Traiter des données soumises à date d'effet, avec et sans SQL.

- Les articles -

Date de mise en ligne : vendredi 2 octobre 2009

Date de parution : 26 mai 2006

Description :

Comment récupérer une donnée soumise à une date d'effet, comme par exemple un prix de vente dans un fichier de tarif.

N.B. : Cet article a fait l'objet d'une réactualisation, avec notamment l'ajout d'une seconde technique SQL. On retrouvera les techniques SQL présentées ici dans un autre article - à venir - consacré aux sous-requêtes scalaires de type "full select".

Environnement iSeries

Au niveau des requêtes SQL, il existe de petites différences entre DB2 et MySQL qui sont indiquées au cas par cas.

Comment récupérer une donnée soumise à une date d'effet, comme par exemple un prix de vente dans un fichier de tarif.

N.B. : Cet article a fait l'objet d'une réactualisation, avec notamment l'ajout d'une seconde technique SQL. On retrouvera les techniques SQL présentées ici dans un autre article - à venir - consacré aux sous-requêtes scalaires de type "full select".

Au niveau des requêtes SQL, il existe de petites différences entre DB2 et MySQL qui sont indiquées au cas par cas.

Nous allons démarrer cet article par la constitution d'un jeu d'essai, que voici :

- création d'une table des prix de vente :

```
CREATE TABLE PRXVTE (
 CODSOC CHAR ( 3) NOT NULL WITH DEFAULT,
 CODART DEC (8 , 0) NOT NULL WITH DEFAULT,
 DATEFF DEC (8 , 0) NOT NULL WITH DEFAULT,
 PXVENT DEC (11 , 2) NOT NULL WITH DEFAULT)
```

- table complétée d'un index ayant la clé suivante :

```
CREATE INDEX PRXVTEL1 ON PRXVTE
 (CODSOC ASC, CODART ASC, DATEFF ASC)
```

- constitution d'un jeu de données :

```
INSERT INTO PRXVTE VALUES('001', 1, 20090101, 1,5)
 INSERT INTO PRXVTE VALUES('001', 1, 20090301, 2,5)
 INSERT INTO PRXVTE VALUES('001', 1, 20090601, 3)
 INSERT INTO PRXVTE VALUES('001', 2, 20090101, 2,5)
 INSERT INTO PRXVTE VALUES('001', 3, 20090301, 3,5)
 INSERT INTO PRXVTE VALUES('001', 3, 20090601, 4)
```

Approche purement algorithmique

N.B. : J'ai choisi d'écrire les algorithmes ci-dessous dans un pseudo-code très proche du langage ADELIA, qui me semble bien adapté à cet exercice. Pour les développeurs RPG, les techniques seront strictement identiques , il suffira de remplacer :

- l'ordre POSITIONNER_AP par un SETGT,
- l'ordre LIRE_PRECEDENT par un READPE,
- l'ordre POSITIONNER_AV par un SETLL,
- l'ordre LIRE_SUIVANT par un un READE.

Pour récupérer le prix de vente le plus récent d'un article par rapport à une date d'effet, on peut écrire le code suivant :

```
POSITIONNER_AP PRXVTEL1 (clé : Code_soc, code_art, Date_effet)
 LIRE_PRECEDENT PRXVTEL1 (clé : Code_soc, code_art)
 SI             PRXVTEL1 Existe
 Prix_vte = PXVENT
 SINON
 Prix_vte = 0
 FIN
```

A noter que si l'index PRXVTEL1 avait une clé descendante (au lieu d'ascendante) sur DATEFF, on écrirait la recherche ainsi :

```
POSITIONNER_AV PRXVTEL1 (clé : Code_soc, code_art, Date_effet)
 LIRE_SUIVANT   PRXVTEL1 (clé : Code_soc, code_art)
 SI             PRXVTEL1 Existe
 Prix_vte = PXVENT
 SINON
 Prix_vte = 0
 FIN
```

Maintenant, comment écrire la même chose en SQL ?

Voici une première technique :

```
SELECT CODSOC, CODART, DATEFF, PXVENT
 FROM PRXVTE
 WHERE CODSOC = '001'
 AND CODART = 1
 AND DATEFF <= 20090320
 ORDER BY CODSOC, CODART, DATEFF DESC
 FETCH FIRST 1 ROWS ONLY
```

- Tri sur code société + code article + date décroissante.
- Sélection sur date <= à date d'effet.
- Fetch first 1 rows only pour ne récupérer que le 1er enregistrement correspondant. C'est-à-dire directement celui le plus proche de la date d'effet.
- Si SQLCOD à 100, alors pas d'enregistrements trouvés. Donc pas de tarif.

Selon le jeu d'essai proposé en début d'article, la requête ci-dessus retournerait le résultat suivant :

```
CODSOC  CODART     DATEFF        PXVENT
 001         1   20090301          2,50
```

N.B. : pour que la requête ci-dessus fonctionne avec MySQL 5, il suffit de remplacer la clause "FETCH FIRST..." par la clause "LIMIT 1", ce qui nous donne à l'arrivée :

```
SELECT CODSOC, CODART, DATEFF, PXVENT
 FROM PRXVTE
 WHERE CODSOC = '001'
 AND CODART = 1
```

```
AND DATEFF <= '2009-03-20'
ORDER BY CODSOC, CODART, DATEFF DESC
LIMIT 1
```

Voici une seconde technique permettant d'obtenir le même résultat :

```
SELECT CODSOC, CODART, DATEFF, PXVENT
FROM PRXVTE
WHERE CODSOC = '001'
AND CODART = 1
AND DATEFF = (
SELECT MAX(DATEFF) FROM PRXVTE
WHERE CODSOC = '001'
AND CODART = 1
AND DATEFF <= 20090320 )
```

Cette seconde technique s'appuie sur le principe des sous-requêtes scalaires de type "full select". Nous approfondirons cette technique dans un autre article spécialement dédié à cette problématique.

La requête ci-dessus fonctionne correctement. C'est celle que je vous proposais dans la précédente version de cet article, mais à la réflexion, je vous recommande d'utiliser plutôt la variante ci-dessous :

```
SELECT A.CODSOC, A.CODART, A.DATEFF, A.PXVENT
 FROM PRXVTE A
 WHERE A.CODSOC = '001'
 AND A.CODART = 1
 AND A.DATEFF = (
 SELECT MAX(B.DATEFF) FROM PRXVTE B
 WHERE B.CODSOC = A.CODSOC
 AND B.CODART = A.CODART
 AND B.DATEFF <= 20090320 )
```

Je préfère cette seconde solution car elle évite d'avoir à modifier les critères de sélection (code société et code article) en plusieurs endroits (dans la requête principale et dans la sous-requête scalaire).

N.B. : la technique ci-dessus fonctionne aussi bien avec DB2 qu'avec MySQL 5.

Pour aller plus loin

Vous pouvez également envisager de coder l'algorithme de recherche du prix à l'intérieur d'une UDF (User Defined Function).

A l'intérieur de cette UDF, vous pourrez coder votre algorithme de recherche de prix soit intégralement en SQL, en utilisant l'une ou l'autre des techniques vues dans cet article, soit en faisant appel à un programme externe, de type RPG ou Cobol, qui lui pourra faire appel à l'un des algorithmes écrits en pseudo-code proposés en début d'article.

Par exemple, si l'on crée une UDF appelée PRIX_VENTE, recevant en entrée les codes sociétés, code article, ainsi qu'une date, et renvoyant en sortie le prix de vente, on pourrait afficher tous les prix du catalogue d'articles au moyen

d'une requête telle que celle ci-dessous :

```
SELECT CODSOC, CODART, PRIX_VENTE(CODSOC, CODART, 20090320)
FROM ARTICLE
```

Si vous souhaitez explorer cette possibilité, il existe un très bon redbook IBM traitant de ce sujet.