IBM Power Systems - IBM i

Modernisation, développement d'applications et DB2 sous IBM i

Technologies, outils et nouveautés 2012-2013

8 et 9 avril 2013 – IBM Client Center Paris, Bois-Colombes

S17 – DB2/SQL Trucs et astuces

Mardi 9 avril – 11h00-12h30

Nathanaël BONNET / Pierre-Louis BERTHOIN - GAIA

Introduction

- SQL est un langage
  - Répandu
    - Le plus universel du monde informatique
  - Puissant
    - De nombreuses possibilités de manipulation des données
    - De la façon la plus efficace
- Tout un chacun connait les manipulations élémentaires
  - SELECT
  - UPDATE, DELETE, INSERT
- Nous allons découvrir ici des fonctionnalités accessibles et utiles

OUTILS

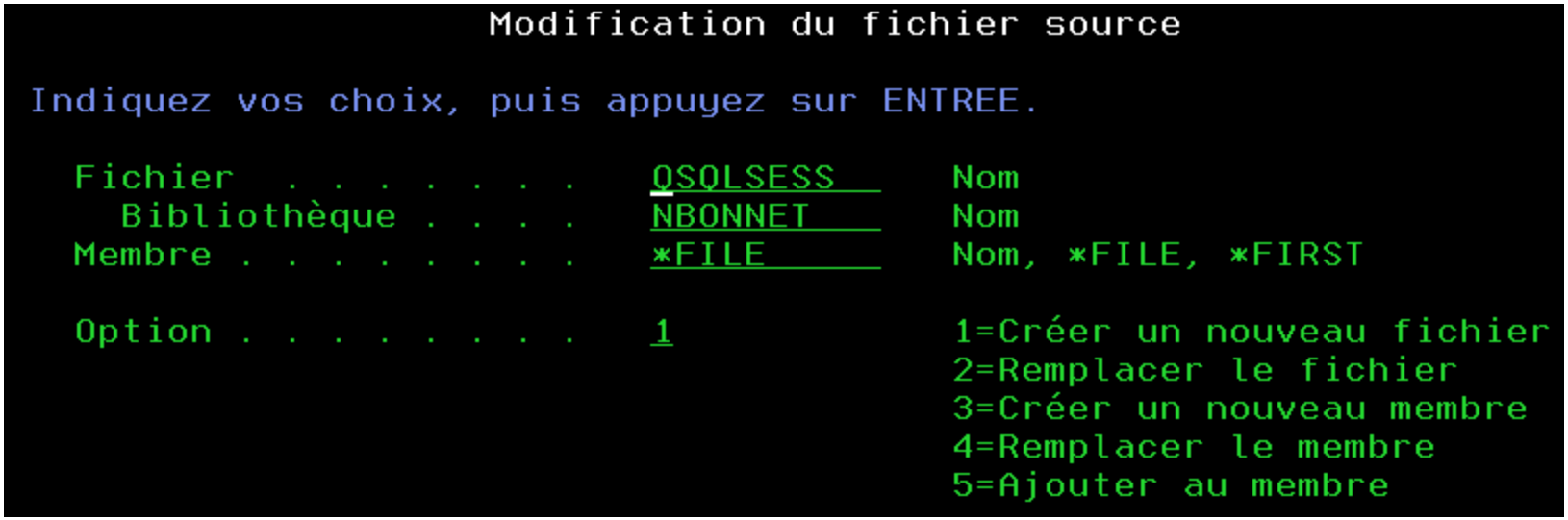
Outils

- Les outils IBM, spécifiques IBM i
  - SQL « interactif »
    - STRSQL !
    - System i Navigator, exécution de script SQL
      - Offre plus de souplesse et plus de possibilités
  - SQL « batch »
    - RUNSQLSTM
    - RUNSQL

STRSQL

- Historique de vos commandes
  - Par défaut, vous retrouver vos dernières commandes
  - Pour sauvegarder dans un fichier spécifique
    - F13 + Option 4 « Sauvegarde de la session dans un fichier source »

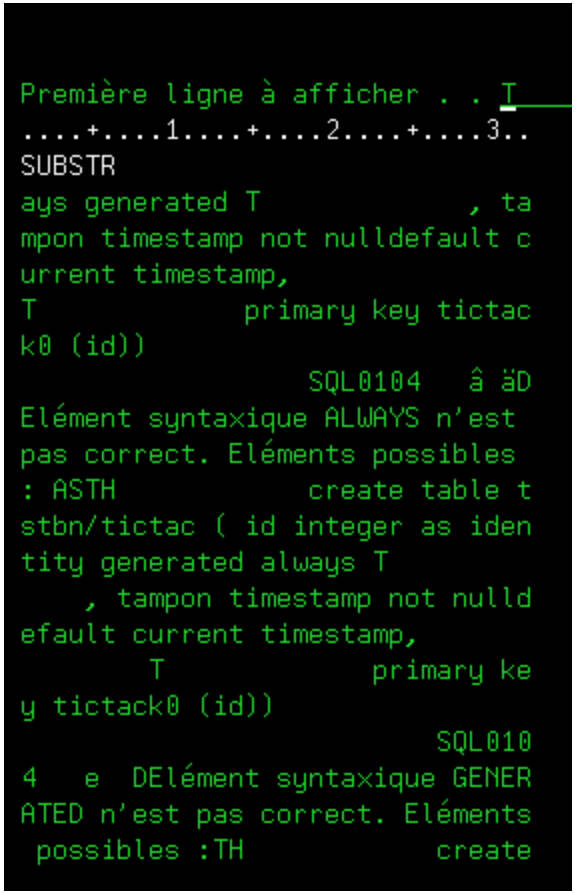
    Ou
    - Option 4 en sortant



STRSQL

- Historique des commandes d'un autre utilisateur

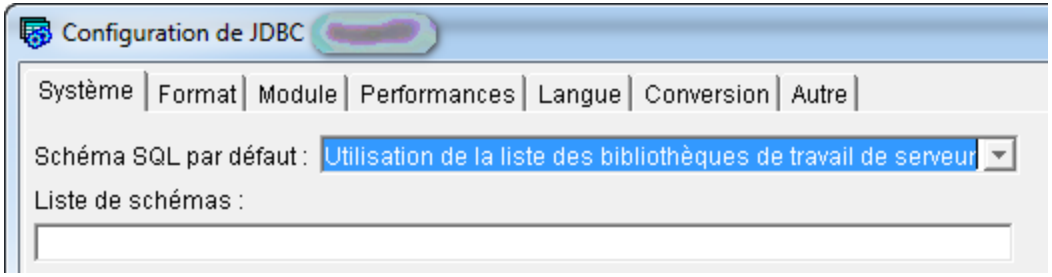
```
DMPSYSOBJ OBJ(ISQLSTBONNET*)
          CONTEXT(QRECOVERY)
          TYPE(19) SUBTYPE(EE)

CRTPF FILE(NBONNET/PF132)
      RCDLEN(132)

CPYSPLF FILE(QPSRVDMP)
        TOFILE(NBONNET/PF132)

SELECT
   substr( PF132 , 88 , 32 )
   FROM nbonnet/pf132
```



  - Dans le spool
    - les instructions SQL se trouvent
    - de la position 88 sur une longueur de 32

System i Navigator

- Se connecte à DB2 par JDBC
  - Travail QZDASOINIT/QUSER/nnnnnn



  - Ces valeurs influent sur la partie utilisateur de la liste de bibliothèque
    - Si une CURLIB est indiquée sur le profil utilisateur, cette dernière n'est pas impactée

Schéma défaut Liste schémas	JOBD	LIBDFT	Blanc
Blanc	JOBD	LIBDFT	**JOBD**
LIB1 LIB2	JOBD + LIB1 + LIB2	LIBDFT + LIB1 + LIB2	LIB1 LIB2

  - Permet d'enregistrer les scripts aussi bien en local sur votre PC que dans un membre PF-SRC

RUNSQLSTM

- RUNSQLSTM
  - Permet d'exécuter un script SQL
    - Depuis un PF-SRC
    - Depuis un fichier de l'IFS
  - Script SQL
    - Ensemble de commande
    - Séparées par « ; »

RUNSQL

- RUNSQL
  - Disponibilité
    - Niveau 14 de SF99701
    - Niveau 25 de SF99601
  - Exécute une instruction SQL
    - En dur
    - Variable CL

```
RUNSQL SQL('DELETE FROM myfile WHERE itsrecid=''D''')

CHGVAR     VAR(&SQL_STMT) VALUE( +
             'Insert into SAMPLE +
              Values(''' *Cat &Class *Cat ''', ' +
              *Cat &Status *Cat +
              ','''2013-04-09''')')
RUNSQL SQL(&SQL_STMT)
```

Exécution de commande CL

- Par System i Navigator et RUNSQLSTM
  - Syntaxe CL: commande ;

```
cl: DSPOBJD OBJ(NBONNET/*ALL)
            OBJTYPE(*ALL)
            OUTPUT(*OUTFILE)
            OUTFILE(NBONNET/DSPOBJD)   ;

select * from nbonnet.dspobjd ;
```

ODDCEN	ODDDAT	ODDTIM	ODLBNM	ODOBNM	ODOBTP	ODOBAT
1	130313	171319	NBONNET	DSPOBJD	*FILE	PF
1	130313	171319	NBONNET	PF132	*FILE	PF
1	130313	171319	NBONNET	QSQLSESS	*FILE	PF

  - Plus simple que la procédure cataloguée QCMDEXC
  - Très utile pour réaliser des scripts complexes avec fichiers de travail, traitements à déclencher etc …
    - Modifier la liste de bibliothèques à la volée via un CL
    - …

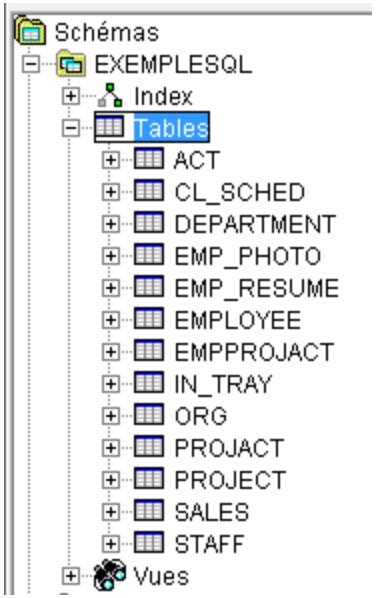
ENVIRONNEMENT DE TEST REPRODUCTIBLE

CREATE_SQL_SAMPLE

- IBM fournit une procédure cataloguée
  - `Qsys/create_sql_sample`

```
call create_sql_sample('EXEMPLESQL') ;
```

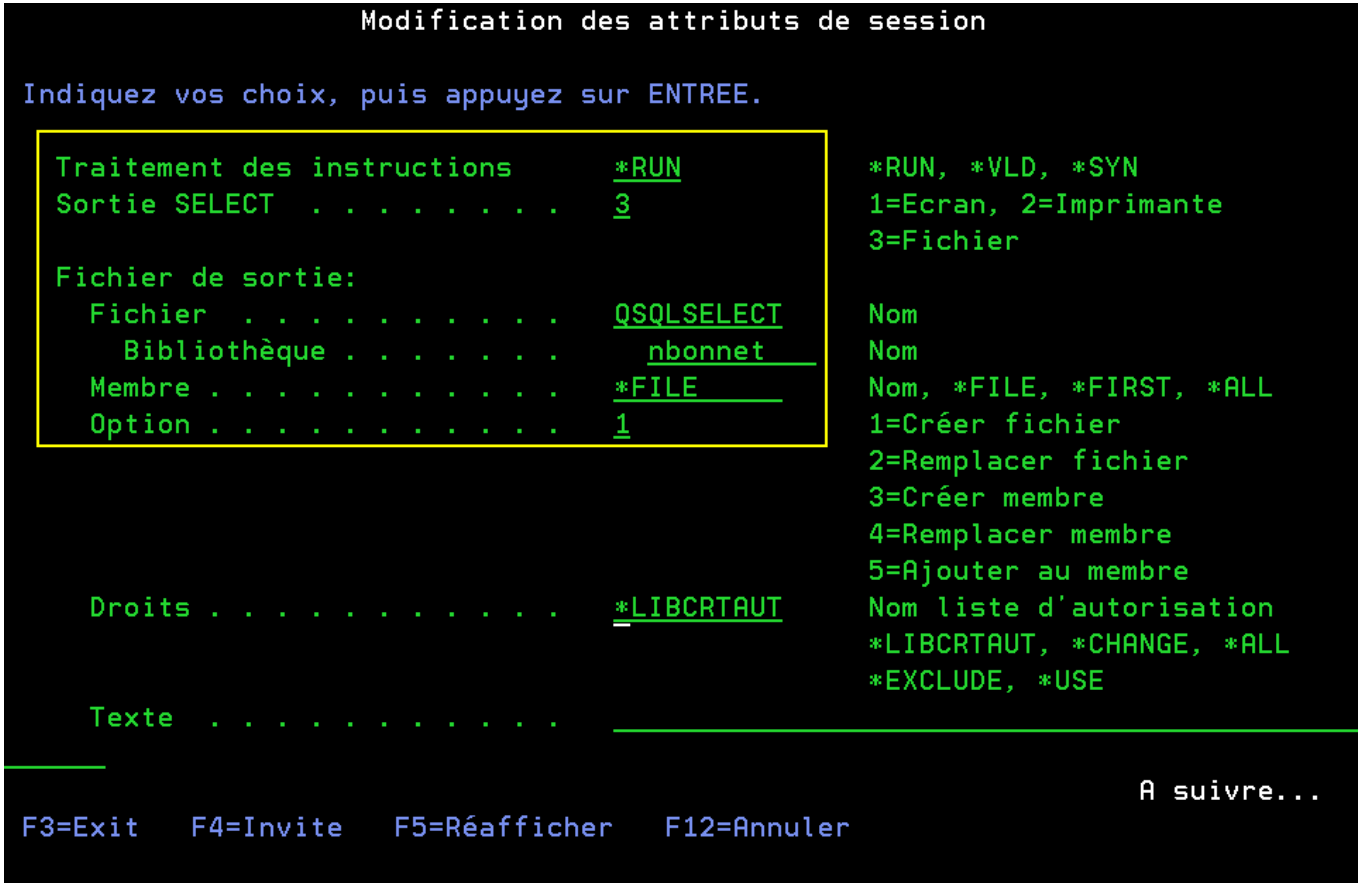
  - Permettant de construire une base de test
    - Créé la collection dont le nom est transmis en paramètre
  - Utilisée dans de nombreux exemples IBM
  - Permet d'avoir un jeu de test reproductible
    - Tables, index, vue
    - LF, DDM
    - Contraintes
    - Alias



ENREGISTRER LE RÉSULTAT D'UNE REQUÊTE (SELECT)

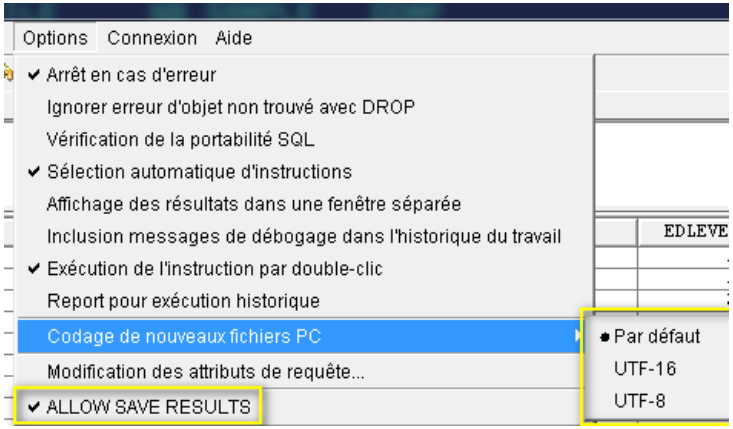
STRSQL

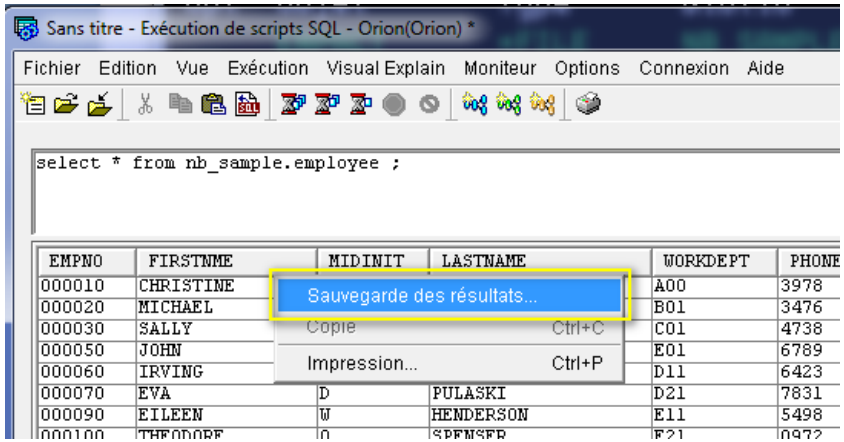
- F13 (Services), option 1 (modification des attributs de session)

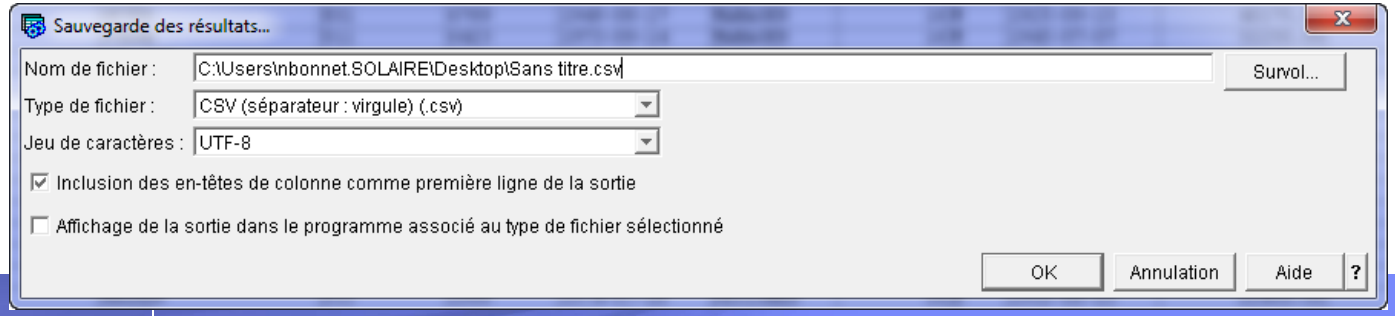


System i Navigator

- Dans les options
  - Réglez les paramètres suivants
  - Puis clique-droit
    - Encodage
    - Formats







Dans une table

- Existante au format de la requête

```
insert into nb_sample.employee_manager
select *
from nb_sample.employee
where JOB = 'MANAGER' ;
```

- Créer une table depuis une table existante

```
create table nb_sample.employee_manager
like nb_sample.employee ;
```

  - Ne copie pas les données

Dans une table

- Créer une table depuis une requête (structure inconnue à priori)

```
create table nb_sample.employee_extr as (
select empno, firstnme,
       lastname, hiredate,
       birthdate, salary,
       year( hiredate ) - year( birthdate) as age_embauche
from nb_sample.employee
where job = 'DESIGNER' )
with data ;     // with no data
```

pour créer la structure uniquement

```
select * from nb_sample.employee_extr ;
```

EMPNO	FIRSTNME	LASTNAME	HIREDATE	BIRTHDATE	SALARY	AGE_EMBAUCHE
000150	BRUCE	ADAMSON	1972-02-12	1947-05-17	25280.00	25
000160	ELIZABETH	PIANKA	1977-10-11	1955-04-12	22250.00	22
000170	MASATOSHI	YOSHIMURA	1978-09-15	1951-01-05	24680.00	27
000180	MARILYN	SCOUTTEN	1973-07-07	1949-02-21	21340.00	24
000190	JAMES	WALKER	1974-07-26	1952-06-25	20450.00	22
000200	DAVID	BROWN	1966-03-03	1941-05-29	27740.00	25
000210	WILLIAM	JONES	1979-04-11	1953-02-23	18270.00	26
000220	JENNIFER	LUTZ	1968-08-29	1948-03-19	29840.00	20
200170	KIYOSHI	YAMAMOTO	1978-09-15	1951-01-05	24680.00	27
200220	REBA	JOHN	1968-08-29	1948-03-19	29840.00	20

ALIAS

Gestion des membres

- Par définition, SQL ne gère pas la notion de membre
  - Une table est mono-membre
    - Nom du membre = nom de la table = nom du format (par défaut)
- Pour utiliser un membre particulier, on passe par un alias

```
CREATE ALIAS LIB.ALIAS FOR LIB/FICHIER(MEMBRE)
```

```
create alias exemplesql.qstrup for qgpl.qclsrc(qstrup) ;

select * from exemplesql.qstrup ;
```

SRCSEQ	SRCDAT	SRCDTA
1.00	0	/**/
2.00	0	/* */
3.00	0	/* 5761SS1 V6R1M0 080215 Sortie RTVCLSRC 19/03/12 08:41:52 */
4.00	0	/* */
5.00	0	/* Nom du programme . . . . . : QSTRUP PN*/
6.00	0	/* Nom de la bibliothèque . . . . . : QGPL PL*/
7.00	0	/* Fichier source initial . . . . . : QCLSRC SN*/
8.00	0	/* Nom de la bibliothèque . . . . . : QGPL SL*/
9.00	0	/* Membre source initial . . . . . : QSTRUP SM*/
10.00	0	/* Modification du fichier source */

  - C'est un objet permanent

```
DROP ALIAS LIB.ALIAS
```

DATE, HEURE, HORODATAGE

Format

- La notion de format de date et heure
  - S'applique au rendu uniquement
  - Pour le stockage et les calculs : un unique format
- Exemple

```
select current date        as current_date,
       date('2013-04-08') as date_1,
       date('08/04/2013') as date_2,
       date('08.04.2013') as date_3
from sysibm.sysdummy1 for read only ;
```

CURRENT_DATE	DATE_1	DATE_2	DATE_3
19.03.2013	08.04.2013	04.08.2013	08.04.2013

*EUR

CURRENT_DATE	DATE_1	DATE_2	DATE_3
2013-03-19	2013-04-08	2013-08-04	2013-04-08

*ISO

CURRENT_DATE	DATE_1	DATE_2	DATE_3
19/03/13	08/04/13	04/08/13	08/04/13

*DMY

Format

- Le format n'a aucun impact sur les calculs de date

```
select current date as current_date,
       date('2013-04-08') + 1 day as date_1,
       date('08/04/2013') - 2 month as date_2,
       date('08.04.2013') + 3 years as date_3,
       date('08.04.2013') - day(date('2013-04-03')) day as
date_4
from sysibm.sysdummy1 for read only ;
```

CURRENT_DATE	DATE_1	DATE_2	DATE_3	DATE_4
2013-03-19	2013-04-09	2013-06-04	2016-04-08	2013-04-05

- Conseil
  - Travailler toujours en *ISO
  - Indiquer des formats de date sur les rendus : DSPF, PRTF

Arithmétique

- Les fonctions
  - YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MICROSECOND
    - Extraction et ajout/soustraction de durée à une date

```
select date('2013-04-08')          as "Aujourd'hui",
       date('2013-04-08') - 1 day as "Hier",
       date('2013-04-08') + 1 day as "Demain",
       day( date('2013-04-08') )   as "N° jour"
from sysibm.sysdummy1 for read only ;
```

Aujourd'hui	Hier	Demain	N° jour
2013-04-08	2013-04-07	2013-04-09	8

Arithmétique

- Les fonctions
  - DAYS
    - Représentation d'une date sous forme d'un entier
      - Nombre de jour écoulés depuis le 0001-01-01

```
select days( date('2013-04-08') ) as "Nb jour",
       date( days( date('2013-04-08') ) ) as "Date",
       date( 1 ) as "Date mini"
from sysibm.sysdummy1 for read only ;
```

Nb jour	Date	Date mini
734966	2013-04-08	0001-01-01

Arithmétique

- Les fonctions
  - Les opérations + et – entre deux dates
    - Retournent une valeur décimale de 8 de long
      - Nombre d'années, mois et jours d'écart
      - Ce n'est pas une date !

```
select date('2013-04-09') - date('2012-03-25')
          as "Diff pseudo date",
       days( date('2013-04-09') ) -
          days( date('2012-03-25') )
          as "Diff jours ",
       date( days( date('2013-04-09') ) -
          days( date('2012-03-25') ) )
          as "Diff date"
from sysibm.sysdummy1 for read only ;
```

Diff pseudo date	Diff jours	Diff date
10015	380	0002-01-15

Depuis le 0001-01-01

Formatage de dates

- De nombreuses données représentant des dates ne sont pas des dates
  - Stockées en numériques
    - 25032013, 250313 …
  - Stockées sur plusieurs colonnes
    - Format IBM
      - Siècle (1,0) : 0, 1, 2 …
      - Date (6,0) : MMDDYY
    - Autres
      - Siècle (2,0), année (2,0), mois (2,0), jour (2,0)
      - Année (4,0), mois (2,0), jour (2,0)
      - En alphanumérique, formats hétérogènes …
  - En dehors des problèmes de validité des données stockées, cela pose le problème de leur manipulation sous forme de date
    - Calcul de période, écart …

Formatage de dates

- Exemple : DSPOBJD

```
CL: DSPOBJD OBJ(QDEVTOOLS/*ALL) OBJTYPE(*ALL) OUTPUT(*OUTFILE)
OUTFILE(NBONNET/QDEVTOOLS) ;
```

```
ODCCEN     ALPHA          1       1       111        E/S      Creation
                                                              Century
  Texte descriptif de la zone . . . . . :  Creation century: 0=19xx, 1=20xx
  ID codé de jeu de caractères    . . . . . :    297
ODCDAT     ALPHA          6       6       112        E/S      Creation
                                                              Date
  Texte descriptif de la zone . . . . . :  Creation date (MMDDYY)
```

Formatage de dates

- Exemple : DSPOBJD

```
select a.odobnm, a.odobtp , a.odctim,
       time( substr( a.odctim , 1 , 2 ) || ':' ||
             substr( a.odctim , 3 , 2 ) || ':' ||
             substr( a.odctim , 5 , 2 ) ) as
          "Creation time",
       a.odccen, a.odcdat,
       date( case a.odccen when 0 then '19' when 1
             then '20' end ||
             substr( a.odcdat , 5 , 2 ) || '-' ||
             substr( a.odcdat , 1 , 2 ) || '-' ||
             substr( a.odcdat , 3 , 2 ) ) as
          "Creation date"
from nbonnet.qdevtools  a
for read only ;
```

ODOBNM	ODOBTP	ODCTIM	Creation time	ODCCEN	ODCDAT	Creation date
QDTS00IZ	*PGM	080852	08.08.52	1	091907	2007-09-19
QDTS00MR	*PGM	145337	14.53.37	1	101007	2007-10-10
QDTS00MS	*PGM	145338	14.53.38	1	101007	2007-10-10

Formatage de dates

- Pour simplifier les syntaxes, créez vos propres fonctions SQL (UDF – User Defined Function)

```
select a.odobnm, a.odobtp ,
       a.odccen, a.odcdat,
       cmdy( a.odccen, a.odcdat ) as "Creation date"
from nbonnet.qdevtools a
where cmdy( a.odccen, a.odcdat )
      between current date - 15 day and
              current date + 15 day
for read only ;
```

  - De nombreux exemples disponibles sur le web

Formatage de dates

- cmdy

```
create function CMDY (
 P_Siecle      varchar( 1 ) ,
 P_DateMJA     varchar( 6 ) )
returns date

language sql
deterministic
contains sql
returns null on null input
no external action
set option datfmt = *iso

begin

declare Siecle char(2) ;

declare DateInvalide condition for '22007' ;
declare exit handler for DateInvalide
   begin
      return cast( null as date ) ;
      signal sqlstate '01HDI' set
message_text = 'Date invalide.' ;
   end ;
```

```
case P_Siecle
   when '0' then
      set Siecle = '19' ;
   when '1' then
      set Siecle = '20' ;
   when '2' then
      set Siecle = '21' ;
   ...
   else
      set siecle = '00' ;
end case ;

return date( Siecle concat
             substr(P_DateMJA, 5, 2) concat
'-' concat
             substr(P_DateMJA, 1, 2) concat
'-' concat
             substr(P_DateMJA, 3, 2) ) ;

end ;
```

TIMESTAMP_FORMAT / TO_DATE

- Depuis la 6.1
  - Retourne un horodatage depuis
    - La représentation alphanumérique de la valeur
    - La représentation alphanumérique du format
- Exemple

```
select       timestamp_format('2013-04-08', 'YYYY-MM-DD')   ,
       date( timestamp_format('2013-04-08', 'YYYY-MM-DD') ),
       date( timestamp_format('2013-08', 'YYYY-DD') ),
       date( timestamp_format('130408', 'YYMMDD') ),
       date( timestamp_format('08', 'DD') )
from sysibm.sysdummy1
for read only ;
```

00001	00002	00003	00004	00005
2013-04-08 00:00:00.000000	2013-04-08	2013-03-08	2013-04-08	2013-03-08

CTE (COMMON TABLE EXPRESSION)

COMMON TABLE EXPRESSION

- C'est une table temporaire
  - Contenant une instruction SELECT
  - Le temps de la requête
  - Plusieurs CTE possibles par requête
  - Un identifiant et une définition unique
  - Peuvent se référencer entre elles
    - Dans l'ordre de déclaration
  - Si défini avec le nom d'une table (vue) réelle
    - Est utilisée en priorité
- Syntaxe générale

```
WITH cte1 (id, nom) AS (SELECT a, b FROM table1),
     cte2           AS (SELECT nom, count(*) FROM cte1)
SELECT *
FROM cte2 ;
```

COMMON TABLE EXPRESSION

- Exemple
  - Ecart pour chaque employé : nombre d'années de présence et salaire par rapport à la moyenne des personnes dans le service

```
with tmp1 (dept, avg_years, avg_salary) as
    (select dept, avg(years),
            cast(avg(salary + comm) as decimal(8 , 2))
     from nb_sample.staff group by dept)
select a.dept, a.id, a.name, a.years,
       a.years - b.avg_years as "Ecart age",
       a.salary,
       a.salary - b.avg_salary as "Ecart salaire"
from nb_sample.staff a
join tmp1 b on a.dept = b.dept
order by dept, name
for read only ;
```

COMMON TABLE EXPRESSION

DEPT	ID	NAME	YEARS	Ecart age	SALARY	Ecart salaire
10	240	Daniels	5	-3	19260.25	-
10	260	Jones	12	4	21234.00	-
10	210	Lu	10	2	20010.00	-
10	160	Molinare	7	-1	22959.20	-
15	50	Hanes	10	4	20659.80	6413.73
15	170	Kermisch	4	-2	12258.50	-1987.57
15	110	Ngan	5	-1	12508.20	-1737.87
15	70	Rothman	7	1	16502.83	2256.76
20	80	James	-	-	13504.60	-2093.98
20	20	Pernal	8	1	18171.25	2572.67
20	10	Sanders	7	0	18357.50	2758.92
20	190	Sneider	8	1	14252.75	-1345.83
38	180	Abrahams	3	-1	12009.75	-3413.27
38	30	Marenghi	5	1	17506.75	2083.73
38	120	Naughton	-	-	12954.75	-2468.27
38	40	O'Brien	6	2	18006.00	2582.98
38	60	Quigley	-	-	16808.30	1385.28

Valeurs nulles

COMMON TABLE EXPRESSION

- Syntaxe pour insert

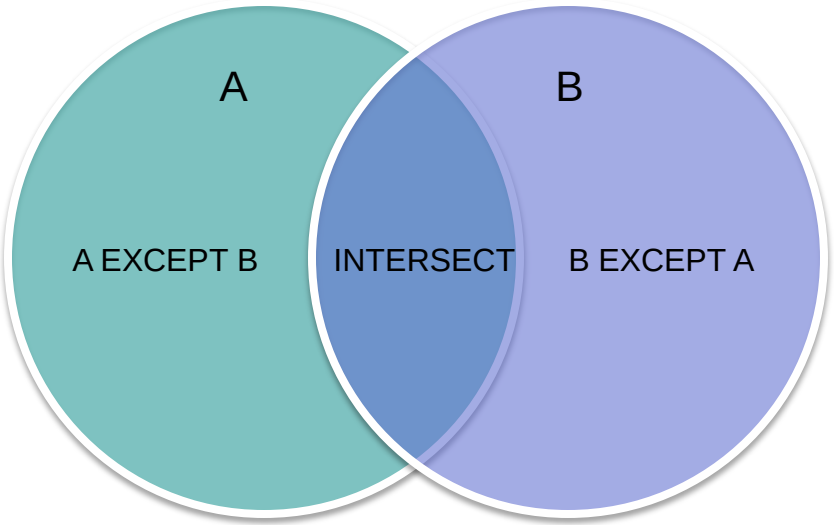
```
create table nb_sample.nb_emp like nb_sample.employee ;

insert into nb_sample.nb_emp
with lst_min as ( select *
                  from nb_sample.employee
                  where workdept = 'A00' )
select * from lst_min ;
```

OPÉRATEURS ENSEMBLISTES

UNION, INTERSECT, EXCEPT

- Syntaxe

```
SELECT …
UNION / UNION ALL / EXCEPT / INTERSECT
SELECT …
```



- Opérateurs
  - UNION
    - Sans doublons
  - UNION ALL
    - Avec doublons
  - EXCEPT
    - Les lignes du 1er ensemble non présentes dans le 2nd
  - INTERSECT
    - Les lignes communes

UNION, INTERSECT, EXCEPT

- Exemple
  - UNION

```
select *
from nb_sample.employee
where workdept = 'B01'
union all
select *
from nb_sample.employee
where sex = 'M'
order by empno
for read only ;
```

WORKDEPT	SEX	EMPNO	FIRSTNME	MIDINIT	LASTNAME
B01	M	000020	MICHAEL	L	THOMPSON
B01	M	000020	MICHAEL	L	THOMPSON
E01	M	000050	JOHN	B	GEYER
D11	M	000060	IRVING	F	STERN
E21	M	000100	THEODORE	Q	SPENSER
A00	M	000110	VINCENZO	G	LUCCHESSI
A00	M	000120	SEAN		O'CONNELL
D11	M	000150	BRUCE		ADAMSON
D11	M	000170	MASATOSHI	J	YOSHIMURA
D11	M	000190	JAMES	H	WALKER
D11	M	000200	DAVID		BROWN
D11	M	000210	WILLIAM	T	JONES

  - EXCEPTION
    - Vide

WORKDEPT	SEX	EMPNO	FIRSTNME	MIDINIT	LASTNAME

  - INTERSECT

WORKDEPT	SEX	EMPNO	FIRSTNME	MIDINIT	LASTNAME
B01	M	000020	MICHAEL	L	THOMPSON

RANG ET PARTITIONNEMENT

Score et numérotation des lignes résultats

- Fonctions OLAP
  - ROW_NUMBER() OVER()
    - Numérotation des lignes suivant un ordre spécifique
  - RANK() OVER()
    - Affectation d'un score suivant des critères
    - Egalité possible
  - DENSE_RANK() OVER()
    - Affectation d'un score suivant des critères
    - Egalité possible
    - Les rangs sont consécutifs

Score et numérotation des lignes résultats

- Exemple
  - Affectation d'un rang en fonction de l'ancienneté des employés

```
select row_number() over(order by years desc) as "row_number",
       rank()       over(order by years desc) as "rank",
       dense_rank() over(order by years desc) as "dense_rank",
       a.*
from nb_sample.staff a
order by years desc
for read only ;
```

row_number	rank	dense_rank	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	1	1	60	Quigley	38	Sales	-	16808.30	650.25
2	1	1	80	James	20	Clerk	-	13504.60	128.20
3	1	1	120	Naughton	38	Clerk	-	12954.75	180.00
4	1	1	200	Scoutten	42	Clerk	-	11508.60	84.20
5	5	2	310	Graham	66	Sales	13	21000.00	200.30
6	6	3	260	Jones	10	Mgr	12	21234.00	-
7	7	4	50	Hanes	15	Mgr	10	20659.80	-
8	7	4	210	Lu	10	Mgr	10	20010.00	-
9	7	4	290	Quill	84	Mgr	10	19818.00	-
10	10	5	270	Lea	66	Mgr	9	18555.50	-
11	10	5	280	Wilson	66	Sales	9	18674.50	811.50
12	12	6	20	Pernal	20	Sales	8	18171.25	612.45
13	12	6	190	Sneider	20	Clerk	8	14252.75	126.50

Score et numérotation des lignes résultats

- Exemple
  - Affectation d'un rang en fonction de l'ancienneté des employés, par département

```
select row_number()
            over(partition by dept order by years desc)
          as "row_number",
       rank() over(partition by dept order by years desc)
          as "rank",
       dense_rank()
            over(partition by dept order by years desc)
          as "dense_rank",
       a.*
from nb_sample.staff a
order by dept asc, years desc
for read only ;
```

Score et numérotation des lignes résultats

row_number	rank	dense_rank	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	1	1	260	Jones	10	Mgr	12	21234.00	-
2	2	2	210	Lu	10	Mgr	10	20010.00	-
3	3	3	160	Molinare	10	Mgr	7	22959.20	-
4	4	4	240	Daniels	10	Mgr	5	19260.25	-
1	1	1	50	Hanes	15	Mgr	10	20659.80	-
2	2	2	70	Rothman	15	Sales	7	16502.83	1152.00
3	3	3	110	Ngan	15	Clerk	5	12508.20	206.60
4	4	4	170	Kermisch	15	Clerk	4	12258.50	110.10
1	1	1	80	James	20	Clerk	-	13504.60	128.20
2	2	2	20	Pernal	20	Sales	8	18171.25	612.45
3	2	2	190	Sneider	20	Clerk	8	14252.75	126.50
4	4	3	10	Sanders	20	Mgr	7	18357.50	-
1	1	1	60	Quigley	38	Sales	-	16808.30	650.25

JOINTURES

Types de jointure

- {INNER} JOIN
  - Jointure intérieure, ou naturelle
  - Produit cartésien réduit à une sélection
- {OUTER} JOIN
  - Jointure externe
  - LEFT {OUTER}
  - RIGHT {OUTER}
  - FULL {OUTER}
- EXCEPTION JOIN
  - Critère d'inexistence. Externe par définition
  - LEFT EXCEPTION
  - RIGHT EXCEPTION

Jointure naturelle

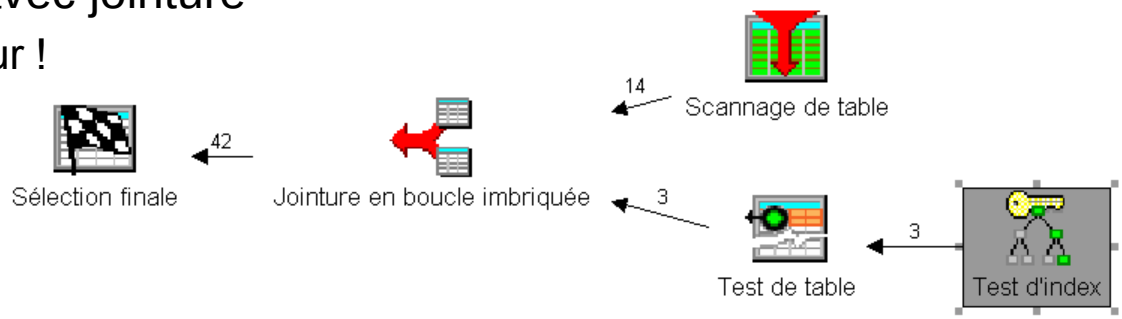
- Exemple
  - Affichage des informations des employés et de leur département
    - Uniquement si les deux informations existent

```
select *
from nb_sample.employee a
join nb_sample.dept b
on a.workdept = b.deptno ;
```

  - Equivalent à (produit cartésien)

```
select *
from nb_sample.employee, nb_sample.dept
where workdept = deptno ;
```

  - Le plan d'exécution est le même dans les deux cas
    - Préférez la syntaxe avec jointure
      - Comme l'optimiseur !



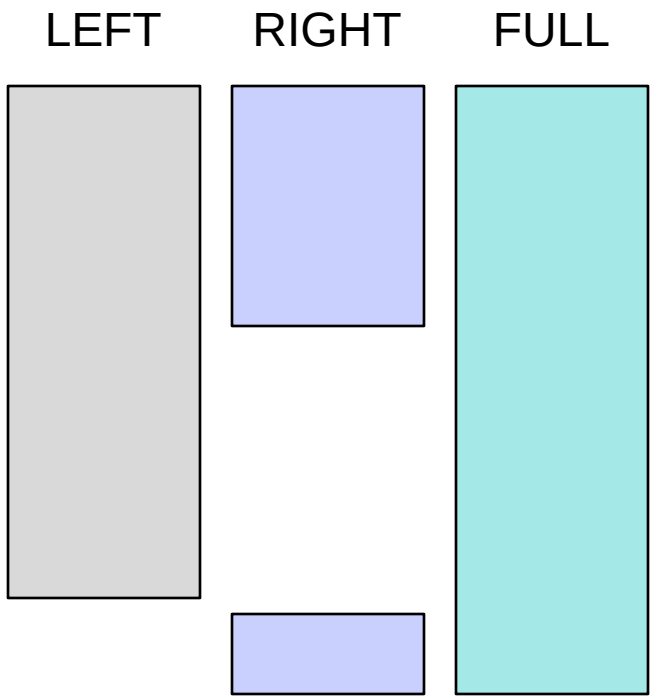
Jointure externe

- Permette d'afficher tous les enregistrements d'une table et les enregistrements de l'autre s'ils existent (NULL sinon)
  - LEFT, RIGHT
    - Tous les enregs de la table à gauche/droite du JOIN
  - FULL
    - Tous les enregs des deux tables
    - FULL ⇔ RIGHT UNION LEFT
  - Exemple
    - Liste des employés, avec leur job dans l'équipe

```
select a.empno, a.firstnme,
       a.job, b.name, a.workdept,
       b.dept, b.job
from                  nb_sample.employee a
     full outer join nb_sample.staff b
                      on a.empno = b.id
order by a.empno
for read only ;
```

Jointure externe

EMPNO	FIRSTNME	JOB	NAME	WORKDEPT	DEPT	JOB
000270	MARIA	CLERK	Lea	D21	66	Mgr
000280	ETHEL	OPERATOR	Wilson	E11	66	Sales
000290	JOHN	OPERATOR	Quill	E11	84	Mgr
000300	PHILIP	OPERATOR	Davis	E11	84	Sales
000310	MAUDE	OPERATOR	Graham	E11	66	Sales
000320	RAMLAL	FIELDREP	Gonzales	E21	66	Sales
000330	WING	FIELDREP	Burke	E21	66	Clerk
000340	JASON	FIELDREP	Edwards	E21	84	Sales
200010	DIAN	SALESREP	-	A00	-	-
200120	GREG	CLERK	-	A00	-	-
200140	KIM	ANALYST	-	C01	-	-
200170	KIYOSHI	DESIGNER	-	D11	-	-
200220	REBA	DESIGNER	-	D11	-	-
200240	ROBERT	CLERK	-	D21	-	-
200280	EILEEN	OPERATOR	-	E11	-	-
200310	MICHELLE	OPERATOR	-	E11	-	-
200330	HELENA	FIELDREP	-	E21	-	-
200340	ROY	FIELDREP	-	E21	-	-
-	-	-	O'Brien	-	38	Sales
-	-	-	James	-	20	Clerk
-	-	-	Gafney	-	84	Clerk



Jointure exception

- Permet de sélectionner les enregistrements d'une table sans correspondance dans l'autre table
  - EXCEPTION JOIN ⇔OUTER JOIN .. WHERE colonne IS NULL
  - EXCEPTION JOIN ⇔sous-requête WHERE NOT EXISTS…
  - Exemple
    - L'ensemble des employés non affectés à une équipe

```
select a.empno, a.firstnme,
       a.job
from                        nb_sample.employee a
     left exception join    nb_sample.staff b
                            on a.empno = b.id
for read only ;
```

EMPNO	FIRSTNME	JOB
200010	DIAN	SALESREP
200120	GREG	CLERK
200140	KIM	ANALYST
200170	KIYOSHI	DESIGNER
200220	REBA	DESIGNER
200240	ROBERT	CLERK
200280	EILEEN	OPERATOR
200310	MICHELLE	OPERATOR
200330	HELENA	FIELDREP
200340	ROY	FIELDREP

    - L'ensemble des personnes d'une équipe inexistant des les employés

```
select b.id, b.name, b.dept
from                         nb_sample.employee a
     right exception join    nb_sample.staff b
                             on a.empno = b.id
for read only ;
```

ID	NAME	DEPT
40	O'Brien	38
80	James	20
350	Gafney	84

SOUS-REQUÊTES

Sous-requêtes

- Les sous-requêtes peuvent être utilisées
  - Dans la clause SELECT en tant que valeur
    - 0 ou 1 valeur
  - Dans la clause WHERE en tant que test
- Exemple
  - Liste des employés pour les départements de plus de 5 employés, avec indication du salaire maximum dans l'entreprise

```
select a.empno, a.firstnme, a.workdept, a.salary,
       ( select max( b.salary ) from nb_sample.employee b )
          as "Salaire max"
from nb_sample.employee a
where a.workdept in ( select c.workdept
                      from nb_sample.employee c
                      group by c.workdept
                      having count(*) > 5 )
for read only ;
```

Tuples (n-uplets)

- Les sous-requêtes permettent également de sélectionner des tuples
- Exemple
  - Liste des objets utilisés à la plus grande date d'utilisationselect a.*

```
from nbonnet.qdevtools a
where ( a.odccen, a.odcdat ) =
      (select b.odccen, b.odcdat
       from nbonnet.qdevtools b
       order by b.odccen desc , b.odcdat desc
       fetch first 1 rows only)
for read only ;
```

- Très utile pour dé-doublonner des enregistrements
  - RRN

NONE, ANY, ALL, n

- En dehors des clauses classiques
  - EXISTS, NOT EXISTS
  - IN, NOT IN
- Les clauses suivantes sont utiles
  - ANY
    - Au moins une ligne résultat doit satisfaire la condition
  - ALL
    - Toutes les lignes résultat doivent satisfaire la condition
  - N
    - Indique un nombre d’occurrences spécifiques

NONE, ANY, ALL, n

- Exemples
  - Liste des employés qui gagnent plus que tous les autres

```
select a.empno, a.firstnme, (a.salary+a.bonus+a.comm) as revenu
from nb_sample.employee a
where ( a.salary + a.bonus + a.comm ) > all
      ( select b.salary + b.bonus + b.comm
        from nb_sample.employee b
        where a.empno <> b.empno )
order by 3 desc
for read only ;
```

  - Résultat
    - 1 unique employé : « le » président

EMPNO	FIRSTNME	REVENU
000010	CHRISTINE	58392.00

NONE, ANY, ALL, n

- Liste des employés tel qu'au moins 1 autre employé gagne moins

```
where ( a.salary + a.bonus + a.comm ) < any
      ( select b.salary + b.bonus + b.comm
        from nb_sample.employee b
        where a.empno <> b.empno )
```

  - Résultat
    - Tous les employés sauf le président

- Liste des employés tel que au moins 35 autres employés gagnent autant ou moins

```
where 35 <= ( select count(*)
              from nb_sample.employee b
              where a.empno <> b.empno and
                    (b.salary + b.bonus + b.comm) <=
                       (a.salary + a.bonus + a.comm))
```

  - Résultat
    - Les 7 plus gros salaires
      - 42 employés au total

EMPNO	FIRSTNME	REVENU
000010	CHRISTINE	58392.00
200010	DIAN	52142.00
000110	VINCENZO	51492.00
000020	MICHAEL	45680.00
000050	JOHN	44510.40
000030	SALLY	42416.00
000070	EVA	40052.30

AUTRES

SKIP LOCKED DATA

- Depuis le 6.1
  - Possibilité d'ignorer les lignes verrouillées
    - Par d'autres transactions SQL
    - Par des programmes RPG, COBOL …

```
select *
from nb_sample.staff
skip locked data
for read only ;
```

INSERT multiples

- Permet de remplacer

```
insert into nb_sample.sales values( current date, 'JOHN',
  'Rhône', 6 ) ;
insert into nb_sample.sales values( current date, 'ROGER',
  'Paris', 8 ) ;
```

- Par
  - « Blocked insert »
    - Possible également en SQL embarqué par DS DIM(x)

```
insert into nb_sample.sales values
( current date, 'JOHN', 'Rhône', 6 ) ,
( current date, 'ROGER', 'Paris', 8 ) ;
```

- Avantages
  - Plus concis
  - Plus performant

Limiter les SELECT à n valeurs maximales

- Permet de s'assurer que l'on a 1 ou n lignes en retour d'un SELECT
  - Exemple : les 3 employés avec le plus petit bonus

```
select *
from nb_sample.employee
order by bonus asc, empno asc
fetch first 3 rows only
for read only;
```

  - Utile lorsque l'on doit obligatoirement s'assurer que l'on a 1 unique valeur

```
update nb_sample.employee a
set a.bonus = (select bonus
                from nb_sample.employee b
                where b.workdept = a.workdept
                order by bonus desc
                fetch first 1 rows only )
where empno = 260 ;
```

Utiliser des données constantes (1/2)

- Pour utiliser dans des requêtes des données qui n'existent pas en base

```
with tmp ( tranche, min, max) as
         ( values ('Tranche 1', 0     , 15000 ),
                  ('Tranche 2', 15001 , 20000),
                  ('Tranche 3', 21000 , 50000) )
select * from tmp
for read only ;
```

TRANCHE	MIN	MAX
Tranche 1	0	15000
Tranche 2	15001	20000
Tranche 3	21000	50000

Utiliser des données constantes (2/2)

- Affectation d'une tranche de salaire à chaque employé

```
with tmp ( tranche, min, max) as
         ( values ('Tranche 1', 0     , 15000 ),
                  ('Tranche 2', 15001 , 20000),
                  ('Tranche 3', 21000 , 50000) )
select a.name, a.salary, a.comm , b.tranche
from nb_sample.staff a
join tmp b on
    ( a.salary + a.comm ) between b.min and b.max
order by salary desc
for read only ;
```

NAME	SALARY	COMM	TRANCHE
Graham	21000.00	200.30	Tranche 3
Wilson	18674.50	811.50	Tranche 2
Pernal	18171.25	612.45	Tranche 2
O'Brien	18006.00	846.55	Tranche 2
Koonitz	18001.75	1386.70	Tranche 2
Edwards	17844.00	1285.00	Tranche 2
Smith	17654.50	992.80	Tranche 2
Gonzales	16858.20	844.00	Tranche 2
Quigley	16808.30	650.25	Tranche 2
Rothman	16502.83	1152.00	Tranche 2
Davis	15454.50	806.10	Tranche 2
Wheeler	14460.00	513.30	Tranche 1
Sneider	14252.75	126.50	Tranche 1
James	13504.60	128.20	Tranche 1
Lundquist	13369.80	189.65	Tranche 1
Gafney	13030.50	188.00	Tranche 1

Valeurs nulles et division par 0 (1/3)

- De façon générale, remplacer une valeur par une autre

```
CASE …
   WHEN cond1 THEN val1
   WHEN cond2 {THEN val2}
  {ELSE val3}
END
```

  - Exemple : fréquence d'utilisation des objets de QPADEVTOOLS

```
select
   odobnm, odobtp, oducnt,
   current date - cmdy( odccen, odcdat ) as "Nb j. créa",
  case when oducnt = 0
       then 0
       else
           int((current date - cmdy(odccen, odcdat)) / oducnt)
       end as "Fréq. util."
from nbonnet.qdevtools
for read only ;
```

Valeurs nulles et division par 0 (2/3)

- On peut aussi écrire

```
...
case oducnt when 0
            then 0
            when 1
            then 1
            else 2
End
...
```

ODOBNM	ODOBTP	ODUCNT	Nb j. créa	Fréq. util.	Fréq. util. (sans CASE)
QRBUTIL	*SRVPGM	3	50512	16837	16837
QRSEUTILS	*SRVPGM	232	50512	217	217
RSEDECW	*SRVPGM	51	50512	990	990
EVFCMSGF	*MSGF	40	50605	1265	1265
QDTSMSG	*MSGF	143	50605	353	353
QLBLMSG	*MSGF	81	50605	624	624
QLNCMSG	*MSGF	78	50605	648	648
QLSCMSG	*MSGF	1	50605	50605	50605
QRPGLEMSG	*MSGF	423	50605	119	119
QRPGMSG	*MSGF	102	50605	496	496
QRSEMSGS	*MSGF	52	50605	973	973
EVFADSPC	*FILE	0	50519	0	+++++++++++++
EVFCICFF00	*FILE	0	50512	0	+++++++++++++
EVFCLOGO	*FILE	0	50519	0	+++++++++++++
H	*FILE	1	50512	50512	50512
QPZA000037	*FILE	1	40808	40808	40808
QRSESVR	*FILE	16	50519	3157	3157
CRTBNDC	*CMD	18	50525	2806	2806
CRTBNDCBL	*CMD	5	50512	10102	10102

Valeurs nulles et division par 0 (3/3)

- Prendre la 1ère valeur non null

```
IFNULL( a , b )
```

  - Retourne a si non null, b sinon (b peut être null)

```
COALESCE( a , b { ... } )
```

  - Retourne la 1ère valeur non nulle (null si tous les arguments sont null)
    - Recommandée
  - Exemple
    - La commission de chaque employé en pourcentage du salaire

```
select name, salary, comm,
       cast( coalesce( comm , 0 ) / salary * 100
             as dec(5 , 2 ) )
          as "com % salary"
from nb_sample.staff
for read only ;
```

    - Sinon, on se retrouve avec null
      - null / n -> null

NAME	SALARY	COMM	com % salary
Sanders	18357.50	-	0.00
Pernal	18171.25	612.45	3.37
Marenghi	17506.75	-	0.00
O'Brien	18006.00	846.55	4.70
Hanes	20659.80	-	0.00
Quigley	16808.30	650.25	3.86
Rothman	16502.83	1152.00	6.98
James	13504.60	128.20	0.94
Koonitz	18001.75	1386.70	7.70
Plotz	18352.80	-	0.00
Ngan	12508.20	206.60	1.65
Naughton	12954.75	180.00	1.38
Yamaguchi	10505.90	75.60	0.71
Fraye	21150.00	-	0.00

Tri personnalisé

- Le CASE permet également de personnaliser des tris
  - Afficher l'ensemble des employés par commissions de la plus importante à la moins important ; les employés ayant une commission entre 200 et 300 doivent apparaitre en 1er

```
select name, salary, coalesce( comm , 0 )
from nb_sample.staff
order by case
           when comm between 200 and 300 then 9999999
           else coalesce( comm , 0 )
         end desc
for read only ;
```

NAME	SALARY	00003
Ngan	12508.20	206.60
Abrahams	12009.75	236.50
Graham	21000.00	200.30
Koonitz	18001.75	1386.70
Edwards	17844.00	1285.00
Rothman	16502.83	1152.00
Smith	17654.50	992.80
O'Brien	18006.00	846.55
Gonzales	16858.20	844.00
Wilson	18674.50	811.50
Davis	15454.50	806.10

Extraire des données balisées par deux caractères

- L'emplacement des données n'est pas connu à priori

```
...<monInfo>l'info à extraire</monInfo>...
```

- Exemple

```
with tmp (data) as ( values('des trucs avant ... <monInfo>l'info
  à extraire</monInfo>... des trucs après'))
select a.data ,
       locate('<monInfo>' , a.data) as "Pos début",
       locate('</monInfo>', a.data) as "Pos fin",
       substr(a.data,
              locate('<monInfo>', a.data) +
                 length('<monInfo>'),
              locate('</monInfo>', a.data ) -
                 locate('<monInfo>', a.data) -
                 length('<monInfo>'))
         as "Valeur"
from tmp a ;
```

DATA	Pos début	Pos fin	Valeur
des trucs avant ... <monInfo>l'in...	21	47	l'info à extraire

Comptage horizontal

- Peu pratique et peu performant
  - Nécessite de connaitre les valeurs à traiter
- Compter le nombre d'employés par sexe et par département

```
select workdept,
       count(*) as "total",
       sum(case sex when 'F' then 1 else 0 end) as "Femme",
       sum(case sex when 'M' then 1 else 0 end) as "Homme",
       sum(case when sex not in ('F','M')
                then 1 else 0 end ) as "Inconnu"
from nb_sample.employee
group by workdept
order by workdept
for read only;
```

WORKDEPT	total	Femme	Homme	Inconnu
A00	5	2	3	0
B01	1	0	1	0
C01	4	4	0	0
D11	11	4	7	0
D21	7	3	4	0
E01	1	0	1	0
E11	7	5	2	0
E21	6	1	5	0

Tester le contenu d'une zone

- En cas de reprise de données par exemple, il est nécessaire de s'assurer qu'une zone alphanumérique ne contient que des données numériques

```
select a.*
from nb_sample.employee a
where translate( a.empno, ' ' , '1234567890' ) <> ''
for read only ;
```

  - Ne fonctionne pas si des blancs sont entre les chiffres, avec les symboles décimaux, monétaires …

Nous contacter

- Par mail
  - nbonnet@gaia.fr
  - contact@gaia.fr
- Nos sites
  - www.gaia.fr
  - www.know400.fr
  - www.as400.fr