SQL

1. Quelques fonctions

➪ *RI pages 91 – 98*

```
SELECT GETDATE()
SELECT DATEPART (ww , GETDATE())-- semaine de l'année
SELECT datepart (yyyy,GETDATE()) -- année
-- identique à
SELECT YEAR('22/02/2012')
SELECT GETDATE() + 1 -- comprend +1 jour
SELECT DATEADD(yy,1,getdate()) -- pour ajouter autre chose, des années par exemple
SELECT DATEDIFF(hh,'22/02/2012','23/02/2012') -- différnece, en heures, entre 2 dates
-- formatage d'affichage
SELECT CONVERT(varchar,getdate(),103)
```

2. Traduction de l'algèbre relationnelle : le SELECT

➪ *RI page 106*

```
-- la projection étape 1 : calcul élémentaire
SELECT * FROM EMPLOYES
SELECT nom, prenom, dateembauche, salaire, codeservice FROM EMPLOYES
-- lors d'une projection, je peux réaliser des calculs élémentaires !
SELECT      UPPER(nom),
            ISNULL(prenom,''),
            CONVERT(varchar,dateembauche,103),
            salaire*1.1,
            codeservice
FROM EMPLOYES

-- lors d'une projection, je peux créer de nouvelles colonnes, et donc leur donner un
nom -- ça s'appelle des alias de colonnes
SELECT      nom = UPPER(nom), -- méthode 1
            ISNULL(prenom,'') prenom , -- méthode 2
            CONVERT(varchar,dateembauche,103) as "date d'embauche",  --méthode 3
            salaire*1.1 "salaire augmenté",
            codeservice  service
FROM EMPLOYES

-- RESTRICTION = clause WHERE
--*********************************
-- seulement les employés du service informatique
SELECT      nom = UPPER(nom), -- méthode 1
            ISNULL(prenom,'') prenom , -- méthode 2
            CONVERT(varchar,dateembauche,103) as "date d'embauche",  --méthode 3
            salaire*1.1 "salaire augmenté",
            codeservice  service
FROM EMPLOYES
WHERE codeservice = 'INFOR'
-- 2 conditions
AND nom LIKE 'R%'
-- une troisième : AND "salaire augmenté" = 2200 -- ne fonctionne pas, il ne connait
pas l'alias de colonne
AND salaire*1.1 = 2200
```

```
--Projection étape 2 : calculs d'agrégat
--****************************************
SELECT nom, codeservice from employes
SELECT  codeservice from employes -- par défaut, pas de regroupement !!!

-- il faut lui spécifier comment on veut regrouper
SELECT codeservice from employes
GROUP BY codeservice

-- lors d'un regroupement ,on peut faire des calculs d'agrégat
SELECT codeservice, COUNT(codeemp) nbemp from employes GROUP BY codeservice

-- pour avoir les codes service différent : DISTINCT
SELECT DISTINCT codeservice from employes -- ici, on n'a pas de regroupement !
                                          -- donc pas de calcul d'agrégat.
-- nombre d'employés PAR service => GROUPY BY SERVICE
-- nombre d'employés total
SELECT COUNT(*) FROM EMPLOYES

SELECT codeservice, COUNT(codeemp) nbemp, nom
from employes GROUP BY codeservice, nom
-- dans le group by, on doit mettre TOUS les champs de la projection, SAUF ceux qui
-- font partie d'un calcul d'agregat


-- Restriction sur un calcul d'agrégat : HAVING
--**********************************************
SELECT codeservice,
COUNT(codeemp) nbemp,
SUM(salaire) "masse salariale",
AVG(salaire) moyenne,
MAX(salaire) "plus grand salaire"
from employes
WHERE codeservice IN ('DIRGE','INFOR')-- + 1 condition
--AND AVG(salaire) >3000 ne peut pas le faire
GROUP BY codeservice
--2eme condition, ceux qui ont une moyenne sup à 3000
HAVING  AVG(salaire) >3000

--PRODUIT CARTESIEN
--**********************
-- je veux afficher le nom de l'employé et le libellé du service
SELECT * FROM EMPLOYES , SERVICES

-- nouvelle syntaxe
SELECT * FROM EMPLOYES CROSS JOIN SERVICES

-- JOINTURE
--*****************
-- Une jointure est une restriction sur le produit cartesien
SELECT * FROM EMPLOYES , SERVICES
WHERE EMPLOYES.codeservice=SERVICES.codeservice

-- nouvelle syntaxe
SELECT * FROM EMPLOYES INNER JOIN SERVICES ON Employes.codeservice =
Services.codeservice
--WHERE nom = 'toto'
```

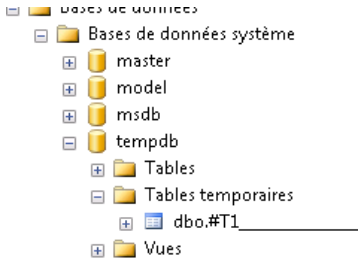
```
--je veux la liste des employés, le libelle de leur service
--et le nombre de jours acquis en 2005
SELECT nom, libelle, nbjoursacquis , employes.codeservice
FROM EMPLOYES INNER JOIN SERVICES ON Employes.codeservice = Services.codeservice
                        INNER JOIN CONGES on conges.codeemp = employes.codeemp
WHERE annee = 2005
-- je rajoute le nombre de jours pris en 2005
-- etape 1 : par mois
SELECT *
FROM EMPLOYES INNER JOIN SERVICES ON Employes.codeservice = Services.codeservice
                        INNER JOIN CONGES on conges.codeemp = employes.codeemp
                        INNER JOIN CONGES_MENS ON conges_mens.annee = conges.annee
                                                        AND conges_mens .codeemp =
conges.codeemp
WHERE conges.annee = 2005

-- on peut utiliser des alias de table pour alléger la requete !
SELECT e.nom, s.*, c.annee
FROM EMPLOYES e INNER JOIN SERVICES s ON e.codeservice = s.codeservice
                        INNER JOIN CONGES c on c.codeemp = e.codeemp
                        INNER JOIN CONGES_MENS cm ON cm.annee = c.annee
                                                        AND cm .codeemp = c.codeemp
WHERE c.annee = 2005


-- etape 2 : au total
SELECT services.codeservice, libelle, nom, nbjoursacquis, SUM(nbjourspris)
FROM EMPLOYES INNER JOIN SERVICES ON Employes.codeservice = Services.codeservice
                        INNER JOIN CONGES on conges.codeemp = employes.codeemp
                        INNER JOIN CONGES_MENS ON conges_mens.annee = conges.annee
                                                        AND conges_mens .codeemp =
conges.codeemp
WHERE conges.annee = 2005
GROUP BY services.codeservice, libelle, nom, nbjoursacquis

--  Le tri
--********************************
SELECT codeemp, nom, salaire, codeservice from employes
ORDER BY codeservice -- ASC par défault ascendant
SELECT codeemp, nom, salaire, codeservice from employes
ORDER BY codeservice DESC -- descendant
SELECT codeemp, nom, salaire, codeservice from employes
ORDER BY codeservice ASC, nom DESC -- 2 tris

SELECT codeemp, nom, salaire *2 "salaire doublé", codeservice from employes
ORDER BY "salaire doublé", 2 -- le tri c'est la dernière étape, donc on peut utiliser
les alias de colonnes
```

SELECT – avancé

```
-- afficher les n premières lignes
SELECT TOP 1 salaire from employes
order by salaire


-- REFLEXION
-- AUTO-JOINTURE
-- afficher la liste des employés avec le nom de leur chef
SELECT * FROM EMPLOYES
SELECT emp.nom, emp.prenom , chef.nom chef
FROM EMPLOYES emp INNER JOIN EMPLOYES chef ON emp.codechef = chef.codeemp
-- on est obligé d'utiliser des alias...

-- il manque un employé !!!

--même problème :
--  afficher la liste des services ET le nom de leurs employés... S'IL EN ONT !!
Select *
FROM SERVICES INNER JOIN EMPLOYES ON services.codeservice = employes.codeservice
-- il manque 3 services !! => ceux qui n'ont pas d'employé

-- LA JOINTURE EXTERNE => OUTER JOIN
--**********************************
Select *
FROM SERVICES LEFT OUTER JOIN EMPLOYES ON services.codeservice = employes.codeservice

Select *
FROM EMPLOYES RIGHT OUTER JOIN SERVICES ON services.codeservice = employes.codeservice

-- afficher la liste des employés avec le nom de leur chef
-- MEME S'ILS N'ONT PAS DE CHEF !
SELECT emp.nom, emp.prenom , chef.nom chef
FROM EMPLOYES emp LEFT OUTER JOIN EMPLOYES chef ON emp.codechef = chef.codeemp


--UNION
--************
select  codeservice, libelle from services
UNION ALL
select codeservice,'employé' from employes


-- sauvegarde de requete
-- création de table
--*****************************
SELECT AVG(salaire) moy INTO TMOyenne FROM EMPLOYES
-- le contenu de cette table ne sera pas modifié dynamiquement
select * from tmoyenne -- elle existe "definitivement"
DROP TABLE tmoyenne
-- quan don veut simplement sauvegarder le résultat d'un calcul temporairement
--=> tables temporaires : # devant le nom
SELECT AVG(salaire) moy INTO #TMOyenne FROM EMPLOYES
select * from #tmoyenne
-- pas besoin de la supprimer car le sera automatiquement lors de ma déconnexion
-- quels sont les employés dont le salaire est supérieur à la moyenne
SELECT nom, salaire FROM EMPLOYES CROSS JOIN #tmoyenne
WHERE salaire > moy
```

RI page 123

Elles sont utilisables pendant la durée de la session si # (on peut les voir dans l'explorateur)


Bases de données système
master
model
msdb
tempdb
Tables
Tables temporaires
dbo.#T1
Vues

3. Statistiques

```
-- tables CTE
--**************
-- table instantanée : leur durée de vie est l'exécution de la requete
-- refaire l'affichage des employés et le nom de leur chef
```

```
WITH CTEChef (codeDuChef, NomDuChef) AS (
SELECT codeemp, nom from employes)
SELECT * from ctechef

WITH CTEChef (codeDuChef, NomDuChef) AS (
SELECT codeemp, nom from employes)
SELECT * from ctechef RIGHT OUTER JOIN EMPLOYES ON
                  codechef = codeduchef
```

```
-- lignes statistiques: COMPUTE
--*****************************
```

```
select codeservice, salaire from EMPLOYES
order by codeservice
COMPUTE SUm(salaire)
COMPUTE SUM(salaire) BY codeservice
```

⇨ Refaire Boutaud avec Compute

rollup et cube

RI page 127

Les lignes statistiques sont intégrées à la requête.

Par employé et par année

- ⇨ Le nb de jours pris
- ⇨ Le total cumulé

\+ Total cumulé pour tous

```
select  nom, c.annee, sum(nbjourspris) 'jours pris par an' from employes e, conges c,
conges_mens cm
where e.codeemp = c.codeemp and
c.codeemp = cm.codeemp and c.annee = cm.annee and c.annee = 2006
group by nom, c.annee
```

puis

```
select  nom, c.annee, sum(nbjourspris) 'jours pris par an' from employes e, conges c,
conges_mens cm
where e.codeemp = c.codeemp and c.codeemp = cm.codeemp and c.annee = cm.annee and
c.annee = 2006 group by nom, c.annee
with rollup
```

- ⇨ Rajoute un calcul par employé puis un calcul final

Remplacer c.annee par

```
snull(convert(char,c.annee), 'total cumulé')
```

Essai expérimental

```
select  nom, isnull(convert(char,c.annee), 'total cumulé'),
sum(nbjourspris) 'jours pris par an' , max(nbjourspris) 'max' , count(nom)
from employes e, conges c, conges_mens cm
where e.codeemp = c.codeemp and
c.codeemp = cm.codeemp and c.annee = cm.annee and c.annee = 2006
group by nom, c.annee
with rollup
```

Remplacer par WITH CUBE : une ligne ne plus

4. Sous-requêtes et transactions

1. Sous requêtes

```
-- requetes imbriquées
-------------------------
-- les employés qui ont un salaire supérieur à la moyenne
SELECT AVG(salaire) FROM EMPLOYES
-- on peut stocker dans une variable
--declare @nomvar decimal(6,2)
--set @nomvar = 3000.4
select nom, prenom, codeservice, salaire FROM EMPLOYES
WHERE salaire > (SELECT AVG(salaire) FROM EMPLOYES)
-- un select dans un select

-- sous requetes utilisables dans de nombreux cas
select nom, prenom, salaire ,
            (SELECT AVG(salaire) FROM EMPLOYES) moyenne
from employes


-- quels sont les services sans employes ?
select codeservice from services
WHERE codeservice NOT IN(
select distinct codeservice from employes)

-- sous requetes correlées
-- exemple la liste des employés dont le salaire
--est supérieur à la moyenne des salaire du service
select nom, salaire , codeservice from employes
SELECT AVG(salaire) moy, codeservice FROM EMPLOYES
GROUP BY codeservice -- sera à exécuter à chaque ligne d'employes

select nom, salaire , codeservice from employes emp
WHERE salaire > (SELECT AVG(salaire) FROM EMPLOYES s where emp.codeservice =
s.codeservice)


-- sous-requete utilisable comme un table
-- ---------------------------------------
--je veux dans un même résultat les employés, leur salaire ET la masse salariale
select * from
(select nom, salaire from employes ) R1 CROSS JOIN
(select sum(salaire) masse from employes ) R2


INSERT INTO EMPLOYES (codeemp, nom, salaire, codeservice) VALUES (newid(),
'NOUVEAU',4520, 'INFOR')
-- je veux que le chef de NOUVEAU soit BIGBOSS…….. Mais je me laisse la possibilité de
me tromper !
```

2. Transactions

➪ ***RI page 148***

➪ Pour le moment, les transactions ont été automatiquement validées (par défaut). Là nous allons les gérer nous même.

BEGIN TRAN nom_trans
SAVE TRAN point-de-contrôle
ROLLBACK TRAN nom_trans => revient, soit au point de contrôle, soit au begin.
COMMIT TRAN nom_trans => valide tout

```
--*    LES TRANSACTIONS
--**********************************
INSERT INTO EMPLOYES (codeemp, nom, salaire, codeservice) VALUES (newid(),
'NOUVEAU',4520, 'INFOR')
-- je veux que le chef de NOUVEAU soit BIGBOSS

BEGIN TRAN updateNouveau
--là je suis dans une transaction : toutes les instructions du DML ensuite, pourront
être annulées
UPDATE EMPLOYES SET codechef = (SELECT codeemp FROM EMPLOYES where nom = 'BIGBOSS')
WHERE nom = 'NOUVEAU'
SELECT * from employes
-- c'est ok, je valide
COMMIT TRAN updateNouveau

BEGIN TRAN SUPPServices
-- supprimer les services qui n'ont pas d'employé
DELETE FROM SERVICES
WHERE  codeservice NOT IN ( select codeservice from employes)
select * from services
-- les autres utilisateurs ne peuvent pas accéder aux informations : verrou

-- pas ok, j'annule
ROLLBACK TRAN SUPPServices
```