SQL Avancé

Introduction

- Le SQL avancée (**Transact SQL** pour SQL Server, **PL SQL** pour Oracle) est le langage **SQL** plus des **instructions de la logique de programmation**.
- Avec le SQL avancé on peut écrire des instructions comme **les conditions, les boucles**…
- Avec le SQL avancé, on pourra écrire des **procédures stockées** et des **déclencheurs**

Les éléments syntaxique de SQL avancé

1. Les blocs d'instructions
2. L'affichage
3. Les variables
4. Les conditions
5. Les boucles

Bloc d'instructions

- Un bloc d'instructions regroupe plusieurs instructions, équivalent de **{ }** dans les langages de programmation.
- Un bloc en MySQL est défini par les mots-clés **BEGIN** et **END**, entre lesquels on met les instructions qui composent le bloc.
- Les instructions doivent être séparés par ;

Exemple d'un bloc

```
BEGIN
   Instruction 1  ;

   BEGIN
      instruction 2 ;

      BEGIN
         instruction 3 ;
      END ;
   END;

   BEGIN
    instruction  4 ;
   END;

END;
```

Une instruction peut être soit une déclaration de variable, affectation, affichage de message , condition, boucle, requête SQL,…

L'affichage en MYSQL

- Pour afficher un message ou une valeur, on utilise la commande **SELECT**.
- Exemples:

```
SELECT  'bonjour';

SELECT  concat( ' le résultat est égal à : ', x);
```

Les variables

- Les types de variables que vous devez connaitre sont :

1. Les variables locales
2. Les variables de session

Les variables locales

- Les **variables locales**, sont des variables qui peuvent être définies dans un **bloc** d'instructions qui constituent **une procédure stockée** ou **fonction stockée ou déclencheur.**
- Les variables locales **ne peuvent pas être lues en dehors** de leur fonction ou procédure stockée.
- La déclaration d'une variable locale se fait avec l'instruction **DECLARE**
- L'action de déclaration doit se trouver tout au début du bloc d'instructions **dans lequel la variable locale sera utilisée** (donc directement après le **BEGIN**).

Déclaration d'une variable locale

- ***Syntaxe***:

```
DECLARE nom_variable type_variable [ DEFAULT valeur_defaut ];
```

- **Rq :** Si aucune valeur par défaut n'est précisée, la variable vaudra **NULL**

Déclaration d'une variable locale

- **Exemple 1:**

```
DECLARE x int DEFAULT 1;
```

- **Exemple 2:**

```
DECLARE v_date DATE DEFAULT CURRENT_DATE();
```

- **Remarque**:
  - **La portée des variables locales est le bloc d'instructions où a été défini.**
  - Les variables locales n'existent que dans le bloc d'instructions dans lequel elles ont été déclarées. Dès que le mot-clé **END** est atteint, toutes les variables locales du bloc sont détruites.

Affecter une valeur à une variable

- L'affectation d'une valeur à une variable se fait avec l'instruction SET
- Syntaxe:

```
SET Variable=valeur;
```

- Exemple:

```
SET Mavar = (SELECT nom FROM personne WHERE id=11) ;
```

Affecter une valeur à une variable

- L'affectation d'une valeur à une variable peut se fait également avec l'instruction INTO

- ***Exemple 1:***

```
SELECT nom INTO Mavar  FROM personne WHERE id=11 ;
```

- ***Exemple 2:***

```
SELECT nom , prenom INTO x , y  FROM personne WHERE id=11 ;
```

Les variables de session

- Une variable de session est une variable définie par l'utilisateur (variable utilisateur)
- Une variable de session est définie **en dehors** de procédure ou fonction stockée.
- Une variable session doit commence par **@**
- Ce type de variable **ne nécessite pas de déclaration**, peut être utilisée dans toute requête ou instruction SQL.

Les variables de session

```
SET @id = 'A';

SELECT CONCAT(@id, 'B');
-- Resultat: AB

-- Assigner un nombre à la variable session
SET @id = 13;

SELECT @id * 3;
-- Resultat: 39

SELECT CONCAT(@id, 'B');
-- Resultat: 13B
```

Les structures conditionnelles

- Les structures conditionnelles permettent de déclencher une action ou une série d'instructions lorsqu'une condition est réalisée.

- MySQL propose deux structures conditionnelles : **IF** et **CASE**.

L'instruction conditionnelle : IF

- Syntaxe:

```
IF          condition1    THEN   instructions

[ELSEIF    condition2   THEN   instructions]
[ELSEIF    condition3   THEN   instructions]
.....
[ELSE   instruction]

END IF;
```

La syntaxe la plus simple:

```
IF  condition  THEN  instruction
END IF;
```

L'instruction conditionnelle : IF

- Exemple:

```
declare X int ;

SELECT   COUNT(*)   INTO  X   FROM stagiaires where
year(DateNaiss)=2002  ;

If   x>300    then select  'nombre de stagiaires est dans la
moyenne' ;
Else    select 'le nombre est petit' ;
End if ;
```

Instruction conditionnelle : case

```
CASE variable

WHEN possibilité1 THEN  instructions
WHEN possibilité2 THEN  instructions
....
[ELSE Instruction]

END CASE;
```

Instruction conditionnelle : case

- Exemple:

```
Declare  x  int ;

Set  x=(select  year(curedate())- year(dateNaissance)
from stagiaires where numinsc=1000);

Case  x
When 20 then   select   'vous êtes mature '   as message;
When 18 then   select  'vous êtes majeur'   as message;
Else select   'vous avez encore besoin de l''assitance de
vos parents'    as message;
End case;
```

Les boucles

- Les boucles vont nous permettre de répéter une instruction plusieurs fois
- Il existe plusieurs boucles:
  - La boucle while
  - La boucle repeat
  - La boucle Loop

La boucle while

- La boucle WHILE permet de répéter une série d'instructions **tant que la condition donnée reste vraie**.
- syntaxe:

```
WHILE condition DO

 Instructions

END WHILE;
```

La boucle while

- Exemple:

```
DECLARE  i  INT  DEFAULT  0;

WHILE  i < 9  DO

  SET  i = i + 1;
  SELECT  i;

END WHILE;
```

La boucle repeat

- La boucle **REPEAT** travaille en quelque sorte de manière opposée à WHILE, puisqu'elle exécute des instructions de la boucle **jusqu'à ce que la condition donnée devienne vraie**.
- Syntaxe:

```
REPEAT
      instructions
UNTIL   condition

END REPEAT
```

La boucle repeat

- Exemple:

```
DECLARE   i   INT   DEFAULT   0;

REPEAT

   SET i = i + 1;
   SELECT  i;


UNTIL i >= 9

END REPEAT;
```

La boucle LOOP

- La commande **LOOP** ultra simple fait juste ça. Elle boucle indéfiniment jusqu'à ce que vous lui demandiez d'arrêter avec la commande **LEAVE**.
- Syntaxe:

```
[label:] LOOP
instructions
END LOOP [label]
```

La boucle loop

- Exemple:

```
DECLARE iter INTEGER DEFAULT  0;

Iterloop : LOOP
   IF iter < 9 THEN
      SET iter = iter + 1;
      SELECT iter;
   ELSE
      LEAVE iterloop;
   END IF;
END LOOP iterloop;
```

L'instruction leave

- Les commandes **LEAVE** est similaire à la commandes **break** utilisée dans d'autres langages de programmation.

- Elle peut être appliquée à n'importe quelle boucle **WHILE**, **LOOP** ou **REPEAT** qui sont actuellement actives en en faisant référence au label donné à la boucle.