Le Langage SQL
Le Langage de Définition des Données
(LDD)

Table des Matières

1. Introduction

A un niveau théorique, les instructions SQL peuvent être réparties en trois catégories :

- Le langage de définition de données, ou LDD, qui définit la structure des données.
- Le langage de manipulation de données, ou LMD, qui recherche et modifie les données.
- Le langage de contrôle de données, ou LCD, qui définit les privilèges d'accès accordés aux utilisateurs de la base de données.

La catégorie LDD comprend trois instructions de base

- CREATE, qui permet de **créer** les objets de schéma
- ALTER, qui permet de **modifier** les objets créés
- DROP, qui permet de **supprimer** ces objets

2. La création des tables

Syntaxe complète :

http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_7002.htm#SQLRF01402

La forme simplifiée pour la création d'une table est la suivante :

```
CREATE TABLE [schema.]table
    ( { column  datatype  [DEFAULT expression] [column_constraint] ...
   [, { column  datatype  [DEFAULT expression] [column_constraint] ...
      | table_constraint} ]...)
```

schema

est le nom du schéma qui contiendra la table. Le schéma par défaut est celui de l'utilisateur qui exécute la requête.

table

est le nom de la table à créer.

column

est le nom d'une colonne de la table. *Le nombre de colonnes possibles dans une table est compris entre 1 et 254.*

datatype

est le type de la colonne.

Ce type peut être :

- un type prédéfini d'Oracle (VARCHAR2(taille), NUMBER(précision, échelle), …)
- *un type ANSI*
- *un type prédéfini par l'utilisateur*
- *un type fourni par Oracle*

DEFAULT expression

spécifie une valeur (expr) qui sera affectée à cette colonne si, lors d'un INSERT, on ne lui en précise pas.

column_constraint

est une contrainte d'intégrité qui fait partie de la définition de la colonne.

table_constraint

est une contrainte d'intégrité qui fait partie de la définition de la table.

Exemples

```
CREATE TABLE exe_dept (
   dep_no         NUMBER(6)     CONSTRAINT pk_exe_dept PRIMARY KEY,
   dep_nom        VARCHAR2(50)
);

CREATE TABLE exe_employe (
   emp_no         NUMBER(6),
   emp_prenom     VARCHAR2(20),
   emp_nom        VARCHAR2(25)  CONSTRAINT nn_emp_nom     NOT NULL,
   emp_email      VARCHAR2(25)  CONSTRAINT nn_emp_email   NOT NULL,
   emp_embauche   DATE  DEFAULT SYSDATE  CONSTRAINT nn_emp_embauche NOT NULL,
   emp_salaire    NUMBER(8,2)   CONSTRAINT nn_emp_salaire NOT NULL,
   emp_commission NUMBER(2,2)   CONSTRAINT nn_emp_comm    NOT NULL
         CONSTRAINT ch_emp_comm CHECK (emp_commission IN (0.05, 0.10, 0.15, 0.20)),
   emp_emp_no     NUMBER(6),
   emp_dep_no     NUMBER(4)    CONSTRAINT fk_exe_employe_dept REFERENCES exe_dept (dep_no),

   CONSTRAINT pk_exe_employe      PRIMARY KEY (emp_no),
   CONSTRAINT fk_exe_employe_chef FOREIGN KEY (emp_emp_no) REFERENCES exe_employe (emp_no),
   CONSTRAINT ch_emp_salaire      CHECK  (emp_salaire > 0),
   CONSTRAINT uk_emp_email        UNIQUE (emp_email)
);
```

L'ajout de commentaires

Il est parfois très utile d'associer des commentaires aux tables/vues et aux colonnes d'une table/vue que l'on crée, afin que la structure de ces objets soit facilement compréhensible.

Ceci peut se faire à l'aide de la commande COMMENT ON.

Syntaxe complète :

http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_4009.htm#SQLRF01109

Commentaires sur une table ou une vue :

```
COMMENT ON TABLE [schema.]table/vue  IS 'Commentaire sur la table/vue';
```

Commentaire sur une colonne :

```
COMMENT ON COLUMN table/vue.colonne  IS 'Commentaire sur la colonne';
```

3. La modification de la structure d'une table

Syntaxe complète :

http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_3001.htm#i2109868

Cette commande permet de modifier la définition d'une table, les modifications possibles sont les suivantes :

- ajouter une colonne,
- redéfinir une colonne (type de donnée, taille, valeur par défaut)
- ajouter une contrainte d'intégrité,
- *activer, désactiver une contrainte d'intégrité,*
- *….*

Syntaxe de base :

```
ALTER TABLE [schema.]table
    [ADD     ( column datatype [DEFAULT expr] [column_constraint] ...
             [, column datatype [DEFAULT expr] [column_constraint] ...
              | table_constraint ] ... )
    [MODIFY ( column [datatype] [DEFAULT expr] [column_constraint] ...
             [, column [datatype] [DEFAULT expr] [column_constraint] ...] ...)
    [DROP    ( column [, column ] )
             | table_constraint ] ...
```

Les paramètres des clauses ADD et MODIFY sont les mêmes que dans la commande CREATE TABLE.

Exemples

```
ALTER TABLE exe_employe
   ADD emp_tel VARCHAR2(20)
   ADD CONSTRAINT uk_emp_nom UNIQUE (emp_nom)
   MODIFY emp_prenom  VARCHAR2(25)
   DROP (emp_commission)
```

4. La suppression d'une table

Syntaxe complète :

http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_9003.htm#SQLRF01806

Cette commande permet de supprimer une table.

Syntaxe de base :

```
DROP TABLE [schema.]table [CASCADE CONSTRAINTS] ;
```

Exemple

```
DROP TABLE exe_employe ;
```

5. Les contraintes d'intégrité déclaratives

Une contrainte d'intégrité est une règle qui sera validée lors de l'insertion ou la modification des données de la table.

Les contraintes d'intégrité déclarées lors de la création ou de la modification font partie de la structure même de la table. Elles portent le nom de « contraintes structurelles » et sont inhérentes au modèle.

Recommandations

- Utiliser une contrainte de type `column_constraint` lorsque la contrainte ne concerne qu'un attribut
- Utiliser une contrainte de type `table_constraint` lorsque la contrainte est sur plusieurs attributs

Contraintes dans Oracle

- **PRIMARY KEY**

  Permet de définir la clé primaire de la table. La valeur NULL n'est pas autorisée.

```
Ex :
dep_no  NUMBER(6)  CONSTRAINT pk_exe_dept PRIMARY KEY
ou
CONSTRAINT pk_exe_employe PRIMARY KEY (emp_nom, emp_prenom)
```

- **FOREIGN KEY**

  Définit la référence d'une clé étrangère à la clé primaire d'une autre table.

```
Ex :
emp_dep_no  NUMBER(6)  CONSTRAINT fk_exe_employe_dept REFERENCES exe_dept (dep_no)
ou
CONSTRAINT fk_exe_employe_dept FOREIGN KEY (emp_dep_no) REFERENCES exe_dept (dep_no)
```

- **NOT NULL**

  La valeur NULL n'est pas autorisée :

```
Ex :
emp_nom      VARCHAR2(30)    CONSTRAINT nn_emp_nom     NOT NULL,
emp_prenom   VARCHAR2(30)    CONSTRAINT nn_emp_prenom  NOT NULL,
```

- **UNIQUE**

  Permet de définir des clés candidates qui satisfont l'unicité et la minimalité mais n'ont pas nécessairement été choisies comme clé primaire. Chaque enregistrement de la table doit avoir une valeur différente ou NULL pour cette (ou ces) colonne(s).

```
Ex :
CONSTRAINT uk_emp_nom_prenom_adresse UNIQUE (emp_nom, emp_prenom, emp_adresse)
```

- **CHECK (condition)**

  Ce type de contraintes permet de spécifier les valeurs possibles pour une colonne.
  A priori, la valeur NULL sera acceptée.

```
Ex :
emp_sal   NUMBER(6)    CONSTRAINT ch_emp_sal   CHECK(emp_sal > 0)
emp_npa   NUMBER(4)    CONSTRAINT ch_emp_npa   CHECK(emp_npa BETWEEN 1000 AND 9999)
emp_ville VARCHAR2(25) CONSTRAINT ch_emp_ville CHECK(emp_ville = UPPER(emp_ville))
emp_type  VARCHAR2(5)  CONSTRAINT ch_emp_type  CHECK(emp_type IN ('DIR','CADRE','X'))
```

  NB : Dans l'expression du CHECK on ne peut pas :

  - référencer des colonnes d'autres tables
  - avoir des requêtes faisant référence à d'autres valeurs dans d'autres tuples
  - référencer des « pseudo colonnes » ex. CURRVAL; NEXTVAL etc.,
  - appeler certaines fonctions spéciales (ex : SYSDATE, USER…)
  - avoir des sous requêtes

Quand définir ces contraintes ?

Au moment de la création de la table, mais ce n'est pas toujours possible :

Exemple de cas :

La table référencée par une fk n'existe pas encore.

On peut donc les rajouter après :

```
ALTER TABLE exe_employe
   ADD CONSTRAINT fk_exe_employe_projet FOREIGN KEY (emp_pro_no)
                                        REFERENCES exe_projet (pro_no)
```

Suppression/désactivation d'une contrainte

On peut supprimer une contrainte, mais attention!

```
ALTER TABLE exe_employe
   DROP CONSTRAINT fk_exe_employe_projet ;
```

On peut le faire aussi en cascade sur toutes les contraintes associées… DANGEREUX !

```
ALTER TABLE exe_employe
   DROP CONSTRAINT pk_exe_employe CASCADE ;
```

→ DROP de toutes les contraintes de clé étrangère associées !

On ne peut pas mettre à jour une contrainte.

On peut cependant désactiver puis réactiver une contrainte…

```
ALTER TABLE exe_employe
   DISABLE CONSTRAINT fk_exe_employe_projet ;
```

Le Langage SQL
Les types de données Oracle

Cod	Built-In Datatype	Description
1	VARCHAR2(*size*) [BYTE|CHAR]	Variable-length character string having maximum length *size* bytes or characters. Maximum *size* is 4000 bytes, and minimum is 1 byte or 1 character. You must specify *size* for VARCHAR2. BYTE indicates that the column will have byte length semantics; CHAR indicates that the column will have character semantics.
1	NVARCHAR2(*size*)	Variable-length character string having maximum length *size* characters. Maximum *size* is determined by the national character set definition, with an upper limit of 4000 bytes. You must specify *size* for NVARCHAR2.
2	NUMBER(*p,s*)	Number having precision *p* and scale *s*. The precision *p* can range from 1 to 38. The scale *s* can range from -84 to 127.
8	LONG	Character data of variable length up to 2 gigabytes, or 2^{31} -1 bytes.
12	DATE	Valid date range from January 1, 4712 BC to December 31, 9999 AD.
180	TIMESTAMP (*fractional_seconds_precision*)	Year, month, and day values of date, as well as hour, minute, and second values of time, where *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND datetime field. Accepted values of *fractional_seconds_precision* are 0 to 9. The default is 6.
181	TIMESTAMP (*fractional_seconds_precision*) WITH TIME ZONE	All values of TIMESTAMP as well as time zone displacement value, where *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND datetime field. Accepted values are 0 to 9. The default is 6.
231	TIMESTAMP (*fractional_seconds_precision*) WITH LOCAL TIME ZONE	All values of TIMESTAMP WITH TIME ZONE, with the following exceptions: • Data is normalized to the database time zone when it is stored in the database. • When the data is retrieved, users see the data in the session time zone.

Cod	Built-In Datatype	Description
182	INTERVAL YEAR (*year_precision*) TO MONTH	Stores a period of time in years and months, where *year_precision* is the number of digits in the YEAR datetime field. Accepted values are 0 to 9. The default is 2.
183	INTERVAL DAY (*day_precision*) TO SECOND (*fractional_seconds_precision*)	Stores a period of time in days, hours, minutes, and seconds, where • *day_precision* is the maximum number of digits in the DAY datetime field. Accepted values are 0 to 9. The default is 2. • *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND field. Accepted values are 0 to 9. The default is 6.
23	RAW(*size*)	Raw binary data of length *size* bytes. Maximum *size* is 2000 bytes. You must specify *size* for a RAW value.
24	LONG RAW	Raw binary data of variable length up to 2 gigabytes.
69	ROWID	Base 64 string representing the unique address of a row in its table. This datatype is primarily for values returned by the ROWID pseudocolumn.
208	UROWID [(*size*)]	Base 64 string representing the logical address of a row of an index-organized table. The optional *size* is the size of a column of type UROWID. The maximum size and default is 4000 bytes.
96	CHAR(*size*)[BYTE | CHAR]	Fixed-length character data of length *size* bytes. Maximum *size* is 2000 bytes. Default and minimum *size* is 1 byte. BYTE and CHAR have the same semantics as for VARCHAR2.
96	NCHAR(*size*)	Fixed-length character data of length *size* characters. Maximum *size* is determined by the national character set definition, with an upper limit of 2000 bytes. Default and minimum *size* is 1 character.
112	CLOB	A character large object containing single-byte characters. Both fixed-width and variable-width character sets are supported, both using the CHAR database character set. Maximum size is 4 gigabytes.
112	NCLOB	A character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, both using the NCHAR database character set. Maximum size is 4 gigabytes. Stores national character set data.
113	BLOB	A binary large object. Maximum size is 4 gigabytes.
114	BFILE	Contains a locator to a large binary file stored outside the database. Enables byte stream I/O access to external LOBs residing on the database server. Maximum size is 4 gigabytes