Retour sur la commande CREATE TABLE.

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

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.

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

Exemple

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

CREATE TABLE etudiants (NumAd NUMBER CONSTRAINT pk4 PRIMARY KEY,
Nom VARCHAR2(20), Prenom VARCHAR2(20),
codePrg VARCHAR2(3),
CONSTRAINT fk1 FOREIGN KEY(codePrg) REFERENCES programme (codePrg) );
```

Clé primaire composées·

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 fk1 FOREIGN KEY (NumEtudiant) REFERENCES
etudiants(NumEtudiant),
CONSTRAINT fk2 FOREIGN KEY (codeCours) REFERENCES COURS(codeCours),
CONSTRAINT pk2 PRIMARY KEY(NumEtudiant,codeCours)
);
```

Remarque importante :
Les contraintes de clé primaire composée et de clé étrangère doivent être définies au niveau de la table et non pas au niveau des colonnes.

Autre remarque : la clé primaire à attribut unique (qui n'est pas composée) peut être définie au niveau table ou au niveau colonne.
Exemple 1 : Clé primaire au niveau de la colonne

```
CREATE TABLE Produit (codeProduit NUMBER
CONSTRAINT pk1 PRIMARY KEY,
NomProduit VARCHAR2(20) NOT NULL,
description VARCHAR2 (30));
```

Clé primaire au niveau table.

```
CREATE TABLE Produit (
codeProduit NUMBER ,
NomProduit VARCHAR2(20) NOT NULL,
description VARCHAR2 (30),
CONSTRAINT pk1 PRIMARY KEY (codeProduit)
);
```

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 NOMEMP ='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 suivantes :

- 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.