEL
  FROM QSYS2.GROUP_PTF_INFO
  WHERE PTF_GROUP_NAME IN ('SF99610','SF99710')
        AND PTF_GROUP_STATUS = 'INSTALLED' ;
```

5.4 Security Services

5.4.1 QSYS2.USER_INFO

La vue USER_INFO contient des informations à propos des profils utilisateurs :
Le tableau suivant fournit le détail des 66 colonnes renvoyées par la vue. Le schéma est QSYS2.

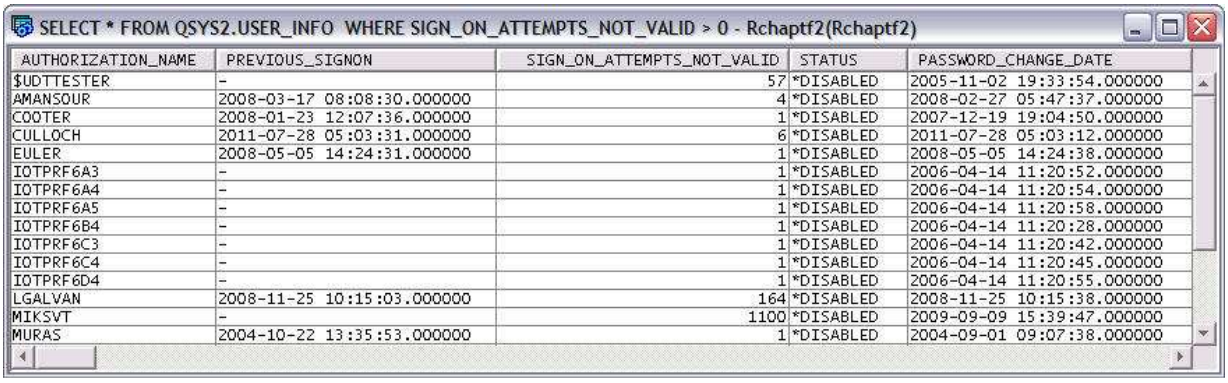
N°	Nom de colonne (long)	Nom court	Type	Longueur
1	AUTHORIZATION_NAME	USER_NAME	VARCHAR	10
2	PREVIOUS_SIGNON	PRVSIGNON	TIMESTAMP	10
3	SIGN_ON_ATTEMPTS_NOT_VALID	SIGNONINV	INTEGER	9
4	STATUS	STATUS	VARCHAR	10
5	PASSWORD_CHANGE_DATE	PWDCHGDAT	TIMESTAMP	10
6	NO_PASSWORD_INDICATOR	NOPWD	VARCHAR	3
7	PASSWORD_EXPIRATION_INTERVAL	PWDEXPITV	SMALLINT	4
8	DATE_PASSWORD_EXPIRES	PWDEXPDAT	TIMESTAMP	10
9	DAYS_UNTIL_PASSWORD_EXPIRES	PWDDAYSEXP	INTEGER	9
10	SET_PASSWORD_TO_EXPIRE	PWDEXP	VARCHAR	3
11	USER_CLASS_NAME	USRCLS	VARCHAR	10
12	SPECIAL_AUTHORITIES	SPCAUT	VARCHAR	88
13	GROUP_PROFILE_NAME	GRPPRF	VARCHAR	10
14	OWNER	OWNER	VARCHAR	10
15	GROUP_AUTHORITY	GRPAUT	VARCHAR	10
16	ASSISTANCE_LEVEL	ASTLVL	VARCHAR	10
17	CURRENT_LIBRARY_NAME	CURLIB	VARCHAR	10
18	INITIAL_MENU_NAME	INLMNU	VARCHAR	10
19	INITIAL_MENU_LIBRARY_NAME	INLMNULIB	VARCHAR	10
20	INITIAL_PROGRAM_NAME	INITPGM	VARCHAR	10
21	INITIAL_PROGRAM_LIBRARY_NAME	INITPGMLIB	VARCHAR	10
22	LIMIT_CAPABILITIES	LMTCPB	VARCHAR	10
23	TEXT_DESCRIPTION	TEXT	VARCHAR	50
24	DISPLAY_SIGNON_INFORMATION	DSPSGNINF	VARCHAR	10
25	LIMIT_DEVICE_SESSIONS	LMTDEVSSN	VARCHAR	10
26	KEYBOARD_BUFFERING	KBDBUF	VARCHAR	10
27	MAXIMUM_ALLOWED_STORAGE	MAXSTGLRG	BIGINT	18
28	STORAGE_USED	STGUSED	BIGINT	18

29	HIGHEST_SCHEDULING_PRIORITY	PTYLMT	CHAR	1
30	JOB_DESCRIPTION_NAME	JOBD	VARCHAR	10
31	JOB_DESCRIPTION_LIBRARY_NAME	JOBDLIB	VARCHAR	10
32	ACCOUNTING_CODE	ACGCDE	VARCHAR	15
33	MESSAGE_QUEUE_NAME	MSGQ	VARCHAR	10
34	MESSAGE_QUEUE_LIBRARY_NAME	MSGQLIB	VARCHAR	10
35	MESSAGE_QUEUE_DELIVERY_METHOD	DLVRY	VARCHAR	10
36	MESSAGE_QUEUE_SEVERITY	SEV	SMALLINT	4
37	OUTPUT_QUEUE_NAME	OUTQ	VARCHAR	10
38	OUTPUT_QUEUE_LIBRARY_NAME	OUTQLIB	VARCHAR	10
39	PRINT_DEVICE	PRTDEV	VARCHAR	10
40	SPECIAL_ENVIRONMENT	SPCENV	VARCHAR	10
41	ATTENTION_KEY_HANDLING_PROGRAM_NAME	ATNPGM	VARCHAR	10
42	ATTENTION_KEY_HANDLING_PROGRAM_LIBRARY_NAME	ATNPGMLIB	VARCHAR	10
43	LANGUAGE_ID	LANGID	VARCHAR	10
44	COUNTRY_OR_REGION_ID	CNTRYID	VARCHAR	10
45	CHARACTER_CODE_SET_ID	CCSID	VARCHAR	6
46	USER_OPTIONS	USROPT	VARCHAR	77
47	SORT_SEQUENCE_TABLE_NAME	SRTSEQ	VARCHAR	10
48	SORT_SEQUENCE_TABLE_LIBRARY_NAME	SRTSEQLIB	VARCHAR	10
49	OBJECT_AUDITING_VALUE	OBJAUD	VARCHAR	10
50	USER_ACTION_AUDIT_LEVEL	AUDLVL	VARCHAR	341
51	GROUP_AUTHORITY_TYPE	GRPAUTTYP	VARCHAR	10
52	USER_ID_NUMBER	UID	BIGINT	18
53	GROUP_ID_NUMBER	GID	BIGINT	18
54	LOCALE_JOB_ATTRIBUTES	SETJOBATR	VARCHAR	88
55	GROUP_MEMBER_INDICATOR	GRPMBR	VARCHAR	3
56	DIGITAL_CERTIFICATE_INDICATOR	DCIND	VARCHAR	3
57	CHARACTER_IDENTIFIER_CONTROL	CHRIDCTL	VARCHAR	10
58	LOCAL_PASSWORD_MANAGEMENT	LCLPWDMGT	VARCHAR	3
59	BLOCK_PASSWORD_CHANGE	PWDCHGBLK	VARCHAR	10
60	USER_ENTITLEMENT_REQUIRED	ENTITLERQD	VARCHAR	3
61	USER_EXPIRATION_INTERVAL	USREXPITV	SMALLINT	4
62	USER_EXPIRATION_DATE	USREXPDATE	TIMESTAMP	10
63	USER_EXPIRATION_ACTION	ACTION	VARCHAR	8
64	HOME_DIRECTORY	HOMEDIR	VARGRAPHIC	1024
65	LOCALE_PATH_NAME	LOCALE	VARGRAPHIC	1024
66	USER_DEFAULT_PASSWORD	DFTPWD	VARCHAR	3

Attention :
- seuls les objets de type *USRPRF sur lesquels l'utilisateur dispose de l'autorité *READ sont renvoyés par la vue.
- les valeurs renvoyées correspondent aux informations fournies par l'API QSYRUSRI.

Exemple : Déterminer quels utilisateurs ont rencontré des problèmes de "SIGN ON".

```
SELECT * FROM QSYS2.USER_INFO
  WHERE SIGN_ON_ATTEMPTS_NOT_VALID > 0 ;
```


SELECT * FROM QSYS2.USER_INFO WHERE SIGN_ON_ATTEMPTS_NOT_VALID > 0 - Rchaptf2(Rchaptf2)

AUTHORIZATION_NAME	PREVIOUS_SIGNON	SIGN_ON_ATTEMPTS_NOT_VALID	STATUS	PASSWORD_CHANGE_DATE
$UDTTESTER	-	57	*DISABLED	2005-11-02 19:33:54.000000
AMANSOUR	2008-03-17 08:08:30.000000	4	*DISABLED	2008-02-27 05:47:37.000000
COOTER	2008-01-23 12:07:36.000000	1	*DISABLED	2007-12-19 19:04:50.000000
CULLOCH	2011-07-28 05:03:31.000000	6	*DISABLED	2011-07-28 05:03:12.000000
EULER	2008-05-05 14:24:31.000000	1	*DISABLED	2008-05-05 14:24:38.000000
IOTPRF6A3	-	1	*DISABLED	2006-04-14 11:20:52.000000
IOTPRF6A4	-	1	*DISABLED	2006-04-14 11:20:54.000000
IOTPRF6A5	-	1	*DISABLED	2006-04-14 11:20:58.000000
IOTPRF6B4	-	1	*DISABLED	2006-04-14 11:20:28.000000
IOTPRF6C3	-	1	*DISABLED	2006-04-14 11:20:42.000000
IOTPRF6C4	-	1	*DISABLED	2006-04-14 11:20:45.000000
IOTPRF6D4	-	1	*DISABLED	2006-04-14 11:20:55.000000
LGALVAN	2008-11-25 10:15:03.000000	164	*DISABLED	2008-11-25 10:15:38.000000
MIKSVT	-	1100	*DISABLED	2009-09-09 15:39:47.000000
MURAS	2004-10-22 13:35:53.000000	1	*DISABLED	2004-09-01 09:07:38.000000

5.4.2 QSYS2.FUNCTION_INFO

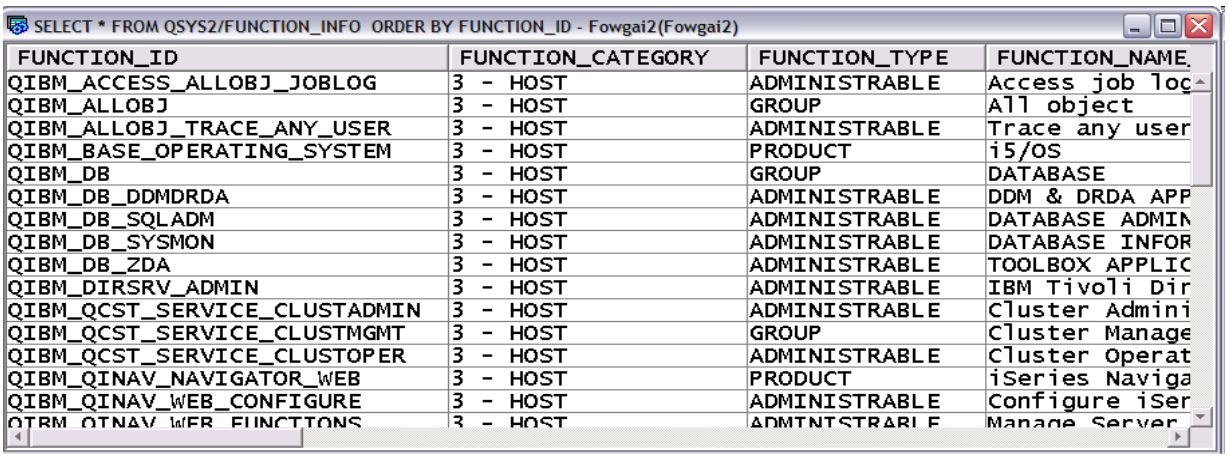
La vue FUNCTION_INFO fournit un équivalent SQL à l'API QSYRTVFI (QsyRetrieveFunctionInformation).

Le tableau suivant fournit le détail des colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	FUNCTION_ID	FCNID	VARCHAR	30
2	FUNCTION_CATEGORY	FCNCAT	VARCHAR	10
3	FUNCTION_TYPE	FCNTYP	VARCHAR	13
4	FUNCTION_NAME_MESSAGE_TEXT	FCNMSGTXT	VARGRAPHIC	330
5	FUNCTION_NAME	FCNNAM	VARGRAPHIC	330
6	FUNCTION_DESCRIPTION_MESSAGE_TEXT	FCNDESCTXT	VARGRAPHIC	330
7	FUNCTION_DESCRIPTION	FCNDESC	VARGRAPHIC	330
8	FUNCTION_PRODUCT_ID	FCNPRDID	VARCHAR	30
9	FUNCTION_GROUP_ID	FCNGRPID	VARCHAR	30
10	DEFAULT_USAGE	DFTUSG	VARCHAR	7
11	ALLOBJ_INDICATOR	ALLOBJ	VARCHAR	8
12	USAGE_INFORMATION_INDICATOR	USGINFO	VARCHAR	3

Exemple:

```
SELECT * FROM QSYS2.FUNCTION_INFO ORDER BY FUNCTION_ID ;
```


SELECT * FROM QSYS2/FUNCTION_INFO ORDER BY FUNCTION_ID - Fowgai2(Fowgai2)

FUNCTION_ID	FUNCTION_CATEGORY	FUNCTION_TYPE	FUNCTION_NAME
QIBM_ACCESS_ALLOBJ_JOBLOG	3 - HOST	ADMINISTRABLE	Access job log
QIBM_ALLOBJ	3 - HOST	GROUP	All object
QIBM_ALLOBJ_TRACE_ANY_USER	3 - HOST	ADMINISTRABLE	Trace any user
QIBM_BASE_OPERATING_SYSTEM	3 - HOST	PRODUCT	i5/OS
QIBM_DB	3 - HOST	GROUP	DATABASE
QIBM_DB_DDMDRDA	3 - HOST	ADMINISTRABLE	DDM & DRDA APP
QIBM_DB_SQLADM	3 - HOST	ADMINISTRABLE	DATABASE ADMIN
QIBM_DB_SYSMON	3 - HOST	ADMINISTRABLE	DATABASE INFOR
QIBM_DB_ZDA	3 - HOST	ADMINISTRABLE	TOOLBOX APPLIC
QIBM_DIRSRV_ADMIN	3 - HOST	ADMINISTRABLE	IBM Tivoli Dir
QIBM_QCST_SERVICE_CLUSTADMIN	3 - HOST	ADMINISTRABLE	Cluster Admini
QIBM_QCST_SERVICE_CLUSTMGMT	3 - HOST	GROUP	Cluster Manage
QIBM_QCST_SERVICE_CLUSTOPER	3 - HOST	ADMINISTRABLE	Cluster Operat
QIBM_QINAV_NAVIGATOR_WEB	3 - HOST	PRODUCT	iSeries Naviga
QIBM_QINAV_WEB_CONFIGURE	3 - HOST	ADMINISTRABLE	Configure iSer
QIBM_QINAV_WEB_FUNCTIONS	3 - HOST	ADMINISTRABLE	Manage Server

5.4.3 QSYS2.FUNCTION_USAGE

La vue FUNCTION_USAGE fournit un équivalent SQL de l'API QSYRTFUI (QsyRetrieveFunctionUsageInfo).

Seuls les utilisateurs ayant l'autorité *SECADM peuvent examiner les informations renvoyées par cette vue.
Les utilisateurs ne disposant pas de cette autorité recevront un SQLCODE -443.

Le tableau suivant fournit le détail des colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	FUNCTION_ID	FCNID	VARCHAR	30
2	USER_NAME	USER_NAME	VARCHAR	10
3	USAGE	USAGE	VARCHAR	7
4	USER_TYPE	USER_TYPE	VARCHAR	5

Exemple :

Déterminer quelles fonctions ont fait l'objet de modifications de droits (GRANT ou REVOKE) :

```
SELECT * FROM QSYS2.FUNCTION_USAGE ORDER BY FUNCTION_ID, USER_NAME ;
```

FUNCTION_ID	USER_NAME	USAGE	USER_TYPE
QIBM_QSY_SYSTEM_CERT_STORE	QDIRSRV	ALLOWED	USER
QIBM_QSY_SYSTEM_CERT_STORE	QTCP	ALLOWED	USER
QIBM_QSY_SYSTEM_CERT_STORE	QYPSJSVR	ALLOWED	USER
QIBM_Q1A_ARC	QSECOFR	ALLOWED	USER
QIBM_Q1A_ARC_CTLG_BRM.ARCGRP	QSECOFR	ALLOWED	USER
QIBM_Q1A_ARC_PCY	QSECOFR	ALLOWED	USER
QIBM_Q1A_BKU	QSECOFR	ALLOWED	USER

5.4.4 SQL_CHECK_AUTHORITY

La fonction scalaire SQL_CHECK_AUTHORITY pemet de contrôler si l'utilisateur courant est habilité à effectuer des requêtes sur un objet donné.

Les paramètres d'appel sont :

- nom de la bibliothèque (schéma)
- nom de l'objet DB2

La valeur renvoyée est de type SMALLINT, sa signification est la suivante :

- 0 : l'utilisateur n'est pas autorisé à "requêter" cet objet, ou l'objet n'est pas de type *FILE, ou l'objet n'existe pas
- 1 : l'utilisateur est autorisé à effectuer des requêtes sur cet objet

Exemple :

```
SELECT SQL_CHECK_AUTHORITY ('QSYS2' , 'FUNCTION_USAGE') FROM
SYSIBM.SYSDUMMY1 ; -- 0

SELECT SQL_CHECK_AUTHORITY ('GJABASE' , 'CONTRAT_TB') FROM
SYSIBM.SYSDUMMY1 ; -- 1
```

5.5 Work Management Services

5.5.1 QSYS2.SYSTEM_VALUE_INFO

La vue SYSTEM_VALUE_INFO renvoie différentes valeurs systèmes. C'est l'équivalent SQL de l'API "Retrieve System Values" (QWCRSVAL).

Les autorités spéciales *ALLOBJ ou *AUDIT sont nécessaires pour pouvoir récupérer le contenu des valeurs systèmes suivantes : QAUDCTL, QAUDENDACN, QAUDFRCLVL, QAUDLVL, QAUDLVL2, et QCRTOBJAUD.

Les colonnes sélectionnées pour lesquels l'utilisateur n'a pas les autorisations adéquates contiendront en sortie '*NOTAVL' ou -1.

Exemple : Examiner les valeurs systèmes de type "maximum".

```
SELECT * FROM SYSTEM_VALUE_INFO
WHERE SYSTEM_VALUE_NAME LIKE '%MAX%' ;
```

SYSTEM_VALUE_NAME	CURRENT_NUMERIC_VALUE	CURRENT_CHARACTER_VALUE
QMAXACTLVL	32767	-
QMAXSIGN	-	000003
QPWDMAXLEN	10	-
QMAXSGNACN	-	3
QMAXJOB	163520	-
QMAXSPLF	9999	-

5.5.2 QSYS2.GET_JOB_INFO

La fonction GET_JOB_INFO est une "table function", c'est à dire une fonction renvoyant une structure équivalente à une table. Cette structure contient ici une seule ligne renvoyant des informations relatives au travail dont l'identifiant a été transmis à la fonction.

Le schéma de la fonction est QSYS2.

Le paramètre d'entrée a une structure bien connue des utilisateurs IBMi, comme le montre l'exemple suivant :

Envoi des informations relatives au travail suivant : 816516/GJA/QPADEV0006.

```
SELECT * FROM TABLE(QSYS2.GET_JOB_INFO('816516/GJA/QPADEV0006')) A;
```

V_JOB_STATUS	V_ACTIVE_JOB_STATUS	V_RUN_PRIORITY	V_SBS_NAME	V_CPU_USED	V_TEMP_STORAGE_USED_...	V_AUX_IO_REQUESTED	V_PAGE_FAULTS	V.
*ACTIVE	DSPW	20	QINTER	13	3	95	74	-

Pour pouvoir utiliser cette fonction, l'appelant doit disposer au minimum de l'autorité spéciale *JOBCTL, ou il doit être autorisé à utiliser les fonctions systèmes QIBM_DB_SQLADM, ou QIBM_DB_SYSMON.

5.6 Storage Services

5.6.1 QSYS2.SYSDISKSTAT

La vue SYSDISKSTAT contient les informations relatives aux disques.

Le tableau suivant fournit le détail des colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	ASP_NUMBER	ASP_NUMBER	SMALLINT	4
2	DISK_TYPE	DISK_TYPE	VARCHAR	4
3	DISK_MODEL	DISK_MODEL	VARCHAR	4
4	UNIT_NUMBER	UNITNBR	SMALLINT	4
5	UNIT_TYPE	UNIT_TYPE	SMALLINT	4
6	UNIT_STORAGE_CAPACITY	UNITSCAP	BIGINT	18
7	UNIT_SPACE_AVAILABLE	UNITSPACE	BIGINT	18
8	PERCENT_USED	PERCENTUSE	DECIMAL	7
9	UNIT_MEDIA_CAPACITY	UNITMCAP	BIGINT	18
10	LOGICAL_MIRRORED_PAIR_STATUS	MIRRORPS	CHAR	1
11	MIRRORED_UNIT_STATUS	MIRRORUS	CHAR	1

Exemple :

```
SELECT * FROM QSYS2.SYSDISKSTAT
```

ASP_N...	DISK_...	DISK_...	UNIT_N...	UNIT_...	UNIT_STORAGE_CAPACI...	UNIT_SPACE_AVAILAB...	PERCENT_US...
1	4327	0070	1	0	70564970496	23976001536	66.022
1	4327	0078	2	0	35282485248	11987042304	66.025
1	4327	0078	3	0	35282485248	11981381632	66.041
1	4327	0078	4	0	35282485248	11986284544	66.027
1	4327	0070	5	0	70564970496	23976603648	66.021

Autre exemple : Renvoie des informations pour toutes les unités SSD.

```
SELECT * FROM QSYS2.SYSDISKSTAT WHERE UNIT_TYPE = 1
```

5.6.2 QSYS2.USER_STORAGE

La vue USER_STORAGE renvoie le pourcentage d'utilisation des ressources disques par profil utilisateur. C'est l'équivalent de l'API QSYRUSRI (Retrieve User Information).

Le tableau suivant fournit le détail des 66 colonnes renvoyées par la vue. Le schéma est QSYS2.

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	AUTHORIZATION_NAME	USER_NAME	VARCHAR	10
2	ASPGRP	ASPGRP	VARCHAR	10
3	MAXIMUM_STORAGE_ALLOWED	MAXSTG	BIGINT	18
4	STORAGE_USED	STGUSED	BIGINT	18

Vous devez disposer de l'autorité *READ sur les profils utilisateurs où la vue ne vous renverra aucune information.
Les données sont fournies par SYSBAS, IASP et profil utilisateur.

Exemple :

```
SELECT * FROM QSYS2.USER_STORAGE
   WHERE USER_NAME = 'GJA';
```

AUTHORIZATION_NAME	ASPGRP	MAXIMUM_STORAGE_ALLOWED	STORAGE_USED
GJA	*SYSBAS	-	1747372

5.7 Journal Services

5.7.1 QSYS2.DISPLAY_JOURNAL

L'affichage des entrées d'un journal via une interface graphique nécessitait jusqu'ici l'utilisation d'API. C'était contraignant, et généralement peu performant.

L'exploitation des entrées de journaux est intéressante pour les administrateurs, car elle leur permet de traquer différents types de problèmes (comme des manipulations de données non conformes aux spécifications des applications utilisées).

La fonction QSYS2/Display_Journal est une nouvelle "table function" permettant à l'utilisateur de visualiser les entrées dans un journal, en exécutant une simple requête SQL.

Exemple 1: afficher toutes les entrées du récepteur courant pour le journal MJATST/QSQJRN.

```
select * from table (
Display_Journal(
'MJATST', 'QSQJRN', -- Journal library and name
'', '', -- Receiver library and name
CAST(null as TIMESTAMP), -- Starting timestamp
CAST(null as DECIMAL(21,0)), -- Starting sequence number
'', -- Journal codes
'', -- Journal entries
'','','','', -- Object library, Object name, Object type, Object member
'', -- User
'', -- Job
'' -- Program

) ) as x;
```

Exemple 2 : trouver tous les changements effectués par SUPERUSER à l'intérieur de la table PRODDATA/SALES

```
select journal_code, journal_entry_type, object, object_type, X.* from table (
QSYS2.Display_Journal(
'PRODDATA', 'QSQJRN', -- Journal library and name
'', '', -- Receiver library and name
CAST(null as TIMESTAMP), -- Starting timestamp
CAST(null as DECIMAL(21,0)), -- Starting sequence number
'', -- Journal codes
'', -- Journal entries
'PRODDATA','SALES','*FILE','SALES', -- Object library, Object name, Object type,
Object member
'', -- User
'', -- Job
'' -- Program
) ) as x
WHERE journal_entry_type in ('DL', 'PT','PX', 'UP') AND "CURRENT_USER" =
'SUPERUSER'
order by entry_timestamp desc
```

Pour de plus amples sur les journaux et leurs récepteurs, reportez-vous à la documentation de l'API QjoRetrieveJournalEntries API dans l'infocenter d'IBM.

5.8 Object Services

5.8.1 QSYS2.OBJECT_STATISTICS

La fonction table OBJECT_STATISTICS renvoie un certain nombre d'informations sur les objets d'une liste.

Exemple : renvoi de tous les objets de type *FILE de la bibliothèque GJABASE

```
SELECT OBJNAME, OBJTYPE, OBJOWNER, OBJDEFINER,
OBJCREATED, OBJSIZE, OBJTEXT, OBJLONGNAME,
LAST_USED_TIMESTAMP, DAYS_USED_COUNT,
LAST_RESET_TIMESTAMP, IASP_NUMBER, OBJATTRIBUTE

FROM TABLE (QSYS2.OBJECT_STATISTICS('GJABASE ','*FILE') ) AS X ;
```

ou, strictement équivalent d'un point de vue fonctionnel :

```
SELECT OBJNAME, OBJTYPE, OBJOWNER, OBJDEFINER,
OBJCREATED, OBJSIZE, OBJTEXT, OBJLONGNAME,
LAST_USED_TIMESTAMP, DAYS_USED_COUNT,
LAST_RESET_TIMESTAMP, IASP_NUMBER, OBJATTRIBUTE

FROM TABLE (QSYS2.OBJECT_STATISTICS('GJABASE ','*ALL') ) AS X

WHERE X.OBJTYPE = '*FILE' ;
```

OBJNAME	OBJT...	OBJOW...	OBJDE...	OBJCREATED	OBJSIZE	OBJTEXT
CONTRAT_TB	*FILE	GJA	GJA	2012-05-24 14:...	249856	Table des con
CONTR00001	*FILE	GJA	GJA	2012-05-24 14:...	184320	-
CONTR00002	*FILE	GJA	GJA	2012-05-24 14:...	184320	-
CONTR00003	*FILE	GJA	GJA	2012-05-24 14:...	184320	-
EXCEPTION2	*FILE	GJA	GJA	2011-05-18 14:...	102400	-
IMPENG	*FILE	GJA	GJA	2012-12-13 10:...	40960	-
IMPFRA	*FILE	GJA	GJA	2012-12-13 11:...	40960	-

Autres exemples :

- trouver les journaux contenus dans la bibliothèque MJATST :

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS('MJATST ','JRN') ) AS X
```

ou

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS('MJATST ','*JRN') ) AS X
```

- trouver les journaux et récepteurs de journaux dans la bibliothèque MJATST.

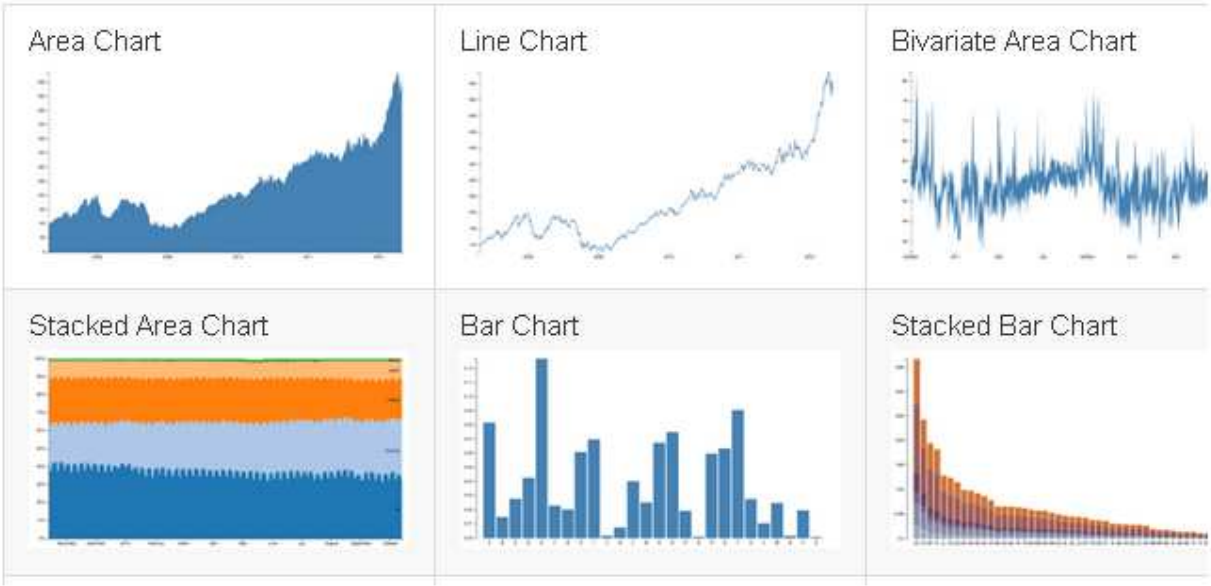
```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS('MJATST ','JRN JRNRCV')
) AS X
```

ou

```
SELECT * FROM TABLE (QSYS2.OBJECT_STATISTICS('MJATST ','*JRN *JRNRCV')
) AS X
```

Grâce à cette fonction, on peut envisager de prendre des clichés périodiques de l'état des bases de données, et éventuellement surveiller leur évolution, sous forme de tableaux HTML, ou de tableaux de bord plus sophistiqués (graphiques), avec des solutions open-sources comme par exemple le projet open source D3 (http://d3js.org).

Exemple de graphiques pouvant être mis en oeuvre facilement avec D3 :
https://github.com/mbostock/d3/wiki/Gallery#basic-charts



On notera que la fonction OBJECT_STATISTICS n'est pas la seule manière de surveiller le contenu des tables. On peut aussi s'appuyer sur la table système QSYS2.SYSTABLESTAT qui permet d'obtenir en temps réel le nombre de lignes de chaque table d'une bibliothèque, ainsi que le nombre de lignes supprimées (pour les REORG notamment), et pas mal d'autres informations (cf. tableau ci-dessous) :

N°	Nom de colonne (long)	Nom court	Type	Longueur
1	TABLE_SCHEMA	TABSCHEMA	VARCHAR	128
2	TABLE_NAME	TABNAME	VARCHAR	128
3	PARTITION_TYPE	PARTTYPE	CHAR	1
4	NUMBER_PARTITIONS	NBRPARTS	INTEGER	9
5	NUMBER_DISTRIBUTED_PARTITIONS	DSTPARTS	INTEGER	9
6	NUMBER_ROWS	CARD	BIGINT	18
7	NUMBER_ROW_PAGES	NPAGES	BIGINT	18
8	NUMBER_PAGES	FPAGES	BIGINT	18
9	OVERFLOW	OVERFLOW	BIGINT	18
10	CLUSTERED	CLUSTERED	CHAR	1
11	ACTIVE_BLOCKS	ACTBLOCKS	BIGINT	18
12	AVGCOMPRESSEDROWSIZE	ACROWSIZE	BIGINT	18
13	AVGROWCOMPRESSIONRATIO	ACROWRATIO	FLOAT	29

14	AVGROWSIZE	AVGROWSIZE	BIGINT	18
15	PCTROWSCOMPRESSED	PCTCROWS	FLOAT	29
16	PCTPAGESSAVED	PCTPGSAVED	SMALLINT	4
17	NUMBER_DELETED_ROWS	DELETED	BIGINT	18
18	DATA_SIZE	SIZE	BIGINT	18
19	VARIABLE_LENGTH_SIZE	VLSIZE	BIGINT	18
20	FIXED_LENGTH_EXTENTS	FLEXTENTS	BIGINT	18
21	VARIABLE_LENGTH_EXTENTS	VLEXTENTS	BIGINT	18
22	COLUMN_STATS_SIZE	CSTATSSIZE	BIGINT	18
23	MAINTAINED_TEMPORARY_INDEX_SIZE	MTISIZE	BIGINT	18
24	NUMBER_DISTINCT_INDEXES	DISTINCTIX	INTEGER	9
25	OPEN_OPERATIONS	OPENS	BIGINT	18
26	CLOSE_OPERATIONS	CLOSES	BIGINT	18
27	INSERT_OPERATIONS	INSERTS	BIGINT	18
28	UPDATE_OPERATIONS	UPDATES	BIGINT	18
29	DELETE_OPERATIONS	DELETES	BIGINT	18
30	CLEAR_OPERATIONS	DSCLEARS	BIGINT	18
31	COPY_OPERATIONS	DSCOPIES	BIGINT	18
32	REORGANIZE_OPERATIONS	DSREORGS	BIGINT	18
33	INDEX_BUILDS	DSINXBLDS	BIGINT	18
34	LOGICAL_READS	LGLREADS	BIGINT	18
35	PHYSICAL_READS	PHYREADS	BIGINT	18
36	SEQUENTIAL_READS	SEQREADS	BIGINT	18
37	RANDOM_READS	RANREADS	BIGINT	18
38	LAST_CHANGE_TIMESTAMP	LASTCHG	TIMESTAMP	10
39	LAST_SAVE_TIMESTAMP	LASTSAVE	TIMESTAMP	10
40	LAST_RESTORE_TIMESTAMP	LASTRST	TIMESTAMP	10
41	LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP	10
42	DAYS_USED_COUNT	DAYSUSED	INTEGER	9
43	LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP	10
44	NUMBER_PARTITIONING_KEYS	NBRPKEYS	INTEGER	9
45	PARTITIONING_KEYS	PARTKEYS	VARCHAR	2880
46	SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR	10
47	SYSTEM_TABLE_NAME	SYS_TNAME	CHAR	10

5.9 Utility Services

Les procédures suivantes fournissent des interfaces permettant d'interagir avec différents éléments du système.

Nous allons les passer en revue brièvement, je vous invite à vous reporter à la documentation officielle pour de plus amples informations (et nous verrons plus en détail les 2 procédures que j'ai indiquées en rouge) :
http://www-01.ibm.com/support/knowledgecenter/api/content/ssw_ibm_i_72/rzajq/rzajqservicesutility.htm

CANCEL_SQL

La procédure CANCEL_SQL demande l'annulation d'une instruction SQL pour le travail spécifié en paramètre.

CHECK_SYSCST

La procédure CHECK_SYSCST compare les entrées dans le tableau QSYS2.SYSCONSTRAINTS entre deux systèmes .

CHECK_SYSROUTINE

La procédure CHECK_SYSROUTINE compare des entrées dans la table QSYS2.SYSROUTINES entre deux systèmes.

DUMP_SQL_CURSORS

La procédure DUMP_SQL_CURSORS répertorie les curseurs ouverts pour un travail donné.

FIND_AND_CANCEL_QSQSRVR_SQL

La procédure FIND_AND_CANCEL_QSQSRVR_SQL identifie un ensemble de travaux ayant une activité SQL, et les annule en toute sécurité .

FIND_QSQSRVR_JOBS

La procédure FIND_QSQSRVR_JOBS renvoie des informations sur un travail de QSQSRVR .

GENERATE_SQL

La procédure GENERATE_SQL génère le code SQL permettant de recréer un objet de base de données. Les résultats sont retournés dans le membre de fichier source de base de données spécifiée, ou sous la forme d'un "result set".

RESTART_IDENTITY

La procédure RESTART_IDENTITY examine une table source, détermine la colonne contenant l'identifiant et détermine également sa prochaine valeur. Cette valeur et le nom de la colonne sont transmis à une table cible qui les utilisera pour la prochaine insertion de ligne.

Dans la suite de ce chapitre, nous allons regarder un peu plus en détail les procédures CHECK_SYSROUTINE et GENERATE_SQL.

5.9.1 QSYS2.GENERATE_SQL

Annoncé sur la TR8, mais en réalité déjà disponible sur la TR7, la procédure stockée GENERATE_SQL génère le code SQL DDL (Data Definition Language) permettant de recréer un objet DB2. Le résultat peut être renvoyé dans un membre de fichier source, ou sous forme de "result set".

A noter : si le source en sortie est dirigé vers la table temporaire QTEMP/Q_GENSQL, avec pour nom de membre Q_GENSQL, alors le résultat est renvoyé simultanément sous forme de "result set".

Exemples :

- Générer le code DDL pour toutes les tables du schéma SAMPLE_CORPDB, et renvoyer le résultat sous forme de "result set" :

```
CALL QSYS2.GENERATE_SQL('%', 'SAMPLE_CORPDB', 'TABLE',
    REPLACE_OPTION => '0');
```

- Générer le code DDL d'une vue ou d'une procédure :

```
CALL QSYS2.GENERATE_SQL('CHECK_SYSROUTINE', 'SYSTOOLS', 'PROCEDURE');
CALL QSYS2.GENERATE_SQL('COMMANDES2', 'FPHSAW', 'VIEW');
```

- Générer le code DDL pour tous les indexs du schéma SAMPLE_CORPDB, dont le nom commence par un ‘X’, et placer le résultat dans le fichier source nommé DDLSOURCE/GENFILE, membre INDEXSRC.

```
CALL QSYS2.GENERATE_SQL('X%', 'SAMPLE_CORPDB', 'INDEX',
                        'GENFILE', 'DDLSOURCE', 'INDEXSRC',
                         REPLACE_OPTION => '0');
```

Pour de plus amples renseignements :

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/QSYS2.GENERATE_SQL%28%29%20procedure

5.9.2 Notion de Signature

Il est important de comprendre ce qui rend une procédure ou une fonction unique, d'un point de vue de la base de données DB2.

Procédures stockées et fonctions utilisateurs (UDF) sont caractérisés sous DB2 par une notion de signature.

Pour les procédures, la définition de signature est caractérisée par les éléments en rouge dans l'exemple ci-dessous, à savoir : schéma spécifique, nom spécifique, et le nombre de paramètres.

```
CREATE PROCEDURE MABIBL/MAPROC (
     IN PARAM1 DECIMAL(3, 0), INOUT PARAM2 INTEGER )
     LANGUAGE SQL
     SPECIFIC MABIBL/MAPROC
     NOT DETERMINISTIC
     MODIFIES SQL DATA
     CALLED ON NULL INPUT
     BEGIN
          -- Mon code SQL ici
     END
```

La signature est vérifiée au moment du CREATE PROCEDURE, avec une erreur SQL0454, qui est retournée si l'on tente de violer le caractère unique de la signature.

Pour les fonctions, la notion de signature est plus complexe. La définition de la signature pour une fonction comprend les éléments en rouge dans l'exemple ci-dessous, soit : un schéma spécifique, un nom spécifique, le nombre de paramètres, et le type des différents paramètres d'entrée/sortie.

```
CREATE FUNCTION MABIBL.CVT_ALP_2_DATE(
   DATE_ENT CHAR(10)
)
RETURNS DATE
LANGUAGE SQL
SPECIFIC  MABIBL.CVT_ALP_2_DATE
DETERMINISTIC
CONTAINS SQL
```

```
RETURN
CASE WHEN DATE_ENT is null or trim(DATE_ENT) = ''
     THEN CURRENT DATE
     ELSE DATE(TO_DATE(DATE_ENT, 'YYYY-MM-DD'))
END
```

La signature est vérifiée au moment du CREATE FUNCTION, avec une erreur SQL0454, qui est retournée si l'on tente de violer le caractère unique de la signature.

Le code source et les principales caractéristiques des routines (procédures) et fonctions sont stockées dans la table système QSYS2/SYSROUTINE.
La liste des paramètres et leur type sont stockés dans la table système QSYS2/SYSPARMS.
La combinaison de ces 2 tables permet à DB2 de contrôler la signature des objets de types routine et fonctions.

Quand un objet de type routine ou fonction est sauvegardé sur un système A, puis restauré sur un système B, les tables système QSYS2/SYSROUTINE et QSYS2/SYSPARMS sont automatiquement mises à jour.

5.9.3 SYSTOOLS.CHECK_SYSROUTINE

La procédure CHECK_SYSROUTINE compare les entrées dans la table système QSYS2.SYSROUTINES du système courant, avec les entrées de la même table sur un autre serveur, pour une base dont le nom est indiqué en paramètre d'entrée de la procédure (ce nom devant être identique sur les 2 serveurs à comparer).

Le schéma est SYSTOOLS.

Les paramètres d'entrée sont les suivants :

- remote-rdb-name : chaîne de caractères contenant le nom de la base de données "remote"
- schema-name : chaîne de caractères contenant le nom d'un schéma du système hôte à comparer avec le même schéma de la base de données "remote"
- result set : un INTEGER indiquant si un result set doit être renvoyé (valeur 0) ou pas (valeur 1).

La procédure renvoie un "result set" au client SQL "appelant". Si aucun "result set" n'est demandé, alors les différences entre les 2 systèmes sont "loguées" dans la table temporaire SESSION.SYSRTNDIFF.

Exemple :

Comparer le système courant avec un système "remote" (L001UT18) pour identifier les routines qui ne sont pas identiques dans la bibliothèque CORPDB_EX.

```
CALL SYSTOOLS.CHECK_SYSROUTINE('LP01UT18', 'CORPDB_EX')  ;
```

Il faut souligner que le code source de la procédure CHECK_SYSROUTINE peut être regénéré (soit via le logiciel System i Navigator, soit via la nouvelle procédure GENERATE_SQL). Il peut dès lors être adapté à vos besoins. Par exemple, si le nom de la base de données "remote" et le nom de la base de données locale diffèrent, il est souhaitable de pouvoir ajouter un paramètre supplémentaire à la procédure, pour tenir compte de cette différence de codification intervenant entre 2

systèmes. C'est ce que j'ai fait en créant une procédure CHECK_SYSROUTINE2, dérivée de la précédente, mais autorisant la transmission de 2 paramètres supplémentaires de manière à pouvoir un environnement de référence (serveur + bibliothèque), à comparer avec un environnement "cible" (serveur + bibliothèque), ce qui, en termes d'utilisation, donne ceci :

```
CALL SYSTOOLS.CHECK_SYSROUTINE2('TEST', 'FEHSAP' , 'PREPROD', 'FPHSAP',
0);
```

5.10 Performance Services

Plusieurs procédures stockées sont mises à la disposition des administrateurs bases de données, pour faciliter le travail de surveillance et d'optimisation des indexs.

Nous allons les passer en revue brièvement, je vous invite à vous reporter à la documentation officielle pour de plus amples informations :
`http://www-01.ibm.com/support/knowledgecenter/api/content/ssw_ibm_i_72/rzajq/rzajqservicesperf.htm`

ACT_ON_INDEX_ADVICE
La procédure ACT_ON_INDEX_ADVICE crée de nouveaux index pour une table en se basant sur les indexs conseillés par l'optimiseur SQL pour cette table.
Exemple d'utilisation :

```
CALL SYSTOOLS.ACT_ON_INDEX_ADVICE('PRODLIB',NULL,NULL,1000,NULL);
```

HARVEST_INDEX_ADVICE
La procédure HARVEST_INDEX_ADVICE génère une ou plusieurs instructions de type CREATE INDEX dans les membres d'un fichier source, pour une table spécifiée, sur la base des indexs conseillés par SQL pour cette table.

REMOVE_INDEXES
La procédure REMOVE_INDEXES supprime les indexs correspondant aux critères spécifiés.
Exemple d'utilisation :

```
CALL SYSTOOLS.REMOVE_INDEXES('MYLIB', 1, '1 MONTH') ;
```

RESET_TABLE_INDEX_STATISTICS
La procédure RESET_TABLE_INDEX_STATISTICS efface les statistiques d'utilisation des index définis sur une ou plusieurs tables.
Exemple d'utilisation :

```
CALL qsys2.Reset_Table_Index_Statistics ('MJATST', 'AMON%') ;
```

On notera également la disponibilité, depuis la V6R1, de la procédure QSYS2.OVERRIDE_QAQQINI, qui peut être utilisée pour générer une table temporaire équivalente au fichier QAQQINI.
Exemple d'utilisation :

```
CALL QSYS2.OVERRIDE_QAQQINI(1, '', '') ;
```

Pour de plus amples précisions :
http://pic.dhe.ibm.com/infocenter/iseries/v6r1m0/index.jsp?topic=/rzajq/rzajqqaqqiniproc.htm

5.11 Health Services

5.11.1 QSYS2.SYSLIMTBL

Un nouveau type d'indicateur de santé, le "suivi automatique des limites du système", est un dispositif mis à la disposition des administrateurs système, pour les aider à prévenir certaines situations de blocage.

L'outil met l'accent sur un sous-ensemble de limites du système (définies par IBM dans la table QSYS2.SQL_SIZING). Chaque fois que les limites définies dans cette table sont atteintes ou dépassées, des informations de suivi sont inscrites dans une table système DB2 appelée QSYS2/SYSLIMTBL. Une vue nommée QSYS2/SYSLIMITS est construite sur la table SYSLIMTBL, et permet d'obtenir rapidement de nombreux renseignements contextuels sur les lignes de la table.

Les différentes limites définies par IBM sont les suivantes (intitulés non traduits) :

ASP limits
- Maximum number of spool files

Database limits
- Maximum number of all rows in a partition
- Maximum number of valid rows in a partition
- Maximum number of deleted rows in a partition
- Maximum number of overflow rows in a partition
- Maximum number of variable-length segments
- Maximum number of indexes over a partition

File system limits
- Maximum number of object description entries in a library

Job limits
- Maximum number of rows locked in a unit of work
- Maximum number of row change operations in a unit of work

Journal limits
- Maximum size of a journal receiver
- Maximum number of objects that can be associated with a *MAX10M journal
- Maximum number of objects that can be associated with a *MAX250K journal
- Maximum sequence number of a *MAXOPT3 journal

- Maximum sequence number of a *MAXOPT1 or *MAXOPT2 journal

Object limits

- Maximum number of members in a source physical file

System limits

- Maximum number of jobs

Exemple 1. Examiner les travaux actifs au fil du temps

```
SELECT SBS_NAME, SIZING_NAME, CURRENT_VALUE, MAXIMUM_VALUE , A.*
FROM QSYS2.SYSLIMITS A
WHERE LIMIT_ID = 19000
ORDER BY CURRENT_VALUE DESC
```

SBS_NAME	SIZING_NAME	CURRENT_VAL...	MAXIMUM_VALUE	LAST_CHANGE_TIMESTAMP	LIMIT_CATEGORY	LIMIT_TYPE	SIZING_
-	MAXIMUM NUMBER OF JOBS	1801	485000	2013-05-12 10:10:07.051744	WORK MANAGEMENT	SYSTEM	MAXIMUI
-	MAXIMUM NUMBER OF JOBS	1401	485000	2013-05-12 10:09:42.928877	WORK MANAGEMENT	SYSTEM	MAXIMUI
-	MAXIMUM NUMBER OF JOBS	1265	485000	2013-05-12 10:10:34.337091	WORK MANAGEMENT	SYSTEM	MAXIMUI
-	MAXIMUM NUMBER OF JOBS	1001	485000	2013-05-11 10:27:37.403905	WORK MANAGEMENT	SYSTEM	MAXIMUI

Exemple 2. Examiner les valeurs maximales définies par IBM par défaut :

```
SELECT SIZING_ID, SUPPORTED_VALUE, SIZING_NAME, COMMENTS
FROM QSYS2.SQL_SIZING
ORDER BY SIZING_ID DESC
```

SIZING_ID	SUPPORTED_VALUE	SIZING_NAME	COMMENTS
25005	10	MAXIMUM SYSTEM USER LENGTH	Maximum length of a system authorization ID
25004	10	MAXIMUM SESSION USER LENGTH	Maximum length of a session authorization ID
25003	-	MAXIMUM CURRENT ROLE LENGTH	-
25002	3483	MAXIMUM CURRENT PATH LENGTH	Maximum length of an SQL path
25001	-	MAXIMUM CURRENT TRANSFORM GROUP LENGTH	-
25000	-	MAXIMUM CURRENT DEFAULT TRANSFORM GROUP LENGTH	-
20004	32718	MAXIMUM DATALINK LENGTH	Maximum length of a datalink
20002	2097151	MAXIMUM STATEMENT OCTETS SCHEMA	Maximum length of an SQL data definition language (D
20001	2097151	MAXIMUM STATEMENT OCTETS DATA	Maximum length of an SQL data manipulation language
20000	2097151	MAXIMUM STATEMENT OCTETS	Maximum length of an SQL statement
19003	10000000	MAXIMUM NUMBER OF SPOOLED FILES IN EACH INDEPENDENT ASP	Maximum number of spooled files in each independent
19002	2610000	MAXIMUM NUMBER OF SPOOLED FILES IN THE SYSTEM AND BAS...	Maximum number of spooled files in the system and ba

5.11.2 Valeurs limites

Pour éviter une consommation trop excessible d'espace de stockage au niveau de la table de SYS2/SYSLIMTBL, DB2 for i va automatiquement supprimer des données selon différents critères.

DB2 for i fournit différentes variables globales, stockées dans le schéma SYSIBMADM :

Les valeurs fixées par IBM par défaut sont les suivantes :

Nom de variable globale	Valeur
QIBM_SYSTEM_LIMITS_PRUNE_BY_ASP	100
QIBM_SYSTEM_LIMITS_PRUNE_BY_JOB	50
QIBM_SYSTEM_LIMITS_PRUNE_BY_OBJECT	20
QIBM_SYSTEM_LIMITS_PRUNE_BY_SYSTEM	100
QIBM_SYSTEM_LIMITS_SAVE_HIGH_POINTS_BY_ASP	25
QIBM_SYSTEM_LIMITS_SAVE_HIGH_POINTS_BY_JOB	5
QIBM_SYSTEM_LIMITS_SAVE_HIGH_POINTS_BY_OBJECT	5
QIBM_SYSTEM_LIMITS_SAVE_HIGH_POINTS_BY_SYSTEM	25

Pour chaque type de limite, il y a deux variables globales. La variable de taille est utilisée pour choisir le nombre des entrées les plus récemment enregistrées devrait être conservé. La variable point haut permet de choisir le nombre d'entrées de la plus haute valeur de la consommation devrait être conservé.

Vous pouvez modifier ces limites en redéfinissant les valeurs de ces variables globales, via le code SQL suivant :

```
CREATE OR REPLACE VARIABLE SYSIBMADM.QIBM_SYSTEM_LIMITS_PRUNE_BY_SYSTEM
       INTEGER
       DEFAULT  50 ;
```

Dans l'exemple ci-dessus, on conserve les 50 lignes les plus récentes pour tous les types de limites du système.
Les modifications de valeurs sont prises en compte après le prochain IPL.

6 Bonus

6.1 DSPFFD amélioré et autres outils

Je rappelle que vous disposez, dans la bibliothèque QSYS2, d'un jeu de tables systèmes dont les noms se passent de commentaire :
- SYSTABLES
- SYSCOLUMNS
- SYSVIEWS
- SYSVIEWDEP
- SYSROUTINES
- SYSROUTINEDEP
- SYSTABLESTAT
- etc..

Vous pouvez par exemple vous appuyer sur la table SYSTABLES pour vérifier qu'une même table se trouvant dans plusieurs bibliothèques a bien la même structure dans chacune de ces bibliothèques.

Exemple de vue obtenue à partir d'un outil "maison" développé en PHP : affichage de toutes les vues DB2 dont le nom est SYSTABLES, quelles que soient les bibliothèques où elles se trouvent :

Schéma	Table (sqlname)	Table (sysname)	Type	Nb.Cols.	Buffer	Description
GJABASE	SYSTABLES	SYSTABLES	V	29	3003	Vue du rép des données SQL
GJARRIGE	SYSTABLES	SYSTABLES	V	29	3003	Vue du rép des données SQL
GJARRIGE2	SYSTABLES	SYSTABLES	V	29	3003	Vue du rép des données SQL
GJABASE2	SYSTABLES	SYSTABLES	V	29	3003	Vue du rép des données SQL
OPENCART2	SYSTABLES	SYSTABLES	V	29	3003	Vue du rép des données SQL
GJARRIGE3	SYSTABLES	SYSTABLES	V	30	5091	Vue du rép des données SQL
SAMPLES	SYSTABLES	SYSTABLES	V	30	5091	Vue du rép des données SQL

Le tableau ci-dessus est produit au moyen d'une requête sur la vue SYSTABLES justement.

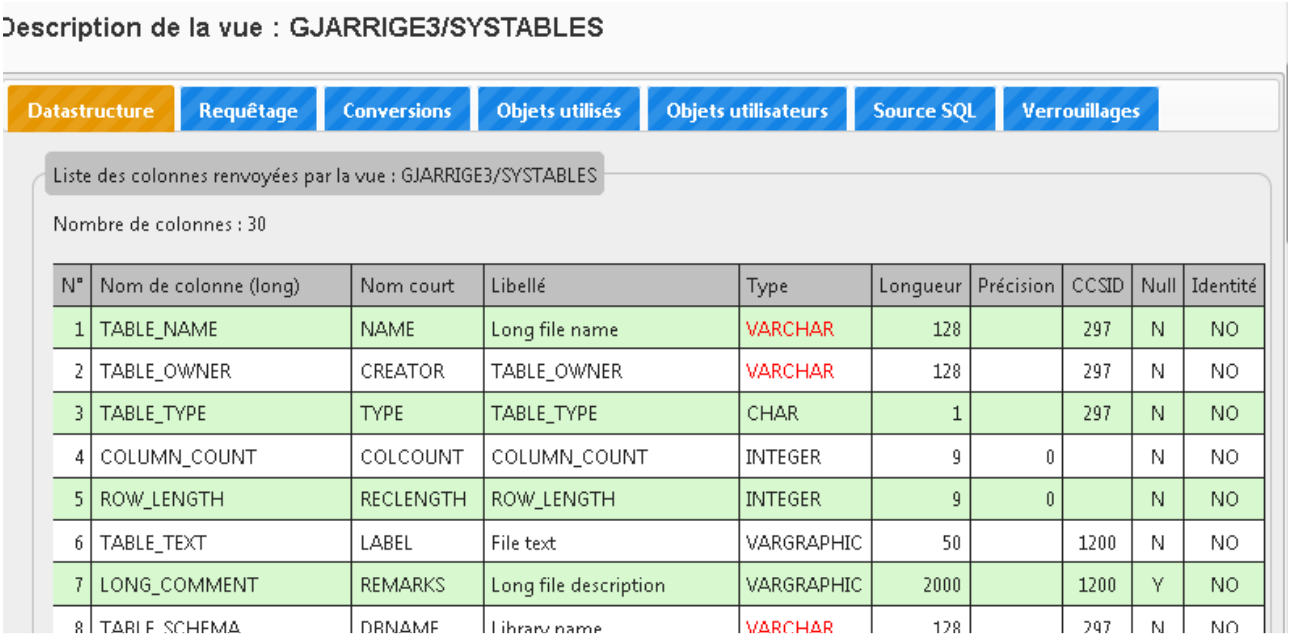
On voit que dans certains cas SYSTABLES contient 29 colonnes, et dans d'autres cas 30 colonnes (ce qui impacte aussi le buffer). Comparer le nombre de colonnes et le buffer constitue un moyen pratique et rapide de détecter des écarts entre bases de données.

ATTENTION : la comparaison de vues et de MQT sur la notion de buffer peut prêter à confusion. Certaines vues ou MQT, strictement identiques d'un point de vue du code source, peuvent présenter des buffers différents, dès lors qu'elles utilisent des fonctions d'agrégation (SUM par exemple) et/ou des formules de calcul.

Question : combien d'entre vous ont, dans le passé, développé un "DSPFFD" amélioré, pour consulter la structure des tables de vos applications ?

Pour développer ce type d'outil, dans les années 90, nous avons pour la plupart utilisé des tables temporaires générées par les commandes DSPFD et surtout DSPFFD.

Voici un "DSPFFD" amélioré produit au moyen d'un peu de code PHP exploitant le résultat d'une requête sur la table système QSYS2.SYSCOLUMNS.



Description de la vue : GJARRIGE3/SYSTABLES

Datastructure | Requêtage | Conversions | Objets utilisés | Objets utilisateurs | Source SQL | Verrouillages

Liste des colonnes renvoyées par la vue : GJARRIGE3/SYSTABLES

Nombre de colonnes : 30

N°	Nom de colonne (long)	Nom court	Libellé	Type	Longueur	Précision	CCSID	Null	Identité
1	TABLE_NAME	NAME	Long file name	VARCHAR	128		297	N	NO
2	TABLE_OWNER	CREATOR	TABLE_OWNER	VARCHAR	128		297	N	NO
3	TABLE_TYPE	TYPE	TABLE_TYPE	CHAR	1		297	N	NO
4	COLUMN_COUNT	COLCOUNT	COLUMN_COUNT	INTEGER	9	0		N	NO
5	ROW_LENGTH	RECLENGTH	ROW_LENGTH	INTEGER	9	0		N	NO
6	TABLE_TEXT	LABEL	File text	VARGRAPHIC	50		1200	N	NO
7	LONG_COMMENT	REMARKS	Long file description	VARGRAPHIC	2000		1200	Y	NO
8	TABLE_SCHEMA	DBNAME	Library name	VARCHAR	128		297	N	NO

La structure affichée ici est justement celle de la table système SYSTABLES dont je vous mets pour info la structure ci-dessous (récupérée via le même outil, mais sans la colonne « libellé », faute de place) :

N°	Nom de colonne (long)	Nom court	Type	Longueur	Précision
1	TABLE_NAME	NAME	VARCHAR	128	
2	TABLE_OWNER	CREATOR	VARCHAR	128	
3	TABLE_TYPE	TYPE	CHAR	1	
4	COLUMN_COUNT	COLCOUNT	INTEGER	9	0
5	ROW_LENGTH	RECLENGTH	INTEGER	9	0
6	TABLE_TEXT	LABEL	VARGRAPHIC	50	
7	LONG_COMMENT	REMARKS	VARGRAPHIC	2000	
8	TABLE_SCHEMA	DBNAME	VARCHAR	128	
9	LAST_ALTERED_TIMESTAMP	ALTEREDTS	TIMESTAMP	10	
10	SYSTEM_TABLE_NAME	SYS_TNAME	CHAR	10	
11	SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR	10	
12	FILE_TYPE	FILETYPE	CHAR	1	
13	BASE_TABLE_CATALOG	LOCATION	VARCHAR	18	
14	BASE_TABLE_SCHEMA	TBDBNAME	VARCHAR	128	

15	BASE_TABLE_NAME	TBNAME	VARCHAR	128	
16	BASE_TABLE_MEMBER	TBMEMBER	VARCHAR	10	
17	SYSTEM_TABLE	SYSTABLE	CHAR	1	
18	SELECT_OMIT	SELECTOMIT	CHAR	1	
19	IS_INSERTABLE_INTO	INSERTABLE	VARCHAR	3	
20	IASP_NUMBER	IASPNUMBER	SMALLINT	4	0
21	ENABLED	ENABLED	VARCHAR	3	
22	MAINTENANCE	MAINTAIN	VARCHAR	6	
23	REFRESH	REFRESH	VARCHAR	9	
24	REFRESH_TIME	REFRESHDTS	TIMESTAMP	10	
25	MQT_DEFINITION	MQTDEF	DBCLOB	2097152	
26	ISOLATION	ISOLATION	CHAR	2	
27	PARTITION_TABLE	PART_TABLE	VARCHAR	11	
28	TABLE_DEFINER	DEFINER	VARCHAR	128	
29	MQT_RESTORE_DEFERRED	MQTRSTDFR	CHAR	1	
30	ROUNDING_MODE	DECFLTRND	CHAR	1	

A partir du moment où vous décidez d'exploiter les trésors contenus dans les tables systèmes DB2, et à condition de maîtriser un langage de développement web (par exemple PHP), il n'y a pas de limites aux types d'outils que vous pouvez développer pour administrer plus facilement vos bases de données DB2.

Je n'avais pas pensé à la montrer lors de la présentation, aussi je l'ai ajoutée après coup, voici pour information, la structure de la vue QSYS2.SYSCOLUMNS. A noter que cette structure diffère elle aussi sensiblement selon les versions d'OS (39 colonnes en V7R1, un peu moins dans les versions antérieures) :

N°	Nom de colonne (long)	Nom court	Type	Longueur	Précision
1	COLUMN_NAME	NAME	VARCHAR	128	
2	TABLE_NAME	TBNAME	VARCHAR	128	
3	TABLE_OWNER	TBCREATOR	VARCHAR	128	
4	ORDINAL_POSITION	COLNO	INTEGER	9	0
5	DATA_TYPE	COLTYPE	VARCHAR	8	
6	LENGTH	LENGTH	INTEGER	9	0
7	NUMERIC_SCALE	SCALE	INTEGER	9	0
8	IS_NULLABLE	NULLS	CHAR	1	
9	IS_UPDATABLE	UPDATES	CHAR	1	
10	LONG_COMMENT	REMARKS	VARGRAPHIC	2000	
11	HAS_DEFAULT	DEFAULT	CHAR	1	

12	COLUMN_HEADING	LABEL	VARGRAPHIC	60	
13	STORAGE	STORAGE	INTEGER	9	0
14	NUMERIC_PRECISION	PRECISION	INTEGER	9	0
15	CCSID	CCSID	INTEGER	9	0
16	TABLE_SCHEMA	DBNAME	VARCHAR	128	
17	COLUMN_DEFAULT	DFTVALUE	VARGRAPHIC	2000	
18	CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER	9	0
19	CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER	9	0
20	NUMERIC_PRECISION_RADIX	RADIX	INTEGER	9	0
21	DATETIME_PRECISION	DATPRC	INTEGER	9	0
22	COLUMN_TEXT	LABELTEXT	VARGRAPHIC	50	
23	SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR	10	
24	SYSTEM_TABLE_NAME	SYS_TNAME	CHAR	10	
25	SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR	10	
26	USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR	128	
27	USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR	128	
28	IS_IDENTITY	IDENTITY	VARCHAR	3	
29	IDENTITY_GENERATION	GENERATED	VARCHAR	10	
30	IDENTITY_START	START	DECIMAL	31	0
31	IDENTITY_INCREMENT	INCREMENT	DECIMAL	31	0
32	IDENTITY_MINIMUM	MINVALUE	DECIMAL	31	0
33	IDENTITY_MAXIMUM	MAXVALUE	DECIMAL	31	0
34	IDENTITY_CYCLE	CYCLE	VARCHAR	3	
35	IDENTITY_CACHE	CACHE	INTEGER	9	0
36	IDENTITY_ORDER	ORDER	VARCHAR	3	
37	COLUMN_EXPRESSION	EXPRESSION	DBCLOB	2097152	
38	HIDDEN	HIDDEN	VARCHAR	1	
39	HAS_FLDPROC	FLDPROC	VARCHAR	1	

6.2 Analyse de dépendances via les tables systèmes

Une vue DB2 peut faire appel à une ou plusieurs tables et/ou vues. Les vues dépendantes, peuvent elle-même faire appel à d'autres vues, ce qui peut aboutir à des niveaux de dépendance élevés.

En cas de nécessité de modifier une vue, il est utile de connaître les dépendances par rapport à l'objet à modifier.

Pour ce faire, on peut par exemple s'appuyer sur la table SYSVIEWDEP et sur le principe de la récursivité tel qu'il est implémenté dans DB2, pour analyser les dépendances entre objets.

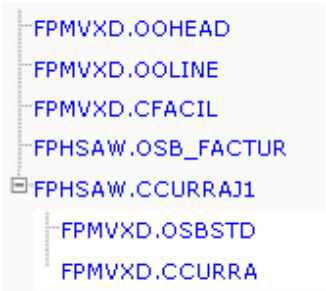
```
WITH
-- création d'une CTE définissant les paramètres de la requête et l'objet
de départ de l'analyse
TMP_PARAM (LIB_REF, OBJ_REF) AS (
   SELECT 'mabib' as LIB_REF,
          'mabib.mavueDB2' as OBJ_REF
   FROM SYSIBM.SYSDUMMY1
),
-- création d'une CTE consolidant les vues et les objets dépendants dans
une seule liste
TMP_LISTOBJ (PARENT_LIB, PARENT_OBJ, CHILD_LIB, CHILD_OBJ) AS (
      SELECT A.TABLE_SCHEMA AS PARENT_LIB, A.TABLE_NAME AS PARENT_OBJ,
             B.OBJECT_SCHEMA AS CHILD_LIB, B.OBJECT_NAME AS CHILD_OBJ
      FROM QSYS2.SYSVIEWS A
      LEFT OUTER JOIN QSYS2.SYSVIEWDEP B
        ON A.TABLE_SCHEMA = B.VIEW_SCHEMA AND A.TABLE_NAME = B.VIEW_NAME
      WHERE A.TABLE_SCHEMA = (SELECT LIB_REF FROM TMP_PARAM)
),
-- CTE simplifiant l'écriture du nom des parents et enfants
TMP_BASE (PARENT, CHILD) AS (
    SELECT trim(PARENT_LIB) CONCAT '.' CONCAT trim(PARENT_OBJ) AS PARENT,
           trim(CHILD_LIB) CONCAT '.' CONCAT trim(CHILD_OBJ) AS CHILD
    FROM TMP_LISTOBJ
) ,
-- Dernière CTE définissant l'arbre hiérarchique (technique récursive)
TREE ( PARENT, CHILD, LVL) AS (
    SELECT PARENT, CHILD, 1
            FROM TMP_BASE
```

```
            WHERE PARENT = (SELECT OBJ_REF FROM TMP_PARAM)
    UNION ALL
    SELECT D.PARENT, D.CHILD, T.LVL + 1
            FROM TMP_BASE D, TREE T
            WHERE D.PARENT = T.CHILD
              AND D.PARENT != D.CHILD AND T.LVL < 20
)
SELECT PARENT, CHILD, LVL FROM TREE
;
```

La requête ci-dessus permet d'obtenir le tableau suivant (en considérant que le point de départ est la vue OO_COMMANDES2 de la bibliothèque FPHSAW) :

PARENT	CHILD	LVL
FPHSAW.OO_COMMANDES2	FPMVXD.OOHEAD	1
FPHSAW.OO_COMMANDES2	FPMVXD.OOLINE	1
FPHSAW.OO_COMMANDES2	FPMVXD.CFACIL	1
FPHSAW.OO_COMMANDES2	FPHSAW.OSB_FACTUR	1
FPHSAW.OO_COMMANDES2	FPHSAW.CCURRAJ1	1
FPHSAW.OSB_FACTUR	FPMVXD.OSBSTD	2
FPHSAW.CCURRAJ1	FPMVXD.CCURRA	2

On peut utiliser ce résultat pour produire une liste HTML (balises <ul> et <li>), et utiliser un module Javascript (comme par exemple le plugin jQuery Treeview) pour produire un affichage de type arborescent tel que celui ci-dessous :



On notera qu'il est possible de décliner cette technique sur une table de références produite par la commande système DSPPGMREF (mais attention, DSPPGMREF produit une table ne contenant que les noms courts des objets DB2).
On peut aussi compléter la technique ci-dessus en ajoutant à la requête la table système QSYS.SYSROUTINEDEP, de manière à disposer de références croisées plus exhaustives, incluant les procédures stockées.

A noter : pour ceux que la récursivité SQL intéresse, je signale que Christian Massé (Volubis) avait consacré une session à ce sujet ici même en 2012 (cf. session S18 accessible via l'URL suivante) :

http://www-05.ibm.com/fr/events/modernisation_i_2012/presentations.html

7 Conclusion

DB2 for i aime de plus en plus les administrateurs (systèmes et bases de données).

Cette tendance devrait encore se renforcer avec les prochaines versions de DB2, que ce soit au travers des "releases", ou des "Technology Refresh".

La balle est dans notre camp pour exploiter au maximum ces possibilités nouvelles que nous offre IBM au travers de la base de données.