

# Les tables de résumé en DB2 (MQT).

par Jean-Alain Baeyens ([autres articles](#))

Date de publication : 24 mai 2008

Dernière mise à jour :

Cet article a pour vocation d'expliquer le fonctionnement des tables de résumé aussi appelées MQT. Cette technologie permet essentiellement d'accélérer le reporting. Disponible sous Linux et Windows depuis la version 5.2, elle est également disponible sur Z/Os depuis la version 8 et sur ISeries depuis la version 5.3 au moins.

I - Généralités.....	3
II - La syntaxe.....	3
III - Maintenir les données en permanence synchronisées.....	3
IV - Différer la synchronisation entre tables de données et tables de résumés.....	4
V - La clause QUERY OPTIMIZATION.....	4
VI - La clause MAINTAINED BY.....	4
VII - Exemples.....	4
V-A - Exemple 1: Avec un simple "GROUP BY".....	4
V-B - Exemple 2: Avec rafraichissement immédiat.....	5
V-C - Exemple 3: Avec "GROUP BY CUBE".....	5
V-D - Exemple 4: La table n'est pas le résultat d'un "GROUP BY".....	6
V-E - Exemple 5: Avec une sous-requête.....	6

## I - Généralités

Les tables de résumé permettent de conserver le résultat d'un "SELECT" réalisé sur vos tables de données. Lors du reporting, vous accédez alors à ces tables résultats plutôt qu'aux tables de base ce qui améliore grandement les performances de votre système. Nous verrons également que DB2 est aussi capable de tenir ces tables à jour soit en direct soit en différé. En outre, l'optimizer DB2 peut, dans certaines circonstances, utiliser automatiquement ces tables à la place des tables de bases.

Les MQT sont disponibles depuis la version 5.2 pour DB2 sous Windows et sous Linux mais également pour Z/Os depuis la version 8 et sur ISeries depuis la version 5.3 au moins. Les explications et exemples ci-dessous se rapportent plus spécifiquement à la version Windows.

## II - La syntaxe

Comme toutes tables, une table de résumé est créée via la commande "CREATE TABLE" mais son contenu est défini au moyen d'une commande "SELECT ...".

```
CREATE TABLE maTable
AS (
    SELECT champs1, champs2, ..., champsN,
           SUM(champX) AS somme1,
           SUM(champY) AS somme2,
           COUNT(champZ) AS somme3
    FROM maTableDeDonnées
    [GROUP BY champs1, champs2, ..., champsN]
)
DATA INITIALLY DEFERRED
REFRESH DEFERRED/REFRESH IMMEDIATE
[ENABLE QUERY OPTIMIZATION/DISABLE QUERY OPTIMIZATION]
[MAINTAINED BY SYSTEM/USER/FEDERATED_TOOL]
```

Les clauses spécifiques sont abordées dans la suite de cet article.



*Vous pouvez faire référence à une autre table de résumé dans le "SELECT" de création.*

## III - Maintenir les données en permanence synchronisées

La clause "REFRESH IMMEDIATE" donne instruction à DB2 de maintenir à jour en permanence et automatiquement les tables de résumés. Evidemment cela nécessite pour chaque "INSERT", "UPDATE" ou "DELETE", que les tables de résumé soient également mises à jour, ce qui consomme quelques ressources.



*Si vous optez pour un rafraîchissement immédiat, la clause "SELECT" de création ne peut contenir ni "DISTINCT" ni "HAVING" ni "CUBE" mais surtout aucune fonction spécifique à un type de donnée comme par exemple "AVG". Par contre, s'il contient la clause "GROUP BY", il doit contenir un Count(\*) ou Count\_Big(\*). Si vous utilisez une fonction d'agrégation comme SUM sur une colonne nullable, vous devez également prévoir d'ajouter un "COUNT" sur cette colonne spécifique. Vous ne pouvez pas non plus inclure de sous-requête dans le select. Consultez la documentation DB2 pour obtenir les nombreuses autres restrictions.*

Les restrictions sur la clause "REFRESH IMMEDIATE" font que, bien souvent, elle ne peut être utilisée. Il faut alors passer au mode différé.

## IV - Différer la synchronisation entre tables de données et tables de résumés.

Dans bien des cas, les requêtes effectuées sur les tables ne portent pas sur les données les plus récentes. Dans ce contexte, il est inutile de maintenir à jour les tables de résumé. Cette mise à jour peut être différée par exemple la nuit. Dans ce cas, utilisez simplement la clause "REFRESH DEFERED".

Pour provoquer la mise à jour utilisez la commande:

```
REFRESH TABLE [maTable]
```

## V - La clause QUERY OPTIMIZATION

L'optimzer DB2 est capable d'utiliser automatiquement les MQT afin d'accélérer le traitement des SELECT sur les tables de bases. La clause "ENABLE QUERY OPTIMIZATION" active cette option. C'est la valeur par défaut. A l'inverse, si vous spécifié la clause "DISABLE QUERY OPTIMIZATION", DB2 n'effectuera plus cette optimisation.

## VI - La clause MAINTAINED BY

"MAINTAINED BY SYSTEM" est le défaut. Les autres options spécifient que c'est, soit l'utilisateur, soit un outil de réplication qui va mettre la table à jour. Dans ces deux dernier cas, l'instruction "REFRESH TABLE" ne peut être utilisée. Avec "REFRESH IMMEDIATE", seule "MAINTAINED BY SYSTEM" peut être utilisé.

## VII - Exemples

Les différents exemples repris ci-dessous sont réalisés avec la base de données "SAMPLE" fournie avec DB2. Il vous sera facile de les reproduire et d'en faire des variations.

### V-A - Exemple 1: Avec un simple "GROUP BY"

Le but de l'exemple ci-dessous est de réaliser une table de reporting donnant la masse salariale de chaque département. Dans ce première exemple, la mise à jour de la table est différée.

#### Commande SQL

```
CREATE TABLE MasseSalariale
AS (
  SELECT WORKDEPT,DECIMAL(SUM(SALARY),8,2) AS SALAIRE_MOYEN
  FROM EMPLOYEE
  GROUP BY WORKDEPT
)
DATA INITIALLY DEFERRED REFRESH DEFERRED
;
```

#### Contenu de la table

WORKDEPT	SALAIRE_MOYEN
A00	354250,00
B01	94250,00
C01	308890,00
D11	646620,00
D21	358680,00
E01	80175,00
E11	317140,00
E21	282520,00


## V-B - Exemple 2: Avec rafraîchissement immédiat

Il s'agit de réaliser la même table que dans l'exemple précédent mais avec rafraîchissement immédiat. Les données sont toujours maintenues à jour.

Pour utiliser la clause "REFRESH IMMEDIATE", nous devons ajouter "COUNT(\*)". Comme nous utilisons une fonction d'agrégation sur une colonne nullable, nous devons également introduire un "COUNT" sur cette colonne.

### Commande SQL

```
CREATE TABLE MasseSalariale
AS (
  SELECT COUNT(*) AS COMPTE, WORKDEPT, SUM(SALARY) AS SALAIRE_MOYEN
  , COUNT(SALARY) AS NBR_SALAIRES
  FROM EMPLOYEE
  GROUP BY WORKDEPT
)
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
;
```

 La fonction "DECIMAL" étant spécifique à un type, elle doit être retirée dans la commande.

### Contenu de la table résultat

COMPTE	WORKDEPT	SALAIRE_MOYEN	NBR_SALAIRES
5	A00	354250,00	5
1	B01	94250,00	1
4	C01	308890,00	4
11	D11	646620,00	11
7	D21	358680,00	7
1	E01	80175,00	1
7	E11	317140,00	7
6	E21	282520,00	6

## V-C - Exemple 3: Avec "GROUP BY CUBE"

Cette exemple a pour but de fournir en résultat une table simulant un tableau à deux entrées pour obtenir le salaire moyen par département et/ou par niveau. Le select faisant appel à la clause "GROUP BY CUBE", le rafraîchissement de la table doit être différé.

### Commande SQL

```
CREATE TABLE SalaireMoyen
AS (
  SELECT EDLEVEL, WORKDEPT, DECIMAL(AVG(SALARY), 8, 2) AS SALAIRE_MOYEN
  FROM EMPLOYEE
  GROUP BY CUBE(WORKDEPT, EDLEVEL)
)
DATA INITIALLY DEFERRED REFRESH DEFERRED
;
REFRESH TABLE SalaireMoyen;
```

### Contenu de la table résultat

EDLEVEL	WORKDEPT	SALAIRE_MOYEN
12	-	35713,33
14	-	47902,85
15	-	43280,00
16	-	60561,07
17	-	50125,71

#### Contenu de la table résultat


18 -	78574,28
19 -	66500,00
20 -	98250,00
- -	58155,35
- A00	70850,00
- B01	94250,00
- C01	77222,50
- D11	58783,63
- D21	51240,00
- E01	80175,00
- E11	45305,71
- E21	47086,66
14 A00	44250,00
18 A00	99625,00
19 A00	66500,00
18 B01	94250,00
16 C01	73800,00
18 C01	68420,00
20 C01	98250,00
16 D11	57513,33
17 D11	60620,00
18 D11	59840,00
14 D21	42180,00
15 D21	43280,00
16 D21	71710,00
17 D21	43260,00
16 E01	80175,00
12 E11	35713,33
14 E11	37750,00
16 E11	89750,00
17 E11	41250,00
14 E21	55630,00
16 E21	38543,33

#### V-D - Exemple 4: La table n'est pas le résultat d'un "GROUP BY"

Il est également possible de réaliser des tables de requête sur un "SELECT" ne faisant pas appel à la clause "GROUP BY".

#### Commande SQL

```
CREATE TABLE TablesJointes
AS (
  SELECT EMPNO, SALARY, DEPTNAME
  FROM EMPLOYEE, DEPARTMENT
  WHERE EMPLOYEE.WORKDEPT = DEPARTMENT.DEPTNO
)
DATA INITIALLY DEFERRED REFRESH DEFERRED
;
```

 *Le même résultat peut être obtenu avec une vue. Par contre, si votre "SELECT" est complexe, avec la vue, il doit chaque fois être recalculé alors qu'avec une MQT, lors de l'exploitation, la consultation se fait sur une simple table.*

#### V-E - Exemple 5: Avec une sous-requête.

Dans cet exemple, nous allons créer une table qui contient, par département, le nombre d'employés occupés à plus d'un projet. Cette requête contient à la fois une sous-requête et une jointure.

#### Commande SQL

```
CREATE TABLE SurOccupation
AS (
```

### Commande SQL

```
SELECT DEPTNAME, COUNT(*) AS NBR
FROM EMPLOYEE, DEPARTMENT
WHERE (SELECT COUNT(*) FROM EMPPROJECT WHERE EMPPROJECT.EMPNO = EMPLOYEE.EMPNO)>1
AND EMPLOYEE.WORKDEPT = DEPARTMENT.DEPTNO
GROUP BY DEPTNAME
)
DATA INITIALLY DEFERRED REFRESH DEFERRED
;
```

### Contenu de la table résultat

DEPTNAME	NBR
ADMINISTRATION SYSTEMS	5
INFORMATION CENTER	3
MANUFACTURING SYSTEMS	5
SOFTWARE SUPPORT	3
SPIFFY COMPUTER SERVICE DIV.	1
SUPPORT SERVICES	1