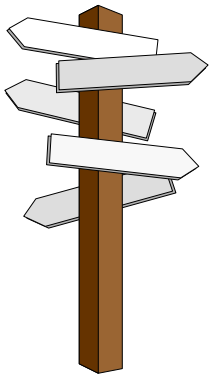
Cours n°6
SQL : Langage de définition des données (LDD)

Chantal Reynaud

Université Paris X - Nanterre UFR SEGMI - IUP MIAGE



Plan

I. Langage de Définition des Données

II. Les types de données

III. Les contraintes

Partie I. Langage de Définition des Données

Le Langage de Définition des Données est la partie de SQL qui permet de décrire les tables et autres objets manipulés par les SGBD.

I. La commande CREATE

II. La commande ALTER TABLE

III. La commande DROP TABLE

I. La commande CREATE TABLE

- L ’ordre CREATE TABLE permet de créer une table en définissant le nom, le type de chacune des colonnes de la table.

CREATE TABLE *table* (*colonne*$_1$ *type*$_1$, *colonne*$_2$ *type*$_2$, ..., *colonne*$_n$ *type*$_n$);

Table est le nom que l ’on donne à la table. *Colonne*$_i$ est le nom d ’une colonne. *Type*$_i$ est le type des données contenues dans *Colonne*$_i$

Exemple :
```
CREATE TABLE STAGE
        (Num-Stage NUMBER(3) NOT NULL PRIMARY KEY,
        Libellé-Stage VARCHAR2(15) NOT NULL,
        Nb-jours NUMBER(2),
        Type-Stage VARCHAR2(15),
        Num-Cat NUMBER);
```

Sous Oracle, un nom doit commencer par une lettre, comporter moins de 30 caractères (lettres, chiffres, _), être différent d ’un autre nom de table ou de vue, être différent d ’un mot réservé SQL. Les lettres minuscules et majuscules sont équivalentes.

I. La commande CREATE TABLE

- Variante

CREATE TABLE *table* (*colonne*$_1$ *type*$_1$, *colonne*$_2$ *type*$_2$, ..., *colonne*$_n$ *type*$_n$)
AS SELECT ...;

Exemple : CREATE TABLE MINI_STAGE
(Numéro NUMBER(3) PRIMARY KEY,
Nom VARCHAR2(15) NOT NULL)
AS SELECT Num-Stage Libellé-Stage FROM STAGE;

Il faut évidemment que les définitions des colonnes de la table créée et du résultat de la sélection soient compatibles en type et en taille.

- On peut également spécifier le mot-clé AS et l 'interrogation directement derrière le nom de la table. Dans ce cas, les noms de colonnes de la table créée auront les mêmes noms, types et tailles que celles de l 'interrogation :

Exemple : CREATE TABLE MINI_STAGE
AS SELECT Num-Stage Libellé-Stage FROM STAGE;

II. La commande ALTER TABLE

- Cette commande permet de modifier la définition d 'une table.

ALTER TABLE *table*
{ ADD
{ MODIFY (*colonne*$_1$ *type*$_1$, *colonne*$_2$ *type*$_2$, ..., *colonne*$_n$ *type*$_n$)
{ DROP

- Exemples :

ALTER TABLE STAGIAIRE
ADD NomEntr VARCHAR2(20);

 Ajout d 'un attribut

ALTER TABLE STAGIAIRE
MODIFY NomEntr VARCHAR2(30);

 Modification dans le type d 'un attribut

ALTER TABLE STAGIAIRE
DROP NomEntr;

 Suppression d 'un attribut

Attention : Certaines modifications peuvent poser des problèmes d 'intégrité dans la base !

III. La commande DROP TABLE

- Cette commande permet de supprimer une table de la base de données. Les lignes de la table et la définition elle-même sont détruites. L ’espace occupé par la table est libéré.

DROP TABLE *table*

- Exemple : DROP TABLE STAGIAIRE;

Partie II. Les types de données

Les types de données Oracle ne suivent pas la norme SQL-2.

I. Types numériques

II. Types chaînes de caractères

III. Types temporels

IV. Types binaires

1. Types numériques

SQL-2

- Nombres entiers : **TINYINT** (sur 1 octet, de 0 à 255), **SMALLINT** (sur 2 octets, de -32768 à 32767), **INTEGER** (sur 4 octets, de -2147483648 à 2147483647)
- Nombres décimaux avec un nombre fixe de décimales : **NUMERIC, DECIMAL**
- Numériques non exacts à virgule flottante : **REAL** (le nombre de chiffres significatifs varie), **DOUBLE PRECISION, FLOAT** (double précision, avec au moins 15 chiffres significatifs).

La définition du nombre de chiffres significatifs varie selon le SGBD.

1. Types numériques

Oracle

- Un seul type numérique **NUMBER**. Par souci de compatibilité, on peut utiliser les types SQL-2 mais ils sont ramenés au type NUMBER

- Lors de la définition d 'une colonne de type numérique, on peut préciser le nombre maximum de chiffres et de décimales qu 'une valeur de cette colonne pourra contenir.

NUMBER
NUMBER (taille_maxi)
NUMBER (taille_maxi, décimales)

- Si le paramètre décimales n 'est pas spécifié, 0 est pris par défaut. La valeur absolue du nombre doit être inférieur à 10^{128}. NUMBER est un nombre à virgule flottante (on ne précise pas le nombre de chiffres après la virgule) qui peut avoir jusqu 'à 38 chiffres significatifs.
- Exemple : NUMBER(6,2) 6 chiffres dont 2 après la virgule

II. Types chaînes de caractères

SQL-2 - Oracle

- Les constantes chaînes de caractères sont entourées par des apostrophes. Si la chaîne contient une apostrophe, celle-ci devra être doublée.
- Il existe deux types pour les colonnes qui contiennent des chaînes de caractères :

  - le type CHAR pour les colonnes qui contiennent des chaînes de longueur constante inférieure à 255 caractères (pour Oracle)

  CHAR (longueur)

  - le type VARCHAR pour les colonnes qui contiennent des chaînes de longueur variable. Tous les SGBD ont une longueur maximale pour ces chaînes (2000 sous Oracle).

  VARCHAR (longueur) ou **VARCHAR2**(longueur) sous Oracle

longueur est la longueur maximale en nombre de caractères qu 'il sera possible de stocker dans le champ

III. Types temporels

SQL-2 - *Oracle*

- SQL-2

DATE : réserve 2 chiffres pour le mois et le jour et 4 pour l ’année
TIME : pour les heures, minutes et secondes
TIMESTAMP : indique un moment précis par une date avec heures, minutes et secondes (6 chiffres après la virgule)

- Oracle :

DATE

Une constante de type DATE est une chaîne de caractères entre apostrophes. Le format dépend des options que l ’administrateur a choisies au moment de la création de la base. S ’il a choisi de franciser la base, le format est jour/mois/année, par exemple '12/03/02' (le format américain par défaut donnerait '13-MAR-02 '). L ’utilisateur peut saisir les dates telles ’1/3/02' mais les dates enregistrées dans la base ont toujours 2 chiffres pour chacun des nombres. Dans Oracle, un type DATE inclut un temps en heures, minutes et secondes.

IV. Types binaires

- SQL-2 n ’a pas normalisé ce type de données
- Ce type permet d ’enregistrer des données telles que les images et les sons, de très grande taille avec divers formats.
- Les différents SGBD fournissent un type pour ces données. Les noms varient : **LONG RAW** pour Oracle, **IMAGE** pour Sybase, **BYTE** pour Informix.

Partie III. Les contraintes

I. Les contraintes de domaine
II. Les contraintes d ’intégrité d ’entité
III. Les contraintes d ’intégrité référentielle
IV. Les assertions
V. La vérification différée des contraintes
VI. Les contraintes particulières

I. Les contraintes de domaine

- Il s 'agit de définir l 'ensemble des valeurs que peut prendre un attribut. Ces contraintes sont décrites dans la définition d 'un attribut, directement après son type et sa longueur.

NOT NULL : on impose que l 'attribut possède une valeur
DEFAULT : on spécifie une valeur par défaut dont le type doit correspondre au type de l 'attribut
UNIQUE : interdit qu 'une colonne contienne deux valeurs identiques

Exemple :

```
CREATE TABLE clients
        (numCli …
        nomCli VARCHAR(25) NOT NULL,
        CaCli INTEGER DEFAULT 0,
        TypeCli VARCHAR(16) DEFAULT  'Particulier '
        …);
CREATE TABLE fournisseurs
        (...
        NomFour CHAR(25) NOT NULL UNIQUE, ..);
```

NomFour et NumFour sont deux clés candidates. NumFour a été choisi comme clé primaire

I. Les contraintes de domaine

- **CHECK**(condition)

Cette clause permet de spécifier une contrainte qui doit être vérifiée à tout moment par les tuples de la table. On peut placer un nombre quelconque de telles clauses dans la définition d 'une table. La clause peut même être placée après la définition de tous les attributs.

Exemples :

```
CREATE TABLE clients
        (…,
        TypeCli VARCHAR(16) DEFAULT  'Particulier'
        CHECK (TypeCli IN ('Particulier',  'Administration', 'PME'))
                CONSTRAINT Type_Clients, …

Prim_CompVARCHAR(8) NOT NULL CHECK (Prim_Comp != Fini_Comp)
                CONSTRAINT PRIMFINI, …


Liv_LigneCde NUMBER DEFAULT 0 CHECK (Liv_LigneCde <= Qté_LigneCde)
                CONSTRAINT LIVQTE, …

CHECK (PU_LigneCde >= 0.8 *
        (SELECT PV_Art FROM articles WHERE NumArt = Art_LigneCde))
                CONSTRAINT PU_LigneCde
```

I. Les contraintes de domaine

- **La déclaration d ’un domaine**

Cette clause permet de spécifier que différents attributs du schéma ont le même ensemble de valeurs et satisfont les mêmes contraintes.

Exemple : CREATE DOMAIN *Qté* NUMBER DEFAULT 0
CHECK (Value >= 0),

Définition des attributs dont les valeurs appartiennent au domaine *Qté* :
Stock_Art *Qté*
Qté_LigneCde *Qté*
...

De cette façon, Stock_Art et Qté_LigneCde héritent des propriétés de *Qté,* valeur entière par défaut nulle, toujours positive ou nulle

La définition d ’un domaine peut comporter un nombre quelconque de contraintes qui peuvent être modifiées par : ALTER DOMAIN

II. Les contraintes d ’intégrité d ’entité

- Elles spécifient la clé primaire d ’une table via la clause **PRIMARY KEY**.
- Une clé primaire doit toujours avoir une valeur déterminée et unique pour la table.
- Quand une clé primaire est constituée de plusieurs attributs (clé segmentée), la clause PRIMARY KEY est placée après la définition des attributs, séparée par une virgule.

Exemples : - NumCli NUMBER **PRIMARY KEY**
- Create TABLE Appartement
(…),
PRIMARY KEY (NumApp, NumImm);

- Remarques :

- tous les attributs d ’une clé segmentée doivent être spécifiés NOT NULL
- PRIMARY KEY peut aussi être séparée de la définition des attributs même s ’il n ’y a qu ’un seul attribut
- Pour une table, il n ’existe qu ’une seul clé primaire
- Dans beaucoup de SGBD, un index est automatiquement construit sur la clé primaire.

III. Les contraintes d ’intégrité référentielle

- **REFERENCES** nom_table_référencée (clé candidate)
- Le nom de la clé candidate est facultatif quand c ’est la clé primaire.
- Exemples :

```
CREATE TABLE Commandes
(NumCde NUMBER PRIMARY KEY,
DateCde DATE NOT NULL,
NumCli NUMBER NOT NULL REFERENCES clients,
MagCde NUMBER NOT NULL REFERENCES magasins);
```

- Utilisation de **FOREIGN KEY** pour spécifier une clé étrangère constituée de plusieurs attributs

- Spécification des actions à effectuer en cas de modification (clause **ON UPDATE**) ou de suppression (clause **ON DELETE**) de valeurs de clés référencées : **CASCADE, SET NULL, SET DEFAULT**

Exemple :

```
NumFour NUMBER REFERENCES fournisseurs
        ON UPDATE CASCADE
        ON DELETE SET NULL;
```

IV. Les assertions

- Il s 'agit de contraintes non directement rattachées à une table. Elles sont utilisées pour spécifier une contrainte portant sur plusieurs tables.

```
CREATE ASSERTION nom_assertion
        ...
```

Exemple :

```
CREATE ASSERTION Val_Glob_Stock
CHECK (
        (SELECT SUM(Art1_PA * Art1_Stock) FROM Articles1) +
        (SELECT SUM(Art2_PA * Art2_Stock) FROM Articles2) <100 000);
```

V. La vérification différée des contraintes

```
[NOT] DEFERRABLE
INITIALLY DEFERRED | IMMEDIATE
```

Par défaut, une contrainte n 'est pas différable.
A chaque définition d 'une contrainte, on peut spécifier le mode de la contrainte : différée ou immédiate.
Si elle est différable, son mode peut être choisi puis modifié dynamiquement à tout instant pendant l 'application;

Spécification du mode d 'une contrainte :

```
SET CONSTRAINTS { liste contraintes | ALL ]
{ DEFERRED | IMMEDIATE }
```

IMMEDIATE : contrainte vérifiée à la fin de chaque requête SQL
DEFERRED : vérification à la fin de chaque transaction. On peut alors spécifier son état initial : INITIALLY DEFERRED (mode DEFERRED au début de la transaction), INITIALLY IMMEDIATE (mode IMMEDIATE au début de chaque transaction).

VI. Les contraintes particulières

Les triggers et les procédures mémorisées complètent les contraintes d ’intégrité en permettant des contrôles et des traitements plus complexes.

- Les triggers (déclencheurs) : procédures compilée, cataloguées dans le dictionnaire, déclenchées automatiquement par des événements liés à des actions sur la base.

```
CREATE TRIGGER nom_trigger
        BEFORE | AFTER
        INSERT | DELETE | UPDATE
        ON nom_table
        FOR EACH ROW
        -- Bloc PL/SQL contenant le traitement à effectuer
```

- Les procédures mémorisées : ensembles d ’instructions SQL précompilées et mémorisées dans le dictionnaire. A la différence d ’un trigger, une procédure mémorisée ne s ’exécute pas à l ’arrivée d ’un événement mais à chaque fois qu ’on l ’appelle. Les procédures mémorisées n ’ont pas fait l ’objet de normalisation.