420-KHG-LG, Semaine 02-Séance2

Rappels : les commandes SQL

Il existe principalement trois types de commandes SQL

1. les commandes de définition des données (LDD, langage de définition des données)

   Ces commandes permettent de créer ou de modifier un objet de la base de données : CREATE, ALTER, DROP, MODIFY,

2. les commandes de manipulation des données (LMD,Langage de Manipulation des Données)

   Ces commandes permettent de faire la mise à jour ou la consultation des données dans une table de la base de données.

   SELECT, UPDATE, INSERT, DELETE

3. les commandes de control de données (LCD) :

   Exemple : GRANT et REVOKE

4. Les commandes de transaction (TCL), qui permettent de gérer les modifications apportées aux données de la base de données.

   Exemples : COMMIT, et ROLLBACK

Définition de données

Création de tables : La commande <CREATE TABLE>

Convention d'écriture :

Les <> indique une obligation

Les () pourra être répété plusieurs fois, il faut juste les séparer par une virgule

Les [] indique une option.

Syntaxe simplifiée :

```
CREATE  TABLE <nom_de_table> (<nom_de_colonne><type_de_données>);
```

Syntaxe générale

```
CREATE  TABLE <nom_de_table> (<nom_de_colonne> <type_de_données>
[DEFAULT <valeur>]
[
[CONSTRAINT <nom_de_contrainte>]
NULL
Ou
NOT NULL
OU
UNIQUE
OU
PRIMARY KEY
OU
FOREIGN KEY
ou
REFERENCES <Nom_de_Table><nom_de_colonne>
Ou
[ON DELETE CASCADE]
Ou
CHECK <nom_de_condition>
]
);
```

Définitions et exemples :

Une contrainte d'intégrité est une règle sur la table qui permet d'assurer que les données stockées dans la base de données soient cohérentes par rapport à leur signification. Avec oracle, on peut implémenter plusieurs contraintes d'intégrité
Au niveau de la table : Lorsque la contrainte porte sur plusieurs colonnes simultanément (clé primaire composée), il est obligatoire de déclarer la contrainte sur la table.

Au niveau de la colonne : se fait pour l'ensemble des contraintes à condition qu'elle porte sur une seule colonne.

Voici les explications des contraintes définies avec le CREATE plus haut :

La contrainte de PRIMARY KEY: permet de définir une clé primaire sur la table. Lorsque la clé primaire est une clé primaire composée, la contrainte doit être définie au niveau de la table et non au niveau de la colonne. Les attributs faisant partie de la clé doivent être entre parenthèse. La contrainte de PRIMARY KEY assure également les contraintes de NOT NULL et UNIQUE

Exemple 1 : Clé primaire au niveau de la colonne

```
CREATE TABLE Etudiants
(
numad NUMBER(10) CONSTRAINT pketudiant PRIMARY KEY,
Nom VARCHAR2(20) NOT NULL,
Prenom VARCHAR2 (20)
);
```

Remarquez :

1. Chaque définition de colonne se termine par une virgule, sauf la dernière colonne
2. La contrainte primary key est définie par le mot réservé CONSTRAINT suivi du nom de la contrainte. Dans cet exemple la contrainte est définie en même temps que la colonne NUMAD, donc c'est une contrainte sur la colonne.

Exemple2: Clé primaire au niveau de la table

```
CREATE TABLE Etudiants
(
numad NUMBER(10),
Nom VARCHAR2(20) NOT NULL,
Prenom VARCHAR2 (20),
CONSTRAINT pketudiant PRIMARY KEY (numad)
);
```

Remarquez :

1. Chaque définition de colonne se termine par une virgule, sauf la dernière colonne
2. La contrainte primary key n'est pas définie en même temps que la colonne NUMAD.(il y a une virgule qui termine la définition de la colonne numad)
3. La contrainte de primary key est définie comme si on définirait une colonne. C'est une contrainte sur la table.
4. Pour dire quel est l'attribut (colonne) qui est primary key il faudra le préciser entre parenthèses.
5. L'écriture suivante est équivalente à l'écriture de **l'exemple 2** , ce qui veut dire que je peux placer la définition de la contrainte où je veux.
6. Si vous avez à définir des contraintes au niveau table (ce qui est recommandé) Alors, la définition de ces contraintes doivent être à la fin de la définition des colonnes (voir exemple 2).

Exemple 3

```
CREATE TABLE Etudiants
(
numad NUMBER(10),
CONSTRAINT pketudiant PRIMARY KEY (numad),
Nom VARCHAR2(20) NOT NULL,
Prenom VARCHAR2 (20)
);
```

La contrainte CHECK :
Indique les valeurs permises qui peuvent être saisies pour la colonne (champ ou attribut) lors de l'entrée des données ou une condition à laquelle doit répondre une valeur insérée. La condition doit impliquer le nom d'au moins une colonne. Les opérateurs arithmétiques (+,*,/,-), les opérateurs de comparaisons et les opérateurs logiques sont permis.

La contrainte DEFAULT : indique la valeur par défaut que prendra l'attribut si aucune valeur n'est saisie.

La contrainte NOT NULL : indique que la valeur de la colonne ou de l'attribut est obligatoire. Si cette contrainte n'est pas précisée alors par défaut la valeur est NULL.

La contrainte UNIQUE : indique que les valeurs saisie pour les colonnes (champs ou attributs) doivent être unique Ce qui veut dire pas de Doublons.

Exemple 1

```
CREATE TABLE personne
(
num NUMBER CONSTRAINT pkpersonne PRIMARY KEY,
nom VARCHAR2 (15),
prenom VARCHAR2 (15),
ville VARCHAR2 (20) DEFAULT 'Montréal' CONSTRAINT ckville CHECK
(ville IN ('Montréal','Laval','Saint-Jérôme'))
);
```

Exemple 2 la contrainte est sur la table

```
CREATE TABLE  employes
(
numep NUMBER(4,0)  CONSTRAINT PKemp PRIMARY KEY,
nom VARCHAR2(30) NOT NULL,
prenom VARCHAR2(30) CONSTRAINT ckprenom NOT NULL,
codedep CHAR(3) DEFAULT 'INF' CONSTRAINT ckdep CHECK (codedep IN
('INF','RSH','CMP','GEM')),
salaire NUMBER (8,2) CHECK (SALAIRE > 20000),
echelon number(2,0),
constraint ckechelon CHECK (echelon between 10 and 25)
);
```

Remarquez :

1. Nous ne sommes pas obligés de donner un nom pour les contraintes. MAIS toutes les contraintes ont un nom dans le système. Si vous n'avez pas donné vous-même un nom à votre contrainte, le système (le SGBD) va s'en charger à votre place.
2. Mis à part les contraintes de UNIQUE, DEFAULT et NOT NULL, moi, je vous oblige à donner un nom significatif à toutes vos contraintes.
3. La contrainte CHECK peut être définie sur la table ou sur la colonne.

Types de données manipulés :

Le type de données représente la première contrainte à préciser lors de la création de table. Pour chaque attribut ou champs de la table, on doit préciser le type de données.
Les principaux types manipulés sont présentés dans le tableau suivant.

Type de données	Explications et exemple
VARCHAR2(**n**)	Chaîne de caractères de longueur variable. La taille maximale de cette chaîne est déterminée par la valeur **n** et peut atteindre 4000 caractères (bytes). La longueur minimale est 1. la précision du n est obligatoire. Exemple : NomPrograme VARCHAR2(20)
CHAR(n)	Chaine de caractères de longueur fixe allant de 1 à 2000 caractères (bytes). La chaîne est complétée par des espace si elle est plus petite que la taille déclarée Exemple CodeProgramme CHAR(3).
LONG	Données de type caractère pouvant stocker jusqu'à 2 gigabytes Exemple :Introduction LONG
NUMBER(n,d)	Pour déclarer un nombre sur maximum **n** chiffres (positions) dont **d** chiffres (positions) sont réservés à la décimale. **n** peut aller de 1 à 38.
DATE	Donnée de type date située dans une plage comprise entre le 1er janvier 4712 av JC et le 31 décembre 9999 ap JC stockant l'année, mois, jour, heures, minutes et secondes
LONG RAW	Chaîne de caractères au format binaire pouvant contenir jusqu'à 2 gigaoctet Exemple :Photo LONG RAW
BLOB	Binary Large Object : Gros objet binaire pouvant aller jusqu'à 4 gigaoctet : Exemple Image BLOB
CLOB	A character large object : Chaîne de caractère de longueur maximale allant jusqu'à 4 gigaoctet :

Pour plus d'information consulter :
http://download.oracle.com/docs/cd/E11882_01/server.112/e10592/sql_elements001.htm#i45441

Manipulation de données

La commande INSERT INTO

Syntaxe 1 : la syntaxe qui suit permet d'insérer des valeurs pour toute une rangée (une ligne ou un enregistrement) dans une table

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

Exemple

INSERT INTO EmployesInfo VALUES (20,'Fafar','Patrice',40000);

Syntaxe 2

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

Exemple

INSERT INTO EmployesInfo (NumEmp, nom) VALUES (12,'Lebeau');

La commande INSERT est la première commande exécutée après avoir créé une table. Cette commande permet de saisir des données dans une table **une rangée à la fois.**

- Lors de l'insertion des donné dans l'ensemble des colonnes de la tables (toutes les colonnes), il n'est pas nécessaire de préciser les noms de celles-ci. Voir syntaxe 1
- 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
- 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.
- Le type numérique (NUMBER) est saisi en notation standard. La virgule décimale est remplacée par un point lors de la saisie

- Le type date doit être saisie selon la norme américaine (JJ-MMM-AA pour 12 jan 99) 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
- 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.
- Il est possible d'utiliser une expression arithmétique dans la commande INSERT à condition que cette colonne soit de type numérique.
- 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 des séquences pour l'insertion automatique d'un numéro séquentiel pour une colonne
- Après les insertions, il est recommandée d'exécuter la commande COMMIT (à voir plus loin)

La commande UPDATE

Syntaxe simplifiée :

UPDATE <nom_de _table> SET
<nom_de_colonne>=<nouvelle_valeur>;

La commande UPDATE permet d'effectuer des modifications des données sur une seule table. Cette modification peut porter sur une ou plusieurs lignes. (Enregistrement)
Lors de la modification des données, les contraintes d'intégrité doivent être respectées. Il est impossible de modifier une valeur de la clé primaire si cette valeur est référée par une valeur de la clé étrangère
De plus, il faut tenir compte de s valeur définie par les contraintes CHECK et NOT NULL

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

- Il est possible d'utiliser des modifications à partir d'une table existante (commande SELECT et une sous-requête -----à voir plus loin)

Utilisation de la clause WHERE

La cluse WHERE permet de fixer la condition sur les données de mise à jour (UPDATE). Cette clause est utilisée également avec DELETE et SELECT.

UPDATE <nom_de _table> SET
<nom_de_colonne>=<nouvelle_valeur>
WHERE <condition>;

Les opérateurs utilisés dans la condition sont les même que ceux du WHERE dans la commande SELECT. (Voir cours séance 1)

```
UPDATE employes SET nom ='BIDON', salaire = 44000 WHERE numemp =10;
```

La commande DELETE

La commande DELETE permet de supprimer de la base de données une ou plusieurs rangées d'une table.

Pour des raisons de sécurité, exécuter cette commande jute après la commande SELECT afin d'être certains des enregistrements que l'on va supprimer

Lors de la suppression des données, les contraintes d'intégrité doivent être répétées. .

Il est impossible de supprimer une valeur de la clé primaire si cette valeur est référée par une valeur de la clé étrangère sauf si l'option ON DELETE CASCADE est définie; dans ce cas TOUS les enregistrements de la table enfant (table de la clé étrangère) qui réfèrent la clé primaire supprimée seront supprimer.

Si l'option ON DELETE CASCADE n'est pas définie alors pour supprimer une clé primaire référencée il faut opter pour une des solutions suivante :

- Supprimer d'abord les rangées de la table enfants qui réfèrent la clé primaire, puis supprimer la clé primaire.
- Désactiver la contrainte d'intégrité référentielle (clé étrangère). Cette action est irréversible. Supprimer ensuite la clé primaire.
- Modifier la contrainte d'intégrité référentielle en ajoutant l'option ON DELETE CASCADE

Syntaxe

```
DELETE FROM <nom_de _table>;
```

Cette syntaxe permet de détruire TOUS les enregistrements d'une table

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

Cette syntaxe permet de détruire les enregistrements d'une table répondant à la condition spécifiée dans la clause WHERE
Exemples:
DELETE FROM employesinfo WHERE NUMEMP =30;
DELETE FROM employesinfo WHERE NOM IN ('Blues','Simpson');

Avant toute opération **COMMIT**, faire un SELECT afin de vérifier que nous n'avons pas fait des suppressions par erreur. Utiliser un **ROLLBACK** dans le cas d'une suppression par erreur.

Atelier 2

Objectifs :

- Commencer la requête de CREATION de table
- Exécuter quelques requêtes DML.

Exercice 1

1. Créer la table JOUEURS avec les attributs suivants :

Attributs	Types	Contraintes
NUMJOUEUR	NUMBER	PRIMARY KEY
NOM	VARCHAR2(40)	NOT NULL
PRENOM	VARCHAR2(40)	
SALAIRE	NUMBER(12,2)	Doit être supérieur 500000
NBBUTS	NUMBER	
NBPOINTS	NUMBER	

2. Insérer les enregistrements suivants :

NUMJOUEUR	NOM	PRENOM	SALAIRE	NBBUTS	NBPOINTS
11	Gomez	Scott	1500000	5	20
40	Stastny	Peter	2000000	33	70
13	Poitras	Robert	700000	15	52
26	Patoche	Alain	1200000	35	85
19	Poirier	Juteux	200000	0	15
39	Plouffe	Martin	9000000	0	5
44	Bluff	Rémi	600000	0	15
26	Saturne	Lune	600000	0	15

3. que se passe-t-il lorsque vous essayer d'insérer un joueur avec un salaire de 200000 ?

   _____

   À la place de mettre 200000 saisir 600000, puis insérer l'enregistrement.

4. Que se passe-t-il lorsque vous essayer d'insérer le dernier enregistrement?

   _____

5. Exécuter un COMMIT;

6. Mettre à jour l'attribut SALAIRE pour les joueurs ayant plus de 30 buts. Ajouter 1% du salaire

7. Mettre à jour le salaire des joueurs en ajoutant 1% pour les joueurs ayants 50 points ou plus ou 30 buts et plus.

8. Exécuter un COMMIT.
9. Supprimer les joueurs qui n'ont aucun but. Attention ne pas valider avec un COMMIT
10. Faire un SELECT * pour voir que les enregistrements ont été supprimés.
11. Exécuter un ROLLBACK pour annuler la suppression.
12. Lister les joueurs par ordre de points.
13. Lister les joueurs ayant plus de 50 points ou plus de 30 buts.

(Selon l'avancement du cours)

La contrainte de FOREIGN KEY: cette contrainte indique que la valeur de l'attribut correspond à une valeur d'une clé primaire de la table spécifiée.

La clé primaire de l'autre table doit être obligatoirement crée pour que cette contrainte soit acceptée. La clé primaire de l'autre table et l'attribut défini comme clé étrangère doivent être de même type et de même longueur

On peut également préciser l'option ON DELETE CASCADE qui indique que les enregistrements (occurrences) soient détruits lorsque l'enregistrement correspondant à la clé primaire de la table référencée est supprimé. Si cette option n'est pas précisée alors aucun enregistrement ne sera supprimé de la table qui contient la clé primaire

Important :

La contrainte de clé étrangère est une contrainte sur la table. Elle ne peut être définie sur un attribut (colonne)

Exemple

```
CREATE TABLE programmes
(
codePrg VARCHAR2(3)
CONSTRAINT pk3 PRIMARY KEY,
nomProg VARCHAR2(20)
);
```

```
CREATE TABLE etudiants
(
NumAd NUMBER CONSTRAINT pketudiant PRIMARY KEY,
Nom VARCHAR2(20) NOT NULL,
Prenom VARCHAR2(20),
codePrg VARCHAR2(3),
CONSTRAINT fk1 FOREIGN KEY(codePrg) REFERENCES programme (codePrg)
);
```

Clé primaire composée.

Il arrive qu'une table ait besoin d'une clé primaire composée pour pouvoir identifier les enregistrements. Dans la plus part des cas, ces deux attributs qui composent la clé sont issus deux autres tables. La table qui contient la clé primaire composée est appelée «table de relation».

Exemple

```
CREATE TABLE Resultat
(
NumEtudiant NUMBER(10,0),
codeCours VARCHAR2(10),
Note NUMBER(5,2),
CONSTRAINT pkresultat PRIMARY KEY(NumEtudiant,codeCours)
);
```

La contrainte de clé primaire composées est une contrainte sur la table. Elle ne peut être définie sur un attribut (colonne)

Application (à faire ensemble)

1- Créer la table Programmes avec les colonnes suivantes

Colonnes	Types et contratines
CodeP	Number(3), Primary Key
NomProgramme	Varchar2(40) not null

2- Créer la table Etudiants avec les colonnes suivantes :

Colonnes	Types et contratines
Numad	Number, Primary Key
Nom	Varchar2(40) not null
Prenom	Varchar2(40)
CodeP	Number(3), clé étrangère faisant référence à CodeP de la table Programmes

Insérer les enregistrements suivants dans Programmes

	CODEP	NOMPROGRAMME
1	420	INFORMATIQUE
2	410	ADMINISTRATION
3	430	SANTÉ ANIMALE
4	440	GÉNIE INDUSTRIELLE

Insérer les enregistrements suivants dans Etudiants

	NUMAD	NOM	PRENOM	CODEP
1	10	PATAOCHE	ALAIN	420
2	11	POIRIER	JUTEUX	420
3	12	FAFAR	CHANTAL	420
4	13	SIMPSON	CHRISTIAN	410
5	14	LEFOU	DUVILLAGE	430
6	15	SATURNE	ALAIN	410
7	16	BIEN	SIMON	(null)
8	17	GOUIN	SÉBASTIEN	(null)

Questions

1. Essayer de supprimer le programme informatique de la table programmes. C'est quoi le message erreur.
2. Essayer de modifier le code programme de l'étudiant dont le numad est 10 par le code 650. C'est quoi le message erreur ?
3. Essayer d'insérer un étudiant de votre choix avec le code programme 417. C'est quoi le message erreur ?

Conclusion :