Déterminer les périodes de vacance ou d'inactivité avec SQL DB2

- Les articles -

Date de mise en ligne : lundi 24 août 2015

Description :

Suite à une mauvaise manipulation, cet article avait malencontreusement disparu du site XDocs400.com. Le revoici donc, avec toutes nos excuses pour la gêne occasionnée.

Environnement iSeries

Détecter les périodes de vacance ou d'inactivité au sein de données d'entreprise est un besoin vital pour certaines organisations. Par exemple, une société qui loue des biens immobiliers aura tout intérêt à ce que ces biens ne demeurent pas vacants trop longtemps. Et quand cela arrive, elle aura besoin de déterminer combien d'argent elle a perdu par rapport aux périodes de vacances des biens qu'elle gère. Dans un autre registre, une société de service qui place du personnel en régie aura besoin de déterminer les périodes d'inactivité (on parle généralement de périodes d'inter-contrat) de ses employés.

Si on prend l'exemple de logement en location, on aura dans notre base de données une table des occupations de logement, avec pour chaque logement, une série de lignes définissant les dates d'entrée et de sortie des occupants respectifs. Si on souhaite dans ce contexte déterminer les périodes de vacances, il nous faut donc trier notre jeu de données par logement et dates d'entrée, et identifier les périodes de vacances en analysant la date de sortie de l'occupant N par rapport à la date d'entrée de l'occupant N+1.

Ce que je viens de décrire est relativement facile à faire dans un programme RPG, un script PHP, voire une procédure stockée DB2. Mais on peut aussi le réaliser via une seule requête SQL, comme nous allons le voir dans cet article.

Pour les besoins de ma démonstration, je vais prendre pour exemple une table des activités de collaborateurs, contenant les périodes d'activité de chaque collaborateur.

Dans un cas de ce type, la difficulté principale consiste à faire "matcher", pour un collaborateur donné, la date de fin d'activité d'une période d'activité, qui se trouve sur une ligne de la table, avec la date de début d'activité d'une autre période d'activité, qui se trouve sur une autre ligne de la même table. Ce sont donc les "trous" entre date de fin d'activité précédente et date de début d'activité suivante que l'on cherche à détecter. Quelquefois, il n'y a pas de seconde ligne définissant une nouvelle activité, il y a dans ce cas un trou plus important délimité par la date de fin d'activité précédente et la date du jour, ou encore entre la date du jour et la date de début d'activité.

L'étude de cas présentée ici est "tirée" d'un cas réel dans lequel les dates n'étaient pas de véritables colonnes de type date, mais des colonnes de type numérique (8.0). Une fonction de conversion de date était donc nécessaire, ce point est couvert par la fonction CVT_ALP_2_DATE présentée ci-dessous, qui fonctionne aussi bien sur DB2 for i, que sur DB2 Express C :

```
CREATE FUNCTION MABASETEST.CVT_ALP_2_DATE (
 DATE_ENT CHAR(10) )
 RETURNS DATE
 LANGUAGE SQL
 DETERMINISTIC
 CONTAINS SQL
 CALLED ON NULL INPUT
 RETURN
 CASE WHEN DATE_ENT IS NULL OR TRIM ( DATE_ENT ) = ''
 THEN NULL
 ELSE DATE ( TO_DATE ( DATE_ENT , 'YYYY-MM-DD' ) )
 END  ;
```

Pour les besoins de la démonstration, nous avons besoin d'une table DB2 que j'ai appelée TSTACTIVITE.

```
-- Attention : pour DB2 Express C, penser à supprimer la notion de CCSID
 CREATE TABLE MABASETEST.TSTACTIVITE (
```

```
 NO_ID CHAR(30) CCSID 297 DEFAULT NULL,
 SYS_ORIG CHAR(6) CCSID 297 DEFAULT NULL,
 LIB_PRENOM_USUEL CHAR(50) CCSID 297 DEFAULT NULL,
 LIB_NOM_USUEL CHAR(50) CCSID 297 DEFAULT NULL,
 EMPLOI CHAR(120) CCSID 297 DEFAULT NULL,
 DAT_DEB_EMPLOI CHAR(10) CCSID 297 DEFAULT NULL,
 DAT_FIN_EMPLOI CHAR(10) CCSID 297 DEFAULT NULL,
 COD_OPER CHAR(8) CCSID 297 DEFAULT NULL,
 TMP_CRE TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
 TMP_MAJ TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
 ) ;
```

La table étant créée, on pourra y injecter un jeu de données ayant la forme suivante :

```
DELETE FROM MABASETEST.TSTACTIVITE ;
 INSERT INTO MABASETEST.TSTACTIVITE
 ( NO_ID, SYS_ORIG, LIB_PRENOM_USUEL, LIB_NOM_USUEL, EMPLOI, DAT_DEB_EMPLOI, DAT_FIN_EMPLOI, COD_OPER,
TMP_CRE, TMP_MAJ )
 VALUES
 ( '11492', '', 'ISABELLE', 'ROBERT', 'VENDEUR(EUSE) COLL', '2008-07-01', '2008-07-31', '', '2012-08-17
13:21:44.480870', '2012-08-17 13:21:44.480870' ),
 ( '11492', '', 'ISABELLE', 'ROBERT', 'VENDEUR(EUSE) COLL', '2008-02-01', '2008-02-29', '', '2012-08-17
13:21:44.481767', '2012-08-17 13:21:44.481767' ),
 ( '11492', '', 'ISABELLE', 'ROBERT', 'VENDEUR(EUSE) COLL', '2007-07-01', '2007-07-31', '', '2012-08-17
13:21:44.482747', '2012-08-17 13:21:44.482747' ),
```

Je ne vous donne ici qu'un échantillon. Vous devrez donc le compléter, en y ajoutant plusieurs employés, et en prévoyant des périodes d'inactivité, ou pas, pour certains de ces employés, l'objectif étant de se rapprocher le plus possible de cas de la vie courante.

La requête d'identification des périodes d'inactivité ci-dessous a été testée avec succès sur DB2 for i et sur DB2 Express C (version 9.7).

Pour bien comprendre son fonctionnement, il est recommandé d'exécuter individuellement chacune des CTE que j'ai décrit brièvement ci-dessous :

- 1ère CTE : permet de sélectionner les lignes qui nous intéressent, on en profite pour convertir les dates alpha en vraies dates DB2
- 2ème CTE : permet la mise en place d'une rupture sur no_id et dat_deb_emploi
- 3ème CTE : pour la création d'un lien artificiel vers une "rupture suivante" (si la rupture suivante n'existe pas, ce n'est pas grave, car "rupture" et "rupture suivante" seront liées par un INNER JOIN)
- 4ème CTE pour la sélection finale des périodes de vacance (on ne prend que les cas où x.vacance > 0 car les autres cas correspondent à des chevauchements de dates

Voici le code de la requête SQL :

```
-- 1ère CTE

 with temp1 as (
 select no_id, MABASETEST.CVT_ALP_2_DATE(dat_deb_emploi) as  dat_deb_emploi,
 MABASETEST.CVT_ALP_2_DATE(dat_fin_emploi) as  dat_fin_emploi
 from MABASETEST.TSTACTIVITE
 where no_id is not null and no_id <> '' and emploi is not null
```

```
 and dat_deb_emploi is not null and dat_fin_emploi is not null
 group by no_id, dat_deb_emploi, dat_fin_emploi
 ) ,

 -- 2ème CTE

 temp2 as (
 select no_id, dat_deb_emploi, dat_fin_emploi,
 row_number() over(partition by no_id order by no_id,
 dat_deb_emploi)  as rupture
 from (select * from temp1
 where dat_deb_emploi is not null and dat_fin_emploi is not null
 order by no_id, dat_deb_emploi) a
 order by no_id, dat_deb_emploi
 ) ,

 -- 3ème CTE

 temp3 as (
 select a.*, a.rupture+1 as rupture_suivante from temp2 a
 ) ,

 -- 4ème CTE

 temp4 as (
 select * from (
 select a.no_id, a.dat_fin_emploi + 1 day as dat_deb_inactif,
 b.dat_deb_emploi - 1 day as dat_fin_inactif,
 days(b.dat_deb_emploi) - days(a.dat_fin_emploi) - 1 as vacance
 from (select * from temp3 order by no_id, dat_deb_emploi) a
 inner join (select * from temp3 order by no_id, dat_deb_emploi) b
 on a.no_id = b.no_id and a.rupture_suivante = b.rupture
 ) x
 where x.vacance > 0
 )

 -- dernière requête produisant le listing des périodes d'inactivité

 select no_id, 'INACTIF' as code,
 (select x.lib_prenom_usuel from MABASETEST.TSTACTIVITE x where a.no_id = x.no_id fetch first 1 row only) as
prenom ,
 (select y.lib_nom_usuel from MABASETEST.TSTACTIVITE y where a.no_id = y.no_id fetch first 1 row only) as
nom ,
 'INACTIVITE' as situation,
 char( dat_deb_inactif, iso ) as dat_deb_inactif,
 char( dat_fin_inactif, iso ) as dat_fin_inactif,
 'MYUSERCODE' as cod_oper, current timestamp as tmp_cre, current timestamp as tmp_maj
 from temp4 a
 ;
```

Le code SQL présenté par SPIP n'est pas particulièrement bien indenté, mais en le reprenant par copier-coller dans un éditeur de type Notepad++, vous devriez obtenir un code plus lisible.

J'espère que vous prendrez du plaisir dans l'analyse de cette étude de cas, et surtout que les techniques utilisées ici vous donneront des idées pour vos propres projets.

Post-scriptum :

Pour des digressions autour du développement en général, et du développement web en particulier, je vous invite à me retrouver sur mon blog http://gregphplab.com