Le langage SQL
Langage de Définition de Données

Ines BAKLOUTI

ines.baklouti@esprit.tn

Ecole Supérieure Privée d'Ingénierie et de Technologies

Plan

Introduction

- SQL = Structured Query Language
  - un langage de définition de données (LDD),
  - un langage de manipulation de données (LMD),
  - un langage d'interrogation de données (LID),
  - un langage de contrôle de données (LCD)
- DDL = Data Definition Language (Langage de Définition de Données LDD)
  - Création d'une structure de données (Create)
  - Modification de la structure d'un objet de la base de données (Alter)
  - Suppression d'une structure de données(Drop)

Plan

Création de table

Syntaxe

CREATE TABLE [schema.]<nom_table>
(<nom_colonne> type [DEFAULT expr],
<nom_colonne> type [DEFAULT expr],
.....
);

Exemple

- CREATE TABLE Etudiants (Netudiant number, nom varchar2(10), prenom varchar2(10)) ;
- DESCRIBE (ou DESC) Etudiants ; –pour voir la description de la table

Table	Column	Type De Données	Longueur	Précision	Echelle	Clé Primaire	Valeur Nullable	Valeur Par Défaut	Commentaire
ETUDIANTS	NETUDIANT	Number	-	-	-	-	✓	-	-
	NOM	Varchar2	10	-	-	-	✓	-	-
	PRENOM	Varchar2	10	-	-	-	✓	-	-
									1 - 3

Création de table

Types de données

Types de données	Description
CHAR [(size [BYTE | CHAR])]	Taille fixe comprise entre 1 et 2000
NCHAR [(size)]	Taille fixe comprise entre 1 et 2000
VARCHAR2 (size)	Taille Variable comprise entre 1 et 4000
NVARCHAR2 (size)	Taille Variable comprise entre 1 et 4000
NUMBER[(precision [, scale]])	Nombre ayant une précision p et une échelle s. La précision est comprise entre 1 et 38. L'échelle varie de -84 à 127
BINARY_FLOAT	32-bit nombre avec virgule flottante. C etype nécessite 5 octets
BINARY_DOUBLE	64-bit nombre avec virgule flottante. C etype nécessite 9 octets
LONG	Données caractères ayant une taille <= 2GO
DATE	Date comprise entre 1/1/4712 AJC et 31/12/999 APJC
TIMESTAMP	Année, mois, jour , heure, minute et seconde, fraction de seconde

Création de table à partir d'une sous-intérrogation

Syntaxe

CREATE TABLE [schema.]<nom_table>
(<nom_colonne> type [DEFAULT expr],
<nom_colonne> type [DEFAULT expr],
. . . . . .
) AS sous_interrogation ;

Exemple

- CREATE TABLE emp as select * from employees where department_id=20 ;
- SELECT * FROM emp ;

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
201	Michael	Hartstein	MHARTSTE	515.123.5555	17/02/96	MK_MAN	13000	-	100	20
202	Pat	Fay	PFAY	603.123.6666	17/08/97	MK_REP	6000	-	201	20

Les contraintes

- Il existe cinq types de contraintes :
  - NOT NULL
  - UNIQUE
  - CHECK
  - PRIMARY KEY
  - FOREIGN KEY
- Il existe deux niveaux de définition de contraintes :
  - contrainte au niveau colonne : contrainte d'intégrité incluse dans la définition de la colonne
  - contrainte au niveau table : contrainte d'intégrité incluse dans la définition de la table
  - syntaxe : CREATE TABLE [schema.]table (nom_colonne type [DEFAULT expr] [contrainte_niveau_colonne], ... [contrainte_niveau_colonne][,...]) ;
- On peut définir une contrainte :
  - lors de la création de la table
  - après la création de la table

Les contraintes

La contrainte NOT NULL

- La contrainte NOT NULL ne peut être définie qu'au niveau de la colonne, pas au niveau de la table

Exemple

- CREATE TABLE Fournisseurs (fournisseur_id number(10) NOT NULL, nom varchar2(20) NOT NULL, contact varchar2(50)) ;
- CREATE TABLE Fournisseurs (fournisseur_id number(10) CONSTRAINT nn_fournisseurs NOT NULL, nom varchar2(20) NOT NULL, contact varchar2(50)) ;

Les contraintes

La contrainte UNIQUE

- Une contrainte d'intégrité de type unique exige que chaque valeur dans une colonne ou dans un ensemble de colonnes soit unique.
- Une contrainte unique autorise la valeur NULL à moins que vous définissiez des contraintes NOT NULL.

Exemple

- Au niveau colonne :
  - CREATE TABLE Fournisseurs (fournisseur_id number(10) UNIQUE, nom varchar2(20) not null, contact varchar2(50)) ;
  - CREATE TABLE Fournisseurs (fournisseur_id number(10) CONSTRAINT uq_fournisseurs UNIQUE, nom varchar2(20) not null, contact varchar2(50)) ;
- Au niveau table :
  - CREATE TABLE Fournisseurs (fournisseur_id number(10), nom varchar2(20) not null, contact varchar2(50), CONSTRAINT uq_fournisseurs UNIQUE(fournisseur_id)) ;

Les contraintes

La contrainte CHECK

- La contrainte Check définit une condition que chaque ligne de la table doit vérifier
- Les expressions suivantes ne sont pas autorisées :
  - appels de fonction telle que SYSDATE
  - interrogations qui font référence à d'autres valeurs dans d'autres lignes.

Exemple

- Au niveau colonne :
  - CREATE TABLE Fournisseurs (fournisseur_id number(10) CHECK (fournisseur_id between 10 and 1000), nom varchar2(20) not null, contact varchar2(50)) ;
  - CREATE TABLE Fournisseurs (fournisseur_id number(10) CONSTRAINT ck_fournisseurs CHECK (fournisseur_id between 10 and 1000), nom varchar2(20) not null, contact varchar2(50)) ;
- Au niveau table :
  - CREATE TABLE Fournisseurs (fournisseur_id number(10), nom varchar2(20) not null, contact varchar2(50), CONSTRAINT ck_fournisseurs CHECK (fournisseur_id between 10 and 1000)) ;

Les contraintes

La contrainte PRIMARY KEY

- Une contrainte clé primaire crée une clé primaire pour la table
- Une seule clé primaire peut être créée par table
- La clé primaire peut être constituée d'une ou plusieurs colonnes. Aucune des colonnes faisant partie de la clé primaire ne peut être NULL

Exemple

- Au niveau colonne :
  - CREATE TABLE Fournisseurs (fournisseur_id number(10) PRIMARY KEY, nom varchar2(20) not null, contact varchar2(50)) ;
  - CREATE TABLE Fournisseurs (fournisseur_id number(10) CONSTRAINT pk_fournisseurs PRIMARY KEY, nom varchar2(20) not null, contact varchar2(50)) ;
- Au niveau table :
  - CREATE TABLE Fournisseurs (fournisseur_id number(10), nom varchar2(20) not null, contact varchar2(50), CONSTRAINT pk_fournisseurs PRIMARY KEY(fournisseur_id)) ;
  - CREATE TABLE Etudiants(nom varchar2(30), prenom varchar2(30), date_naiss date, CONSTRAINT pk_etudiants PRIMARY KEY (nom,prenom)) ;

Les contraintes

La contrainte FOREIGN KEY

Syntaxe

- Au niveau table :

```
CREATE TABLE <nom_table>
(<col1> type null/not null,
<col2> type null/not null,
...,
CONSTRAINT <fk_table_colonne> FOREIGN KEY (col1, col2, ... coln) REFERENCES
<table_parente> (col1, col2, ... coln)
ON DELETE {CASCADE|SET NULL|SET DEFAULT|RESTRICT}
ON UPDATE {CASCADE|SET NULL|SET DEFAULT|RESTRICT}
);
```

- Au niveau colonne :

```
CREATE TABLE <nom_table>
(<col1> type null/not null CONSTRAINT <fk_table_colonne> REFERENCES <table_parente>
(col) ON DELETE {CASCADE|SET NULL|SET DEFAULT|RESTRICT}
ON UPDATE {CASCADE|SET NULL|SET DEFAULT|RESTRICT},
<col2> type null/not null,
...,
);
```

Les contraintes

La contrainte FOREIGN KEY

- FOREIGN KEY : définit la colonne dans la table fille au niveau contrainte de table
- REFERENCES : identifie la table et la colonne dans la table parente

Notez Bien

- Créer les tables parentes avant les tables filles
- Supprimer les tables filles avant les tables parentes

Les contraintes

La contrainte FOREIGN KEY : mot clé ON DELETE

Les options possibles avec le mot clé ON DELETE :

- CASCADE : supprime les lignes dépendantes dans la table fille lorsqu'une ligne de la table parente est supprimée
- SET NULL : convertit les valeurs des clés étrangères dépendantes en valeurs NULL. Cette contrainte ne peut être exécutée que si toutes les colonnes de clé étrangère de la table cible acceptent des valeurs NULL.
- SET DEFAULT place la valeur par défaut (qui suit ce paramètre) dans la ligne de la table étrangère en cas d'effacement d'une valeur correspondant à la clé (option n'est pas prise en charge dans oracle)
- RESTRICT indique une erreur en cas d'effacement d'une valeur correspondant à la clé (option n'est pas prise en charge dans oracle)

Les contraintes

La contrainte FOREIGN KEY : mot clé ON UPDATE

Les options possibles avec le mot clé ON UPDATE (option n'est pas prise en charge dans oracle) :

- CASCADE : met à jour les lignes dépendantes dans la table fille lorsqu'une ligne de la table parente est mise à jour
- SET NULL : convertit les valeurs des clés étrangères dépendantes en valeurs NULL. Cette contrainte ne peut être exécutée que si toutes les colonnes de clé étrangère de la table cible acceptent des valeurs NULL.
- SET DEFAULT place la valeur par défaut (qui suit ce paramètre) dans la ligne de la table étrangère en cas de mise à jour d'une valeur correspondant à la clé
- RESTRICT indique une erreur en cas de mise à jour d'une valeur correspondant à la clé

Les contraintes

La contrainte FOREIGN KEY

Exemple 1

CREATE TABLE Fournisseurs (fournisseur_id number(10) not null PRIMARY KEY, nom varchar2(50) not null, contact varchar2(50)) ;

CREATE TABLE Produits (produit_id number(10) not null, fournisseur_id number(10) not null, CONSTRAINT fk_produits FOREIGN KEY (fournisseur_id) REFERENCES Fournisseurs (fournisseur_id) ON DELETE cascade) ;

Exemple 2

CREATE TABLE Fournisseurs (fournisseur_id number(10) not null, nom varchar2(50) not null, contact varchar2(50),CONSTRAINT pk_fournisseur PRIMARY KEY (fournisseur_id ,nom)) ;

CREATE TABLE Produits (produit_id number(10) not null, fournisseur_id number(10) not null, nom varchar2(50) not null, CONSTRAINT fk_produits FOREIGN KEY (fournisseur_id , nom) REFERENCES Fournisseurs (fournisseur_id ,nom)) ;

Modification de table

L'instruction ALTER TABLE : permet de modifier la structure de la table

- colonne :
  - ajouter une colonne (ADD)
  - modifier une colonne existante (MODIFY) –propriété d'oracle
  - supprimer une colonne (DROP)
- contrainte :
  - ajouter une contrainte (ADD CONSTRAINT)
  - supprimer une contrainte (DROP CONSTRAINT)

Modification de table

Ajouter une colonne

Syntaxe

ALTER TABLE <nom_table> ADD (<column> type [DEFAULT expr] [, <column> type][DEFAULT expr],...) ;

Exemple

ALTER TABLE Fournisseurs ADD(adresse varchar2(50), telephone number(8) NOT NULL) ;

Modification de table

Modifier une colonne

La modification d'une colonne peut concerner :

- le type de données,
- la taille,
- la valeur par défaut d'une colonne (affecte uniquement les nouvelles insertions dans la table)

Syntaxe

ALTER TABLE <nom_table> MODIFY (<column> type [DEFAULT expr] [, <column> type][DEFAULT expr],...) ;

Exemple

ALTER TABLE Fournisseurs Modify(adresse varchar2(100), telephone number(13)) ;

Modification de table

Supprimer une colonne

Syntaxe

ALTER TABLE <nom_table> DROP (column1, column2, ...) ;

Exemple

ALTER TABLE Fournisseurs DROP (adresse, telephone) ;

Modification de table

Ajouter une contrainte

Syntaxe

ALTER TABLE <nom_table> ADD [CONSTRAINT <nom_contrainte>] type_contrainte (<nom_colonne>) ;

Exemples

- ALTER TABLE Fournisseurs ADD CONSTRAINT uq_fournisseurs UNIQUE(fournisseur_id) ;
- ALTER TABLE Fournisseurs ADD CONSTRAINT ck_fournisseurs CHECK (fournisseur_id between 10 and 1000) ;
- ALTER TABLE Fournisseurs ADD CONSTRAINT pk_fournisseurs PRIMARY KEY (fournisseur_id) ;
- ALTER TABLE Produits add constraint fk_produits FOREIGN KEY (fournisseur_id, nom) REFERENCES Fournisseurs(fournisseur_id, nom) ;

Pour ajouter une contrainte NULL/NOT NULL on utilise l'option MODIFY
Exemple :
ALTER TABLE Fournisseurs MODIFY contact CONSTRAINT nn_fournisseurs_contact NOT NULL ;

Modification de table

Supprimer une contrainte

Syntaxe

ALTER TABLE <nom_table> DROP CONSTRAINT <nom_contrainte> ;

Exemple

- ALTER TABLE Fournisseurs DROP CONSTRAINT ck_fournisseurs ;
- ALTER TABLE Fournisseurs DROP CONSTRAINT pk_fournisseurs

Modification de table

Activer/Désactiver une contrainte

Syntaxe

ALTER TABLE <nom_table> ENABLE | DISABLE CONSTRAINT <nom_contrainte> ;

Exemple

- CREATE TABLE T1(a1 number PRIMARY KEY, b1 varchar2(10)) ;
- CREATE TABLE T2(a2 varchar2(10) PRIMARY KEY, b2 number CONSTRAINT fk_T2 REFERENCES T1(a1)) ;
- ALTER TABLE T1 ADD CONSTRAINT fk_T1 FOREIGN KEY (b1) REFERENCES T2(a2) ;

$\Longrightarrow$ Pour insérer des lignes dans ces deux tables il faut désactiver l'une des contraintes FOREIGN KEY :

- ALTER TABLE T1 DISABLE CONSTRAINT fk_T1 ;

$\Longrightarrow$ Insérer les lignes dans T1 et T2 :

- INSERT INTO T1 VALUES(1,'a') ;
- INSERT INTO T1 VALUES(2,'b') ;
- INSERT INTO T2 VALUES('b',1) ;

$\Longrightarrow$ Activer la contrainte fk_T1 :

- ALTER TABLE T1 ENABLE CONSTRAINT fk_T1 ;

La vérification des valeurs insérées se fait lors de l'activation de la contrainte fk_T1

Suppression de table

Syntaxe

DROP TABLE <nom_table> ;

Remarque

Il existe aussi la commande TRUNCATE qui permet de vider la table.
Syntaxe :
TRUNCATE TABLE nom_table ;

Renommage de table

Syntaxe

RENAME <ancien_nom> TO <nouveau_nom> ;

Exemple

RENAME Fournisseurs TO LesFournisseurs ;

Plan

Vue

- Une vue est une table logique sur une ou plusieurs autres tables ou vues,
- Seule la définition de la vue (requête) est enregistrée dans la base, et pas les données de la vue,
- Limite l'accès à la base de données,
- Facilite la création des requêtes complexes,
- Présente les mêmes données sous différentes formes,
- Il existe deux types de vues :
  - vue simple :
    - utilise une seule table
    - ne contient ni fonction ni groupe de données
    - permet d'exécuter des instructions LMD (UPDATE, DELETE, INSERT)
  - vue complexe :
    - utilise plusieurs tables
    - contient des fonctions ou des groupes de données
    - ne permet pas des instructions LMD (UPDATE, DELETE, INSERT)

Création et modification de vue

Syntaxe

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW <nom_vue> [(alias [, alias], . . .)]
AS SELECT <requête> [WITH CHECK OPTION [CONSTRAINT <nom_contrainte>]]
[WITH READ ONLY [CONSTRAINT <nom_contrainte>]] ;

- FORCE : crée la vue que les tables existent ou non
- alias : indique les noms des expressions sélectionnées par la requête de la vue. Le nombre d'alias doit être égal au nombre d'expressions sélectionnées
- WITH CHECK OPTION : n'autorise l'insertion et la mise à jour des lignes que pour les lignes auxquelles la vue peut accéder (vérifient les conditions de la clause WHERE)
- CONSTRAINT : donne un nom de contrainte aux restrictions WITH CHECK OPTION et WITH READ ONLY (messages d'erreur mieux compréhensibles)
- WITH READ ONLY : aucune opération LMD ne peut être exécutée.

Notez Bien

Pour pouvoir insérer dans une vue, la vue doit comprendre au moins les colonnes sur lesquelles on a une contrainte NOT NULL

Création et modification de vue

Exemple

- CREATE TABLE Fournisseurs (fournisseur_id number(10) PRIMARY KEY, nom varchar2(20) not null, contact varchar2(50), code_region number(3)) ;

1. CREATE OR REPLACE VIEW vue1_Fournisseurs_10 (numero,nom,region) AS SELECT fournisseur_id, nom, code_region from fournisseurs where code_region=10 ;
2. CREATE OR REPLACE VIEW vue2_Fournisseurs_10 (numero,nom,region) AS SELECT fournisseur_id, nom, code_region from fournisseurs where code_region=10 WITH CHECK OPTION CONSTRAINT ck_10 ;
   - INSERT INTO vue2_Fournisseurs_10 VALUES (100, 'Daniel',100) ;

   $\Longrightarrow$ORA-01402 : vue WITH CHECK OPTION - violation de clause WHERE
   - INSERT INTO vue2_Fournisseurs_10 VALUES (100, 'Daniel',10) ;

   $\Longrightarrow$1 ligne(s) insérée(s).
3. CREATE OR REPLACE VIEW vue3_Fournisseurs_10 (numero,nom,region) AS SELECT fournisseur_id, nom, code_region from fournisseurs where code_region=10 WITH READ ONLY ;

Suppression de vue

Syntaxe

DROP VIEW <nom_vue> ;

Exemple

DROP VIEW vue3_Fournisseurs_10 ;

Plan

Séquence

Une séquence :

- Génère automatiquement des numéros uniques
- Est Partageable entre plusieurs utilisateurs et éventuellement entre plusieurs tables
- Permet de créer une valeur de clé primaire

Création de séquence

Syntaxe

CREATE SEQUENCE <nom_sequence>
[INCREMENT BY <pas>]
[START WITH <valeur>]
[{MAXVALUE <valeur_max> | NOMAXVALUE}]
[{MINVALUE <valeur_min> | NOMINVALUE}]
[{CYCLE | NOCYCLE}]
[{CACHE <cache> | NOCACHE}]

- INCREMENT BY : définit l'intervalle entre les numéros (pas d'incrémentation)
- START WITH : premier numéro de la séquence
- MAXVALUE | NOMAXVALUE : valeur maximale
- MINVALUE | NOMINVALUE : valeur minimale
- CYCLE | NOCYCLE : la séquence peut continuer à générer ou non des valeurs si la valeur maximale (ou minimale) est atteinte (la valeur par défaut NOCYCLE)
- CACHE | NOCACHE : nombre de valeurs pré-allouées et conservées en mémoire (la valeur par défaut est cache=20)

Création de séquence

Exemples

- CREATE SEQUENCE sequence1 increment by 1 start with 1 maxvalue 5 ;
- CREATE SEQUENCE sequence2 increment by 5 start with 10 maxvalue 100 NOCACHE NOCYCLE ;
- CREATE SEQUENCE sequence3 increment by 1 start with 1 maxvalue 20 CACHE 10 CYCLE
- CREATE SEQUENCE sequence4 increment by 5 start with -10 minvalue -20 maxvalue 5 cache 2 cycle ;

Notez Bien

La valeur du cache doit etre inférieur ou égale au nombre de valeurs d'un cycle

Modification de séquence

Syntaxe

ALTER SEQUENCE <nom_sequence>
[INCREMENT BY <pas>]
[START WITH <valeur>]
[{MAXVALUE <valeur_max> | NOMAXVALUE}]
[{MINVALUE <valeur_min> | NOMINVALUE}]
[{CYCLE | NOCYCLE}]
[{CACHE <cache> | NOCACHE}]

Exemple

ALTER SEQUENCE sequence1 maxvalue 10 cycle ;

Utilisation de séquence

- L'utilisation d'une séquence se fait par des "pseudo-colonnes" CURRVAL et NEXTVAL. On parle de pseudo-colonne car cela se manipule un peu comme une colonne de table, mais ce n'est pas une colonne de table.
  - CURRVAL : retourne la valeur courante de la séquence
  - NEXTVAL : incrémente la séquence et retourne la nouvelle valeur

Exemple

- CREATE TABLE Tseq(a number, b varchar2(5)) ;
- CREATE SEQUENCE seq increment by 5 start with -10 minvalue -20 maxvalue 5 cache 2 cycle ;
- INSERT INTO Tseq values (seq.nextval, 'a')
- SELECT * FROM Tseq ;

A	B
-10	a
-5	a
0	a
5	a
-20	a

- SELECT seq.currval FROM dual ;

CURRVAL
-20

Suppression de séquence

Syntaxe

DROP SEQUENCE <nom_sequence> ;

Exemple

DROP SEQUENCE seq ;

Plan

Index

- Un index est un objet de la base de données qui permet d'accélérer la recherche des lignes. Il contient deux champs :
  - la clé d'index
  - l'adresse du bloc de données contenant la clé
- Un index peut être créé juste après la création d'une table ou sur une table contenant déjà des lignes
- Un index peut porter sur plusieurs colonnes : la clé d'accès sera la concaténation des différentes colonnes
- Les index sont indépendants logiquement et physiquement des tables qu'ils indexent

Création d'index

- La création d'index peut être :
  - Automatique : un index unique est crée automatiquement lors de la définition d'une contrainte PRIMARY KEY ou contrainte UNIQUE
  - Manuelle : un index non unique peut être crée manuellement (clés d'index peuvent être dupliquées)

Syntaxe

CREATE [UNIQUE] INDEX <nom_index> ON <nom_table>(col1, col2,...)

Exemple

CREATE UNIQUE INDEX idx_fournisseurs ON Fournisseurs (contact) ;
$\Longrightarrow$des valeurs dupliquées ne sont pas permise dans la colonne contact même si il y a pas une contrainte UNIQUE sur cette colonne

Remarque

Il est possible de renommer un index
Syntaxe :
ALTER INDEX <ancien_nom> RENAME TO <nouveau_nom> ;

Création d'index

- Créez index si :
  - La colonne doit être souvent utilisée dans la clause WHERE ou une condition de jointure
  - La colonne contient un grand nombre de valeurs NULL
  - Deux ou plusieurs colonnes sont souvent utilisées conjointement dans une clause WHERE ou une condition de jointure
  - La table est de grande taille et la plupart des requêtes doivent extraire moins de 2 à 4% des lignes
- Ne pas créez index si :
  - La table est de petite taille
  - La table est souvent mise à jour
  - Les colonnes ne sont pas souvent utilisées comme condition dans une requête
  - La plupart des requêtes sont prévues pour extraire un très grand pourcentage de lignes

Supression d'index

Syntaxe

DROP INDEX <nom_index> ;

Exemple

DROP INDEX idx_fournisseurs

Plan

Synonyme

- Un synonyme est un alias sur un Objet de la base ou Schéma, une sorte de raccourcis.
- L'objet peut être une table, une vue, une séquence, une procédure, une fonction, un package, etc
- Le synonyme peut être publique ou privé.
  - publique : accessible à partir de tous schéma et utilisateurs
  - privé il sera accessible uniquement à partir du schéma dans lequel il a été créé
- Créer des synonymes pour :
  - masquer le vrai nom des objets et leur localisations
  - simplifier les noms des objets
  - éviter le pré-fixage dans les requêtes avec le nom de son propriétaire

Création et supression de synonyme

- Création :

Syntaxe

CREATE [OR REPLACE] [PUBLIC] SYNONYM
<nom_synonyme> FOR [schéma.]<nom_objet> ;

- Supression

Syntaxe

DROP SYNONYM <nom_synonyme> ;