Introduction aux bases de donnés

Les commande UPDATE et DELETE
La commande CREATE TABLE: IDENTITY

Les commandes: UPDATE DELETE

Plan de la séance

- Retour sur la dernière séance:
  - Point de vue de l'étudiant
  - Point de vue de l'enseignant.
- Rappels:
  - CREATE TABLE
  - INSERT INTO
- La commande UPDATE
- La commande DELETE
- La commande CREATE TABLE: l'option IDENTITY
- Suite du labo2

Rappel: CREATE TABLE

- La commande CREATE TABLE permet de créer une table dans une base de données.
- Pour créer une table nous avons besoin de connaître:
  - La liste des attributs de la table
  - Le type de données pour chaque attribut (NUMBER, VARCHAR2(n), CHAR(n), DATE, etc…)
  - Les contraintes d'intégrité, s'il y'en a, pour chaque attribut (PRIMARY KEY, CHECK, NOT NULL, UNIQUE..)
  - Et l'ensemble des contraintes d'intégrité sur la table, s'il y'en a.
- Nous devons connaître aussi le nom de la table qui sera créée.

Rappel: INSERT INTO

- Cette commande permet d'insérer des données dans une table, une ligne à la fois. C'est une commande du DML (Data Manipulation Language).
- C'est la commande qui permet peupler la base de données
- Deux syntaxes sont possibles pour cette commande

```
INSERT INTO <nom_de_table> VALUES (<liste de valeurs>);

INSERT INTO <nom_de_table>(<nom_de_colonne>) VALUES (<liste_de_valeurs>);
```

La commande UPDATE

- Tout comme la commande INSERT INTO, la commande UPDATE est une commande du DML (Data Manipulation Language)
- La commande UPDATE permet d'effectuer des modifications des **données sur une seule table.**
- Contrairement à la commande INSERT INTO, la commande UPDATE permet de la modification de plusieurs enregistrements (lignes) en même temps
- Lors de la modification des données, les contraintes d'intégrité doivent être respectées

Syntaxe:

```
UPDATE <nom_de _table> SET
(<nom_de_colonne>=<nouvelle_valeur>)
[WHERE <condition>];
```

Exemple

```
UPADTE employes SET salaire = salaire*1,01 where empno =10
```

La commande UPDATE

- La clause WHERE a le même rôle que pour le SELECT. Elle permet de cibler les lignes à mettre à jour. (à modifier)
- Il est possible de mettre à jour plusieurs colonnes en même temps comme le montre la syntaxe.

UPDATE employesinfo SET salaire = salaire +(salaire*0.5)
WHERE nom ='Fafar'; → ajoute 1% du salaire à l'employé dont le nom est Fafar.

UPDATE employesinfo SET salaire = salaire +(salaire*0.1) → ajoute 1% du salaire à tous les employés

UPDATE employesinfo SET salaire = salaire +(salaire*0.1), commission =200 ; → on met à jour le salaire et la commission pour tous les employés

La commande DELETE

- Tout comme les commande INSERT INTO et UPDATE la commande DELETE est une commande du DML (Data Manipulation Language)
- La commande DELETE permet de supprimer une ou plusieurs lignes d'une table
- Lors de la suppression des données, les contraintes d'intégrité référentielle doivent être respectées (voir plus loin)
- Syntaxe:

```
DELETE FROM <nom_de _table>
[WHERE <condition>];
```

Exemple

```
DELETE FROM employes WHERE empno =10
DELETE FROM employes WHERE adresse LIKE '%Montréal%'
```

Officialiser ses transactions

- Les opérations DML, une fois exécutées doivent être confirmées pour que la sauvegarde soit effective dans la base de données.
- COMMIT: elle permet d'officialiser une mise à jour (INSERT, UPDATE, DELETE) ou une transaction (série de commandes de manipulation de données effectuées depuis le dernier COMMIT) sur la base de données.
- ROLLBACK : permet d'annuler une transaction avant un COMMIT ; une fois le COMMIT exécuté aucun ROLLBACK n'a d'effet sur la BD

```
insert into clients (nom, prenom) values('LeRoy','Gibbs');
insert into clients (nom, prenom) values('LeChat','Simba');
insert into clients (nom, prenom) values('LeBeau','Cheval');
COMMIT
```

Conclusion, les commandes DML

Il existe 3 commandes du DML (Data Manipulation Language):

- La commande INSERT INTO, qui permet d'ajouter des données dans une table une ligne à la fois. Lors des insertions les contraintes d'intégrités doivent être respectées.
- la commande UPDATE, qui permet de modifier les données d'une table. La clause WHERE est utilisée pour cibler les lignes à modifier. Lors des modifications, les contraintes d'intégrité doivent être respectées.
- La commande DELETE, qui permet de supprimer des données dans une table. La clause WHERE est utilisée pour cibler les lignes à supprimer. Lors des modifications, les contraintes d'intégrité référentielles doivent être respectées

Après exécution des opérations DML, vous devez exécuter COMMIT pour officialiser les transactions.

Pour annuler une opération DML on exécute un ROLLBACK avant le COMMIT. Après un COMMIT, aucun ROLLBACK n'a d'effet.

CREATE TABLE: L'option IDENTITY, BY DEFAULT

À partir de la version 12c de la base de données, Oracle a introduit l'option IDENTITY pour incrémenter automatiquement la clé primaire. L'avantage d'avoir une telle option est que la clé primaire ne sera jamais dupliquée. On diminue les erreurs

L'option IDENTITY convient surtout pour les tables ayant comme clé primaire un numéro séquentiel : Clients, fournisseurs, joueurs, commandes, factures etc.. Le type de données pour la clé primaire est **NUMBER.**

```
CREATE TABLE clients
(
id_client number(4,0) GENERATED BY DEFAULT AS IDENTITY,
nom varchar2(30) not null,
prenom varchar2(30),
CONSTRAINT pk_client PRIMARY KEY(id_client)
);
```

CREATE TABLE :L'option IDENTITY, BY DEFAULT

Explications :

- Le id_client est une clé primaire qui s'insère automatiquement. Elle commence à 1 et s'incrémente de 1.
- Lorsque j'insère ces lignes, je n'ai pas d'erreurs puis que la clé s'insère automatiquement

```
insert into clients (nom, prenom) values('LeRoy','Gibbs');
insert into clients (nom, prenom) values('LeChat','Simba');
insert into clients (nom, prenom) values('LeBeau','Cheval');
```

Important: Le client LeRoy, Gibbs aura le numéro 1, le client Lechat Simba aura le numéro 2 …etc…

- À chaque insertion la clé monte de 1. Il y a une séquence
- Je peux briser la séquence en faisant une insertion manuelle de la clé primaire comme suit :

```
insert into clients(id_Client, nom, prenom)  values (10, 'Ce nouveau','Client');
```

L’option IDENTITY, BY DEFAULT START WITH

Nous pouvons décider la valeur de départ pour le premier enregistrement inséré et la valeur de l’incrément. Dans l’exemple suivant, on commence à 10, et on incrémente de 2

```
CREATE TABLE clientsClg
(
id_client number(4,0) GENERATED BY DEFAULT AS IDENTITY START WITH 10
INCREMENT BY 2,
nom varchar2(30) not null,
prenom varchar2(30),
CONSTRAINT pk_clientclg primary key(id_client)
);
```

insert into clientsclg (nom, prenom) values('LeRoy','Des Singes')

insert into clientsclg (nom, prenom) values('Lefou','Du Village')

insert into clientsclg (nom, prenom) values('Soleil','Vert’)

Le client LeRoy Des Singes aura le numéro 10

Le client Lefou Du Village aura le numéro 12

Les commandes UPDATE DELETE

Conclusion

Questions