Requête SQL dynamique

Introduction

L'utilisation d'une requête SQL dynamique plutôt que des instructions SQL uniquement Adélia s'avère utile, voire nécessaire, dans 2 cas :

1. Eviter la création de plusieurs curseurs ou chargements Adélia afin de sélectionner des enregistrements base de données conformément à plusieurs critères de sélection aléatoires,
2. Permettre d'utiliser des paramètres SQL n'ayant pas d'équivalence dans les instructions SQL Adélia afin de simplifier les traitements (GROUP BY, IN, NOT IN, etc.)

Dans le cadre d'une requête SQL dynamique, **la requête SQL est codée directement en langage SQL dans une variable de travail alphanumérique**.
La requête est construite dynamiquement en fonction des critères de sélection aléatoires renseignés dans le programme et/ou conformément aux paramètres SQL souhaités (IN, NOT IN, etc.).

Instructions concernées et syntaxe

CURSEUR NomCurseur :VarRequeteSQL.
CHARGEMENT NomChargement NomListeGraph:liste *SQL_D VarRequeteSQL SuiteVariablesHotes Pas.
EXEC_SQL (:VarRequeteSQL).

Construction de requête : description et exemples

Construction d'une requête

Exemple avec critère de sélection aléatoire.

Calcul de l'ancienneté moyenne de tous les employés, ou seulement de celle des hommes ou des femmes.
L'entité PERSONNEL est à l'origine de la table PERSONP.

Sans requête, il faut définir 2 curseurs

```
[Déclaration]
* Curseur sans sélection pour traiter tous les employés :
CURSEUR CURS_ANCIEN_TOUS PERSONNEL                              -
                         *COL(PE_DAT_ENTREE)

* Curseur avec sélection sur le code sexe :
CURSEUR CURS_ANCIEN_F_H  PERSONNEL                              -
                         *COL(PE_DAT_ENTREE)                    -
                         *COND(PE_COD_SEXE_PER = :ZPE_CODE_SEXE)
```

Avec requête dynamique

```
[Déclaration]
* Variable contenant la requête SQL
ALPHA(300) W_REQUETE_SQL

* Curseur en référence à la variable requête (préfixée par « : »)
CURSEUR CURS_ANCIEN :W_REQUETE_SQL

[(Traitement)]
* Construction de la requête SQL dans la variable W_REQUETE_SQL
W_REQUETE_SQL = *BLANK
W_REQUETE_SQL = 'SELECT PEDATPENT FROM schema.PERSONP'
SI ZPE_CODE_SEXE = 'F';'H'
 W_REQUETE_SQL = W_REQUETE_SQL /// -
    ' WHERE PEPSEX = ''' /// ZPE_CODE_SEXE /// ''''
FIN
```

La variable requête **W_REQUETE_SQL** est une variable alpha de longueur libre, suffisante, bien sûr, pour contenir toute l'instruction SQL. La requête SQL qu'elle contient est donc une chaîne de caractères alphanumérique qui doit être délimitée par des cotes.
Soit dans notre exemple :

```
* Sélection de tous les employés
 W_REQUETE_SQL = 'SELECT PEDATPENT FROM schema.PERSONP’
```

> **Attention**
>
> Dans la requête en langage SQL, il faut indiquer le nom réel des colonnes et des tables au lieu des noms Adélia (sauf pour les variables hôtes). En outre, la syntaxe des requêtes doit respecter la syntaxe attendue par le SGBD cible. Ainsi, pour une base de données DB2 400 (native ou SQL) le schéma n’est pas spécifié. Le fichier ou la table étant recherché dans la liste de bibliothèques ou collections en ligne.

Utilisation de variables programme

Par concaténation, il est possible d’inclure dans la requête des valeurs provenant de variables hôtes (noms Adélia de variables programme), numériques et alphanumériques, afin de bâtir dynamiquement la requête.

Variable hôte alphanumérique

Reprenons l’exemple précédent avec ajout de la sélection sur le code sexe :

```
SI ZPE_CODE_SEXE = 'F';’H’
   W_REQUETE_SQL = W_REQUETE_SQL ///                                -
                   ' WHERE PEPSEX = ''' /// ZPE_CODE_SEXE /// ''''
FIN
```

Dans ce cas, la variable requête **W_REQUETE_SQL** est complétée, par concaténation, avec sélection sur le code sexe.

La première partie de ce complément d’instruction SQL est une chaîne de caractères.
Celle-ci doit donc être délimitée par deux cotes. Soit :

```
W_REQUETE_SQL = W_REQUETE_SQL /// ' WHERE PEPSEX = '
```

Mais la variable code sexe **ZPE_CODE_SEXE** est elle aussi une variable alphanumérique dont la valeur, **F** ou **H**, doit donc être délimitée dans la requête par des cotes.

Pour inclure une cote dans une chaîne de caractères alphanumériques il faut utiliser deux côtes : «''».
D’où les trois cotes terminant la première chaîne de caractères du complément d’instruction de la requête. Deux pour créer la première cote délimitant la valeur alphanumérique du code sexe, et une pour délimiter la fin de la chaîne de caractères du complément d’instruction :

```
W_REQUETE_SQL = W_REQUETE_SQL /// ' WHERE PEPSEX = '''
```

Le contenu de « W_REQUETE_SQL » est donc maintenant :

« SELECT **PEDATPENT** FROM schema.**PERSONP** WHERE PEPSEX = ' »

Ensuite, la chaîne de caractères est concaténée avec la valeur du code sexe contenue dans la variable hôte **ZPE_CODE_SEXE**, soit **F** ou **H** :

```
W_REQUETE_SQL = W_REQUETE_SQL /// ' WHERE PEPSEX = ''' /// ZPE_CODE_SEXE
```

Le contenu de « W_REQUETE_SQL » est donc maintenant :

« SELECT **PEDATPENT** FROM schema.**PERSONP** WHERE PEPSEX = '**F** »

Il faut maintenant coder dans la requête la cote fermant la valeur de la variable hôte **‘F’**. Pour cela, comme nous l’avons vu, il faut utiliser deux côtes : «**''**».
Et pour que cette cote puisse être concaténée avec le contenu de la variable requête il faut que celle-ci soit elle-même délimitée par deux cotes. D’où les quatre cotes de fin :

```
' WHERE PEPSEX = ''' /// ZPE_CODE_SEXE /// ''''
```

Le contenu de « W_REQUETE_SQL » est donc maintenant :

« SELECT **PEDATPENT** FROM schema.**PERSONP** WHERE PEPSEX = '**F**' »

Variable hôte alphanumérique

Dans le cas d'une valeur numérique, il n'est pas nécessaire de la délimiter par des cotes. La variable peut donc être directement concaténée à la variable requête.
Exemple :

```
* Soit la sélection de la date d'entrée de chaque employé ayant un code * société égal à celui saisi par l'
utilisateur.
 W_REQUETE_SQL = 'SELECT PEDATPENT FROM PERSONP WHERE PECODSTE = ' /// ZPE_COD_SOCIETE
```

Quelques exemples

Exemple de curseur

Critère de sélection aléatoire et utilisation de paramètres SQL non disponibles sur l'instruction de déclaration de curseur Adélia : fonction « COUNT » et paramètre « GROUP BY ».
Dans une procédure, alimentation d'une liste mémoire avec le résultat de la recherche du nombre d'employés par société. L'analyse doit prendre en compte soit tous les employés, soit faire une sélection sur le code sexe :

Procédure NB_EMPLOYE_SOCIETE

```
[DECLARATION]
* Variable requête
ALPHA(300) W_REQUETE_SQL

* Colonnes de la liste mémoire contenant le résultat de la requête
ALPHA(30) W_NOM_SOCIETE
NUM_BIN_4 W_NB_EMPLOYE

* Paramètres de la procédure
ALPHA(1) WP_CODE_SEXE
LISTE LSM_NB_EMP_SOC W_NOM_SOCIETE W_NB_EMPLOYE

* Curseur dynamique
CURSEUR CUR_NB_EMP_SOC :W_REQUETE

* Déclaration des paramètres gérés en entrée/sortie par la procédure
PARAM WP_CODE_SEXE LSM_NB_EMP_SOC

[(Traitement)]
* Construction de la requête SQL dans la variable W_REQUETE
W_REQUETE_SQL = *BLANK
W_REQUETE_SQL = 'SELECT SONOMSOC, COUNT(PECODMAT) FROM PERSONP, SOCIETE -
                 WHERE PECODSTE=SOCODSTE'
*
SI WP_CODE_SEXE = 'F';'H'
   W_REQUETE_SQL = W_REQUETE_SQL /// ' AND' ///               -
                    ' PECODSEX = ''' /// WP_CODE_SEXE /// ''''
FIN
*
W_REQUETE_SQL = W_REQUETE_SQL /// ' GROUP BY SONOMSOC'
*
OUVRIR_SQL_C CUR_NB_EMP_SOC
* Alimentation des colonnes de la liste par SONOMSOC et COUNT(PECODMAT)
LIRE_AV_SQL_C CUR_NB_EMP_SOC :W_NOM_SOCIETE, :W_NB_EMPLOYE
TANT_QUE *SQLCODE = *NORMAL
   INSERER_ELT LSM_NB_EMP_SOC
    LIRE_AV_SQL_C CUR_NB_EMP_SOC :W_NOM_SOCIETE, :W_NB_EMPLOYE
REFAIRE
FERMER_SQL_C CUR_NB_EMP_SOC
```

Exemple de chargement

(Voir la syntaxe de l'instruction « CHARGEMENT » en début de document).

Procédure NB_EMPLOYE_SOCIETE

```
[Déclaration]
* Variable requête
ALPHA(1000) W_REQUETE_SQL
ALPHA(7)    W1_COND
*
CHARGEMENT CHRG_ENT FEN_LISTE.LST_DE_ENT:LISTE *SQL_D W_REQUETE_SQL     -
           ZZ_NUM_PROMESSE ZZ_NUM_DER_ENT ZZ_NUM_DER_ORI ZZ_COD_TYP_DER –
           ZZ_COD_FOU_APP ZZ_NOM_FOU_APP _VALEUR_PAS

[Traitement]
* Constitution de la requête
* Dans un premier temps la requête est renseignée avec les éléments
* invariants de la sélection.
W_REQUETE     = *BLANK
W_REQUETE_SQL = 'SELECT DENOPROM, DENODERENT, DENODERORI, DECDTPDER,    -
                 FOCDFOAPP, FONMFOAPP FROM ENTDERP, NATUREP, FOUAPP     –
                 WHERE DECDNTPRO=NACDNTPRO AND FONOFOUAPP = DENOFOUAPP'
*
W1_COND = ' AND'
*
* La requête est ensuite complétée en fonction de la saisie des champs
* constituant le filtre de sélection des enregistrements en base de données.
*
* N° promesse. Variable hôte numérique. Il suffit donc de la concaténer à la requête
SI CHS_NUM_PROMESSE <> 0
 W_REQUETE = W_REQUETE /// W1_COND /// ' DENOPROM = ' // CHS_NUM_PROMESSE
FIN
* N° dérogation. Variable hôte alphanumérique et utilisation de la fonction SQL « LIKE ‘xxx%’». Permet de
sélectionner les enregistrements commençant par la chaîne de caractères « CHS_NUM_DER_ENT ». Il faut reprendre
ici le même principe vu précédemment pour coder une cote dans la variable requête. Il faut pour cela utiliser
deux côtes : «''»
SI CHS_NUM_DER_ENT <> *BLANK
 W_REQUETE = W_REQUETE /// W1_COND /// ' DENODERENT LIKE ''' // CHS_NUM_DER_ENT /// '% '''
FIN
* Sélection du n° de DA.
SI P_NUM_DA <> 0
 * Sélection pour la DA. Variable hôte numérique et utilisation de la
 * fonction SQL « IN » qui permet de sélectionner les enregistrements ayant
 * une valeur égale à celles extraite dans la seconde sélection.
 * A noter l’utilisation d’une fonction prédéfinie directement dans la
 * requête : &NUM_ALPHA
 W_REQUETE = W_REQUETE /// W1_COND /// ' DENOPROM IN –
 (SELECT LVNOPROM FROM LIGVTEDP WHERE LVNODAV = ' // -
 &NUM_ALPHA(CHO_NUM_DA) /// ')'
FIN
 * Date commercialisation. Variable hôte de type date. Utilisation de la
 * fonction SQL « DATE » et de la fonction prédéfinie Adélia &DATE_ALPHA afin
 * de gérer le format et les séparateurs de date.
SI ZW_DAT_COMM = _DAT_COMM_ACTU
 W_REQUETE = W_REQUETE /// W1_COND // ' PRDTCOMPRD <= DATE( ''' //&DATE_ALPHA ('*ISO';W_DATE)// ''' )'
FIN
W_REQUETE = W_REQUETE /// ' ORDER BY DENODERENT'
...
```

Le chargement de la liste s’effectue ensuite par une boucle de chargement CHARGT_LST/FIN_ CHARGT_LST.

Adélia iSeries : Gestion de sous-fichier

Instructions concernées et syntaxe

SQL_SFL_D VarRequeteSQL

```
* Remplissage SFL par requête Dynamique SQL_SFL_D
W_REQUETE = ‘SELECT PPMAT, PPNPE FROM PERSONP’
SQL_SFL_D W_LIB_REQ 1 ZPCOD_MATRICULE ZPNOM_PERSONNE

** La vue *1 est alors indiquée en remplissage par requête SQL et sans fichier guide
EFFACER 1
GESTION_SFL 1
   MAJ_SFL
FIN_GESTION_SFL
```