Les curseurs

Problématique

- Nous avons vu qu'il était possible d'exploiter le résultat d'un SELECT dans un bloc d'instructions, en utilisant la commande **SELECT colonne(s) INTO variable(s),** qui assigne les valeurs sélectionnées à des variables.
- Cependant, SELECT … INTO ne peut être utilisé que pour des requêtes qui ne ramènent **qu'une seule ligne de résultats.**
- Si la requête SELECT ramène plusieurs lignes, comment affecter toutes les valeurs à ces variables? C'est le rôle des curseurs.

C'est quoi un curseur?

- Les curseurs permettent de **parcourir un jeu de résultats d'une requête SELECT**, quel que soit le nombre de lignes récupérées, et d'en exploiter les valeurs.

- Un curseur est une zone mémoire qui est générée côté serveur (mise en cache) et qui permet de traiter **individuellement chaque ligne (ligne par ligne)** renvoyée par un SELECT.

Code client	Societe	Contact	Fonction
ALFKI	Alfred's Futterkiste	Maria Anders	Représentant(e)
ANATL	L'anatra laccata	Anna Grandi	Propriétaire
ANTOB	Antonio Berbi Salumi	Antonio Berbi	Propriétaire
AROUT	Around the Horn	Thomas Hardy	Représentant(e)
BERGS	Berglunds snabbköp	Christina Berglund	Propriétaire
B_AUS	Blauer See Delkatessen	Hanna Moos	Propriétaire
B_ONP	Blondel père e: fils	Frédérique Citeaux	Directeur du marketng
BOLID	Bolinderska boden	Kalle Berqlund	Propriétaire

Code client	Societe	Contact	Fonction
ALFKI	Alfred's Futterkiste	Maria Anders	Représentant(e)
ANATL	L'anatra laccata	Anna Grandi	Propriétaire
ANTOB	Antonio Berbi Salumi	Antonio Berbi	Propriétaire
AROUT	Around the Horn	Thomas Hardy	Représentant(e)
BERGS	Berglunds snabbköp	Christina Berglund	Propriétaire
BLAUS	Blauer See Delikatessen	Hanna Moos	Propriétaire
BLONP	Blondel père et fils	Frédérique Citeaux	Directeur du marketing
BOLID	Bolinderska boden	Kalle Berglund	Propriétaire

Comment utiliser les curseurs

- Quatre étapes sont nécessaires pour utiliser un curseur :

1. **Déclaration** du curseur après la déclaration des variables.
2. Utilisation de l'instruction **OPEN** pour initialiser le jeu de résultats pour le curseur.
3. Utilisation de l'instruction **FETCH** pour récupérer la ligne suivante pointée par le curseur et déplacer le curseur vers la ligne suivante dans le jeu de résultats.
4. **Fermeture du curseur.**

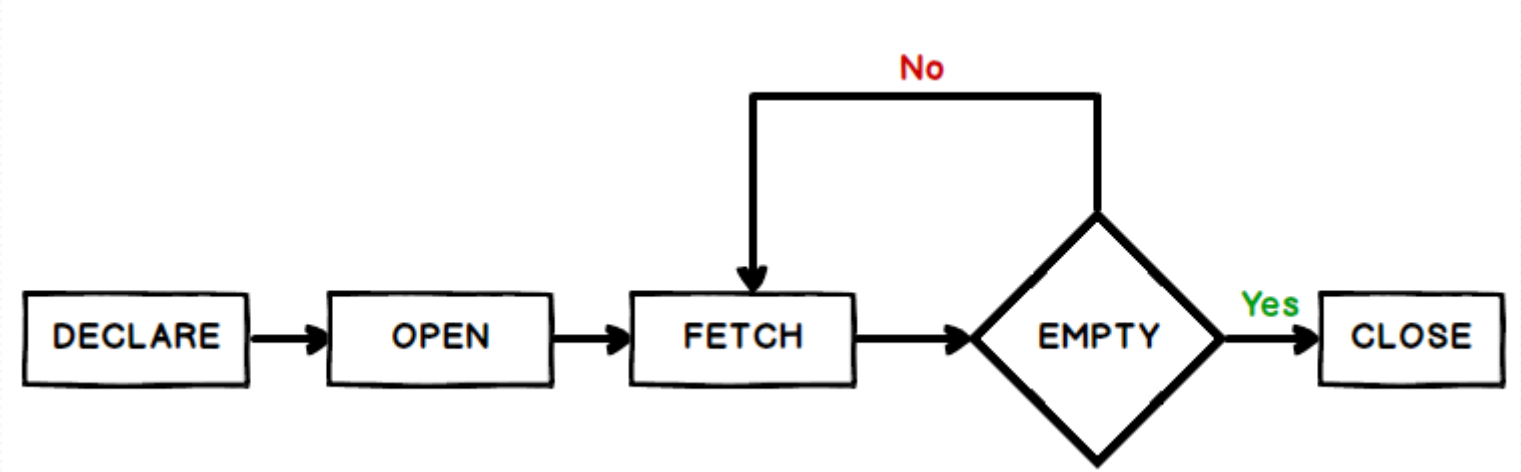
Syntaxe

La syntaxe générale pour l'utilisation d'un curseur est comme suit:

```
DECLARE nom_curseur   CURSOR  FOR instruction_SELECT
OPEN nom_curseur;
FETCH nom_curseur INTO liste_variables;
CLOSE nom_curseur;
```

Fonctionnement des curseurs

- Ce schéma montre la succession des étapes dans un curseur:



Déclaration du curseur

- Comme toutes les instructions **DECLARE**, la déclaration d'un curseur doit se faire au début du bloc d'instructions pour lequel celui-ci est défini.
- Plus précisément, on déclare les curseurs après les **variables locales** et les conditions, mais **avant les gestionnaires d'erreur.**

```
DECLARE nom_curseur CURSOR FOR requete_select ;
```

- Un curseur est donc composé d'un **nom**, et d'une requête SELECT.

```
DECLARE curseur_client CURSOR FOR SELECT * FROM Client;
```

Ouverture et fermeture du curseur

- En déclarant le curseur, on a donc associé un nom et une requête **SELECT**.
- L'ouverture du curseur va provoquer l'exécution de la requête SELECT, ce qui va produire un **jeu de résultats.**
- Une fois qu'on aura parcouru les résultats, il n'y aura plus qu'à fermer le curseur.
- Si on ne le fait pas explicitement, le curseur sera fermé à la fin du bloc d'instructions.

```
OPEN nom_curseur ;          -- Parcours du curseur et instructions diverses
CLOSE nom_curseur ;
```

Parcours du curseur

- Une fois que le curseur a été ouvert et le jeu de résultats récupéré, le curseur place un **pointeur** sur la **première ligne de résultats**.

- Avec la commande **FETCH**, on récupère la ligne sur laquelle pointe le curseur, et on fait avancer le pointeur vers la ligne de résultats suivante.

```
FETCH nom_curseur INTO variable(s);
```

Exemple

- Récupérer le nom du premier et deuxième stagiaire à l'aide de curseur :

```
Delimiter |

Create procedure exemple ()
Begin
Declare nomstg varchar(20);
Declare  curseur1  CURSOR FOR  select nom from stagiaires ;
Open  curseur1 ;
FETCH curseur1  into nomstg ;
SELECT nomstg ;
FETCH curseur1  into nomstg ;
SELECT nomstg ;
Close curseur1 ;
End |

Delimiter ;
```

Parcourir correctement les lignes du résultat

- Dans la procédure exemple(), on voulait récupérer les deux premières lignes, on a donc utilisé deux FETCH. Cependant, la plupart du temps, on ne veut pas seulement utiliser les deux premières lignes, mais toutes ! Or, sauf exception, on ne sait pas combien de lignes seront sélectionnées.
- On veut donc parcourir une à une les lignes de résultats, et leur appliquer un traitement, sans savoir à l'avance combien de fois ce traitement devra être répété. Pour cela, on utilise une boucle ! WHILE, REPEAT ou LOOP.
- Il n'y a plus qu'à trouver une condition pour arrêter la boucle une fois tous les résultats parcourus.

Condition d'arrêt de la boucle

- On sait que dans la boucle while , repeat ou bien Loop il faut utiliser une condition d'arrêt de la boucle.
- Pour cela , on va utiliser les gestionnaires d'erreur, **NOT FOUD** qui change la valeur d'une variable locale. Cette variable locale vaut 0 au départ, et passe à 1 quand le gestionnaire est déclenché (donc quand il n'y a plus de ligne).

Exemple 1 avec boucle while

```
Delimiter |
Create procedure exemple ()
Begin
Declare fin int default 0;
Declare nomstag varchar(20);
Declare  cursor1  CURSOR FOR  select nom from stagiaires ;
Declare continue handler for NOT FOUND set fin=1;
Open  cursor1 ;
while fin!=1 then

FETCH cursor1  into nomstg ;
SELECT nomstg ;

End while;

Close cursor1 ;
End |
Delimiter ;
```

Exemple 1 avec boucle repeat

```
Delimiter |
Create procedure exemple ()
Begin
Declare fin int default 0;
Declare nomstag varchar(20);
Declare  curseur1  CURSOR FOR  select nom from stagiaires ;
Declare continue handler for NOT FOUND set fin=1;
Open  curseur1 ;
repeat

FETCH curseur1  into nomstg ;
SELECT nomstg ;

Until fin=1
end repeat ;
Close curseur1 ;
End |
Delimiter ;
```

Exemple 1 avec boucle LOOP

```
Delimiter |
Create procedure exemple ()
Begin
Declare fin int default 0;
Declare nomstag varchar(20);
Declare  curseur1  CURSOR FOR  select nom from stagiaires ;
Declare continue handler for NOT FOUND set fin=1;
Open  curseur1 ;
Maboucle : LOOP

FETCH curseur1  into nomstg ;
SELECT nomstg ;
if fin=1 then
 leave Maboucle;
End if ;
END LOOP ;
Close curseur1 ;
End |
Delimiter ;
```

Exemple 2

```
delimiter |
CREATE PROCEDURE lister_employes ()
BEGIN
DECLARE finished INTEGER DEFAULT 0 ;
DECLARE v_id INT ;
DECLARE v_nom VARCHAR (100) ;
DECLARE v_prenom VARCHAR (100) ;
DECLARE info VARCHAR (400) DEFAULT '' ;
DECLARE cur_info_employe CURSOR FOR SELECT `N employe`,nom,prenom FROM employes;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
OPEN cur_info_employe;
set info='';
while finished!=1 do
fetch cur_info_employe into v_id,v_nom,v_prenom;
set info=concat(info, '',v_id,'-',v_nom,'-',v_prenom);
end while;
select info;
CLOSE cur_info_employe;
END |
delimiter ;
```

Exemple 2 (suite)

- Procédure utilisant un curseur permettant de parcourir (à l'aide de la boucle **while**) tous les employés et stocke leurs informations dans la variable info.
- Le curseurs fait référence à chaque ligne et stocke les valeurs lue dans les varibales v_id,v_nom et v_prenom

N employe	Nom	Prenom
1	Davolio	Nancy
2	Fabre	André
3	Laffont	Jeanine
4	Parmström	Petra
5	Buchwald	Benjamin
6	Suyama	Michael
7	King	Jonathan
8	Cau	Laurence
9	Damiani	Annabella

- La boucle s'arrête une fois il n y'a plus de lignes à lire.
- L'indicateur de cela est la variable **finished=1** qui est détecté grâce au gestionnaire d'erreurs (handler).

Curseur pour modification

- Si on veut verrouiller les lignes d'une table interrogée par un curseur dans le but de mettre à jour la table, il faut utiliser la clause **FOR UPDATE**.

Exemple d'utilisation de curseur pour modification

```
Delimiter |
Create procedure exemple ()
Begin
Declare fin int default 0;
Declare numstg int ;
Declare nomstag varchar(20);
Declare  cursor1  CURSOR FOR  select  num, nom from stagiaires FOR UPDATE ;
Declare continue handler for NOT FOUND set fin=1;
Open  cursor1 ;
While fin !=1  do
FETCH cursor1  into numstg , nomstg ;

If  nomstg='EL FADIL' then
UPDATE STAGIAIRES SET nom='FADIL' where num=numstg ;
End if ;
END WHILE ;
Close cursor1 ;
End |
Delimiter ;
```