Introduction aux bases de donnés

Les commandes DML

Les commandes: CREATE TABLE INSERT INTO

Plan de la séance

- Retour sur la dernière séance:
  - Point de vue de l'étudiant
  - Point de vue de l'enseignant.
- Rappel: la commande CREATE TABLE
- La commande INSERT INTO
- La commande UPDATE
- La commande DELETE

La commande INSERT INTO

Définition et syntaxe

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

Syntaxe 1, cette syntaxe indique que l'on doit fournir les valeurs valides pour toutes les colonnes de la table

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

Exemple: pour la table de l'exemple 5

```
INSERT INTO Employes values(1,'Leroy','Rémi', 60000);
```

La commande INSERT INTO

Définition et syntaxe

Syntaxe2 , cette syntaxe indique que l'on doit fournir les valeurs valides pour chaque colonne indiquée dans la commande INSERT INTO

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

Exemple: pour la table de l'exemple 5

```
INSERT INTO Employes (empno, nom, salaire) values(1,'Leroy', 60000);
```

INSERT INTO, Important

Important:

- La commande INSERT INTO insère une ligne à la fois.
- Aucune insertion n'est possible si les contraintes d'intégrités ne sont pas respectées.

INSERT INTO Employes (nom, salaire) values (Leroy', 60000); ne marchera pas car il n' y a pas de empno . Le empno est obligatoire (NOT NULL).

INSERT INTO Employes (empno, nom, salaire) values(1,'Leroy', 60000); va marcher

INSERT INTO Employes (empno, nom, salaire) values(1,'Lesinge', 80000); Ne marchera pas, car empno doit-être unique

INSERT INTO Employes (empno, nom, salaire) values(2,'Lesinge', 20000); Ne marchera pas, car le salaire est plus petit que 50000

INSERT INTO, Important

Important:

- Une valeur de type caractère (CHAR ou VARCHAR2) doit être mise entre apostrophes. Si la chaine de caractère contient des apostrophes, ceux-ci doivent être doublés.

```
INSERT INTO employes VALUES (3,'O''Brian','Kévin',100000);
```

- Le type numérique (NUMBER) est saisi en notation standard. La virgule décimale est remplacée par un point lors de la saisie

```
INSERT INTO employes VALUES (4,'O''Neil','Carl',800000.88)
```

- Le type date doit être saisie selon la norme américaine (JJ-MMM-AA pour 12 jan 21) et entre apostrophes. Pour saisir une date dans n'importe quel format, il faut s'assurer de la convertir dans le format avec la fonction TO_DATE

Si vous ne connaissez pas le format de la date, exécutez: **SELECT SYSDATE FROM DUAL;**

- Lorsque la valeur d'une colonne n'est pas connue et que celle-ci possède une contrainte de NOT NULL, alors on peut saisir le NULL entre apostrophe comme valeur pour cette colonne.

INSERT INTO, Important

Important:

- Il est possible d'utiliser une expression arithmétique dans la commande INSERT INTO à condition que cette colonne soit de type numérique.

```
INSERT INTO employes VALUES(5,'Patoche','Alain',(50000 +(50000*0.1)/2));
```

- Si des valeurs dans certaines colonnes ne doivent pas être saisies (contiennent des valeurs par défaut) alors la précision des colonnes dans lesquelles la saisie doit s'effectuer est obligatoire. Noter que les valeurs à saisir doivent être dans le même ordre de la spécification des colonnes. Voir syntaxe 2

Table personnes de l'exemple 4 (dans la colonne Villes, Montréal sera inséré par défaut)

```
INSERT INTO personnes(numero,nom,prenom,courriel) VALUES(11,'Saturne', 'Lune','Saturne@gmail.com');
```

INSERT INTO, l'option IDENTITY BY DEFAULT

Voici la table Clients

```
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)
);
```

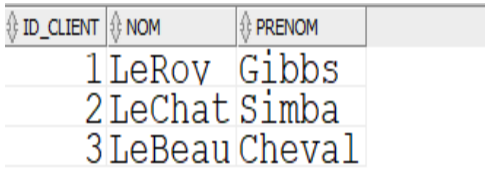
- Le id_client est une clé primaire qui s'insère automatiquement. Elle commence à 1 et s'incrémente de 1.
- Lorsque j'insère ces lignes, je n'ai pas d'erreurs puis que la clé s'insère automatiquement

Dans les insertions suivantes, il n' y a pas la colonne id_client. Le système insère un numéro 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');

INSERT INTO, l'option IDENTITY BY DEFAULT

Lorsque j'exécute un SELECT* FROM Clients, j'obtiens le résultat ci-après



ID_CLIENT	NOM	PRENOM
1	LeRov	Gibbs
2	LeChat	Simba
3	LeBeau	Cheval

A chaque insertion, la clé primaire augmente de 1.

Je peux briser la séquence en faisant:

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

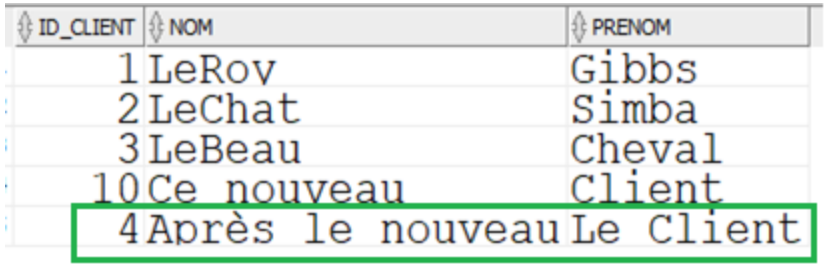
Après le SELECT* FROM Clients, j'obtiens le résultat ci-après.

ID_CLIENT	NOM	PRENOM
1	LeRov	Gibbs
2	LeChat	Simba
3	LeBeau	Cheval
10	Ce nouveau	Client

Et si je reviens à la séquence, est-ce le id_client est 4 ou 11 ?

insert into clients (nom, prenom) values('Après le nouveau','Le Client');

Remarque: lorsque la séquence arrive à 10, il y 'aura une erreur



ID_CLIENT	NOM	PRENOM
1	LeRov	Gibbs
2	LeChat	Simba
3	LeBeau	Cheval
10	Ce nouveau	Client
4	Après le nouveau	Le Client

INSERT INTO, l'option IDENTITY BY DEFAULT START WITH .. INCREMENT BY

Voici la table ClientsCLG

```
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)
);
```

- Le id_client est une clé primaire qui s'insère automatiquement. Elle commence à 10 et s'incrémente de 2.
- Lorsque j'insère ces lignes, je n'ai pas d'erreurs puis que la clé s'insère automatiquement

Dans cet exemple, la clé primaire commence à 10 et s'incrémente de 2

- 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');

ID_CLIENT	NOM	PRENOM
10	LeRoy	Des Singes
12	Lefou	Du Village
14	Soleil	Vert

INSERT INTO, l'option IDENTITY ALWAYS

Voici la table fournisseurs

```
CREATE TABLE fournisseurs
(id_fournisseur number(4,0) GENERATED ALWAYS AS IDENTITY,
nom varchar2(40) not null,
prenom varchar2(30) ,
constraint pk_fournisseur primary key (id_fournisseur));
```

Dans l'exemple suivant, la clé primaire est toujours IDENTITY, donc aucune insertion manuelle de la clé primaire n'est possible. Si je fais:

- insert into fournisseurs values (10,'Yacoub','Saliha');

J'obtiens l'erreur suivante:

```
Erreur à la ligne de commande: 44 Colonne: 1
Rapport d'erreur -
Erreur SQL : ORA-32795: impossible d'insérer la valeur dans une colonne d'identité avec les mots-clés GENERATED ALWAYS
32795.0000 -  "cannot insert into a generated always identity column"
*Cause:    An attempt was made to insert a value into an identity column
           created with GENERATED ALWAYS keywords.
*Action:   A generated always identity column cannot be directly inserted.
           Instead, the associated sequence generator must provide the value.
```

INSERT INTO, Important

Important:

- Il est possible d'utiliser des insertions à partir d'une table existante (commande SELECT et une sous-requête -----à voir plus loin)
- Il est possible d'utiliser une séquence pour l'insertion automatique d'un numéro séquentiel pour une colonne (à voir plus loin)
- Après les insertions, il est recommandé d'exécuter la commande COMMIT pour enregistrer vos données (à voir plus loin)

INSERT INTO, Conclusion

Conclusion

- INSERT INTO est une commande DML qui permet d'insérer des données dans une table. Une ligne à la fois.
- La commande INSERT INTO ne pourra pas s'exécuter si les contraintes d'intégrités ne sont pas respectées. En tout temps, il faut vérifier les contraintes d'intégrité.
- Pour que les insertions (les données) sont définitivement enregistrées dans la BD, il suffit d'exécuter un COMMIT.

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.

Les commandes CREATE TABLE INSERT INTO

Conclusion

Questions