Langage de définitions de données

Ilham SLIMANI
slimani.ilham@gmail.com

Création de table

Syntaxe

- Pour pouvoir créer une table dans votre schéma, il faut avoir le privilège CREATE TABLE.
- La syntaxe SQL est la suivante :

CREATE TABLE [*schéma.]nomTable*

(colonne1 type1 [DEFAULT valeur1] [NOT NULL]

[, colonne2 type2 [DEFAULT valeur2] [NOT NULL]]

[CONSTRAINT nomContrainte1 typeContrainte1]…) ;

- La syntaxe simplifiée est:

CREATE TABLE [nomTable colonne1 type1, colonne2 type2 …);

3

Syntaxe

- *nomTable : peut comporter des lettres majuscules ou minuscules (accentuées ou pas),* des chiffres et les symboles, par exemple : _, $ et #.
- *colonnei typei : nom d'une colonne (mêmes caractéristiques que pour les noms des* tables) et son type (NUMBER, CHAR, DATE…).

4

Syntaxe

- La directive DEFAULT fixe une valeur par défaut.
- La directive NOT NULL interdit que la valeur de la colonne soit nulle.
- *nomContraintei typeContraintei : noms de la contrainte et son type (clé primaire, clé étrangère, etc.).*

5

Conventions

- il est préférable d'utiliser les conventions suivantes :
  - tous les mots-clés de SQL sont notés en MAJUSCULES ;
  - les noms de tables sont notés en Minuscules (excepté la première lettre) ;
  - les noms de colonnes et de contraintes en minuscules.

6

Commentaire

CREATE TABLE MêmesévénementsàNoël (colonne CHAR); CREATE TABLE Test (colonne NUMBER(38,8)); CREATE table **test (Colonne** NUMBER(38,8));	CREATE TABLE **-- nom de la table** **TEST(** **-- description** COLONNE NUMBER(38,8)) **-- fin, ne pas oublier le point-virgule.** **;** CREATE TABLE Test (/* **une plus grande description** des colonnes */ COLONNE NUMBER(38,8));
Sans commentaire	**Avec commentaire**

7

Exemple

Compagnie

comp	nrue	rue	ville	nomComp

- La table contient cinq colonnes :
  - quatre chaînes de caractères
  - et une valeur numérique de trois chiffres.
- La table inclut en plus deux contraintes :
  - DEFAULT qui fixe *Paris comme valeur par défaut de* la colonne ville ;
  - NOT NULL qui impose une valeur non nulle dans la colonne nomComp

8

Exemple

- CREATE TABLE soutou.Compagnie

(comp CHAR(4),
nrue NUMBER(3),
rue CHAR(20),
ville CHAR(15) **DEFAULT 'Paris',**
nomComp CHAR(15) **NOT NULL);**

9

Contraintes

- Les contraintes ont pour but de programmer des règles de gestion au niveau des colonnes des tables.
- Elles peuvent alléger le développement
  - si on déclare qu'une note doit être comprise entre 0 et 20, les programmes de saisie n'ont plus à tester les valeurs en entrée mais seulement le code retour après connexion à la base ; on déporte les contraintes côté serveur)

10

Contraintes

- Les contraintes peuvent être déclarées de deux manières :
- En même temps que la colonne (valable pour les contraintes monocolonnes), ces contraintes sont dites « en ligne » (*inline constraints). L'exemple précédent en déclare deux.*
- Une fois la colonne déclarée, ces contraintes ne sont pas limitées à une colonne et peuvent être personnalisées par un nom (*out-of-line constraints).*

11

Contraintes

- Quatre types de contraintes sont possibles :
- **CONSTRAINT *nomContrainte***
  - UNIQUE (*colonne1 [,colonne2]…)*
  - PRIMARY KEY (*colonne1 [,colonne2]…)*
  - FOREIGN KEY (colonne1 [,*colonne2]…)* REFERENCES [*schéma.]nomTablePere (colonne1 [,colonne2]…)* [ON DELETE { CASCADE | SET NULL }]
  - CHECK (*condition)*

12

UNIQUE

- La contrainte UNIQUE impose une valeur distincte au niveau de la table.

Exemple: Deux artistes ne peuvent avoir les mêmes nom et prénom.

```
CREATE TABLE Cinema
(nom VARCHAR (30)
NOT NULL,
adresse VARCHAR(50)
UNIQUE,
PRIMARY KEY (nom)
)
```

```
CREATE TABLE Artiste
(id INTEGER,
 nom VARCHAR (30),
prenom VARCHAR (30)
NOT NULL,
 anneeNaiss INTEGER,
UNIQUE (nom,
prenom))
```

```
CREATE TABLE Artiste
(id INTEGER,
 nom VARCHAR (30),
prenom VARCHAR (30)
NOT NULL,
 anneeNaiss INTEGER,
CONSTRAINT arti_un
UNIQUE (nom,
prenom))
```

13

PRIMARY KEY

- La contrainte PRIMARY KEY déclare la clé primaire de la table. Sur la ou les colonnes concernées. Les colonnes clés primaires ne peuvent être ni nulles ni identiques
- Exemple: (trois façons de création d'une contrainte)

```
CREATE TABLE clients
( Ncli char(15),
Nom char(30) NOT NULL,
 Prenom char(30) NOT
NULL,
Age integer,
PRIMARY KEY (Ncli)
)
```

```
CREATE TABLE clients
( Ncli char(15) PRIMARY
KEY,
Nom char(30) NOT NULL,
 Prenom char(30) NOT
NULL,
Age integer,
)
```

```
CREATE TABLE clients
( Ncli char(15),
Nom char(30) NOT NULL,
 Prenom char(30) NOT
NULL,
Age integer,
CONSTRAINT cli_pk
PRIMARY KEY(ncli)
)
```

14

FOREIGN KEY

- La contrainte FOREIGN KEY déclare une clé étrangère entre une table enfant (*child) et une* table père (*parent).*
- *Exemple:*

```
CREATE TABLE compte
( ncp char(15) ,
ncli char(15) NOT NULL,
 datc DATE,
PRIMARY KEY (ncp)
FOREIGN KEY (ncli)
REFERENCES
clients(ncli)
)
```

```
CREATE TABLE compte
( ncp char(15) NOT
NULL,
ncli char(15)
REFERENCES
clients(ncli),
 datc DATE,
PRIMARY KEY (ncp)
)
```

```
CREATE TABLE compte
( ncp char(15) NOT
NULL,
ncli char(15) ,
datc DATE,
PRIMARY KEY (ncp),
CONSTRAINT
compte_fk FOREIGN
KEY(ncli) REFERENCES
clients(ncli))
```

15

FOREIGN KEY

- *ON DELETE* est suivi d'arguments entre accolades permettant de spécifier l'action à réaliser en cas d'effacement d'une ligne de la table faisant partie de la clé étrangère :
  - *CASCADE* indique la suppression en cascade des lignes de la table étrangère dont les clés étrangères correspondent aux clés primaires des lignes effacées
  - *RESTRICT* indique une erreur en cas d'effacement d'une valeur correspondant à la clé
  - *SET NULL* place la valeur NULL dans la ligne de la table étrangère en cas d'effacement d'une valeur correspondant à la clé
  - *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

16

FOREIGN KEY

- *ON UPDATE* est suivi d'arguments entre accolades permettant de spécifier l'action à réaliser en cas de modification d'une ligne de la table faisant partie de la clé étrangère :
  - *CASCADE* indique la modification en cascade des lignes de la table étrangère dont les clés primaires correspondent aux clés étrangères des lignes modifiées
  - *RESTRICT* indique une erreur en cas de modification d'une valeur correspondant à la clé
  - *SET NULL* place la valeur NULL dans la ligne de la table étrangère en cas de modification d'une valeur correspondant à la clé
  - *SET DEFAULT* place la valeur par défaut (qui suit ce paramètre) dans la ligne de la table étrangère en cas de modification d'une valeur correspondant à la clé

17

FOREIGN KEY

- Exemple:

```
CREATE TABLE compte
( ncp char(15) NOT NULL,
ncli char(15) ,
datc DATE,
PRIMARY KEY (ncp),
CONSTRAINT compte_fk FOREIGN KEY(ncli)
   REFERENCES clients(ncli) ON UPDATE CASCADE
   ON DELETE RESTRICT ) )
```

18

CHECK

- La contrainte CHECK impose un domaine de valeurs ou une condition simple ou complexe entre colonnes (exemple : CHECK (note BETWEEN 0 AND 20), CHECK (grade='Copilote' OR grade='Commandant')).
- Exemple:
  - CREATE TABLE clients(

```
Nom char(30) NOT NULL,
 Prenom char(30) NOT NULL,
 Age integer,
 check (age < 100),
 Email char(50) NOT NULL,
 check (Email LIKE "%@%") )
```

19

Évolution d'un schéma

Évolution d'un schéma

- Il est possible de modifier une base de données d'un point de vue structurel (colonnes et index) mais aussi comportemental (contraintes).
- L'instruction principalement utilisée est **ALTER TABLE** (commande du LDD) qui permet
  - d'ajouter, de renommer, de modifier et de supprimer des colonnes d'une table.
  - Elle permet d'ajouter, de supprimer, d'activer, de désactiver et de différer des contraintes.

21

Renommer une table

- L'instruction RENAME renomme une table.

- **Syntaxe :**
  - RENAME *ancienNom TO nouveauNom;*
  - ALTER TABLE *ancienNom* RENAME TO *nouveauNom*;
- **Exemple:**
  - RENAME Pilote TO Naviguant;
  - ALTER TABLE Pilote RENAME TO Naviguant;

22

Modifications structurelles

- Considérons la table suivante que nous allons faire évoluer:
  - CREATE TABLE Pilote
  (brevet VARCHAR(4),
  Nom VARCHAR(20));

Ajout de colonnes

- La directive ADD de l'instruction ALTER TABLE permet d'ajouter une nouvelle colonne à une table. Cette colonne est initialisée à NULL pour tous les enregistrements (à moins de spécifier une contrainte DEFAULT, auquel cas tous les enregistrements de la table sont mis à jour avec une valeur non nulle).

Ajout de colonnes (Exemple)

- Le script suivant ajoute trois colonnes à la table Pilote. La première instruction insère la colonne nbHVol en l'initialisant à NULL pour tous les pilotes.
  - **ALTER TABLE Pilote ADD (nbHVol NUMBER(7,2));**
- La deuxième commande ajoute deux colonnes initialisées à une valeur non nulle. La colonne ville ne sera jamais nulle.
  - **ALTER TABLE Pilote ADD (compa VARCHAR(4) DEFAULT 'AF', ville VARCHAR(30) DEFAULT 'Paris' NOT NULL);**

25

Renommer des colonnes

- Il faut utiliser la directive RENAME COLUMN de l'instruction ALTER TABLE pour renommer une colonne existante.

N.B: Le nom de la nouvelle colonne ne doit pas être déjà utilisé par une colonne de la table.

- L'instruction suivante permet de renommer la colonne ville en adresse :
  - **ALTER TABLE Pilote RENAME COLUMN ville TO adresse;**

26

Modifier le type des colonnes

- La directive MODIFY de l'instruction ALTER TABLE modifie le type d'une colonne existante.

Instructions SQL	Commentaires
ALTER TABLE Pilote MODIFY compa VARCHAR(6) DEFAULT 'SING';	Augmente la taille de la colonne compa et change la contrainte de valeur par défaut
ALTER TABLE Pilote MODIFY compa CHAR(4) NOT NULL;	Diminue la colonne et modifie également son type de VARCHAR en CHAR tout en le déclarant NOT NULL
ALTER TABLE Pilote MODIFY compa NULL;	Rend possible l'insertion de valeur nulle dans la colonne compa.

27

Supprimer des colonnes

- La directive DROP COLUMN de l'instruction ALTER TABLE permet de supprimer une colonne.
- Il n'est pas possible de supprimer avec cette instruction:
  - des clés primaires (ou candidates par UNIQUE) référencées par des clés étrangères ;
  - des colonnes à partir desquelles un index a été construit
  - toutes les colonnes d'une table.
- La suppression de la colonne adresse de la table Pilote est programmée par l'instruction suivante :
  - **ALTER TABLE Pilote DROP COLUMN adresse;**

28

Ajout de contraintes

- Il est possible de créer des tables seules, puis d'ajouter les contraintes.
- La directive ADD CONSTRAINT de l'instruction ALTER TABLE permet d'ajouter une contrainte à une table.
- La syntaxe générale est la suivante :
  - ALTER TABLE [*schéma.]nomTable*
    ADD [CONSTRAINT *nomContrainte] typeContrainte;*

29

Ajout des contraintes(exemple)

- Ajoutons la clé étrangère à la table Avion au niveau de la colonne proprio en lui assignant une contrainte NOT NULL :
  - ALTER TABLE Avion

ADD (CONSTRAINT nn_proprio CHECK (proprio IS NOT NULL),

CONSTRAINT fk_Avion_comp_Compag FOREIGN KEY(proprio) REFERENCES Compagnie(comp));

30

Suppression de contraintes

- La directive DROP CONSTRAINT de l'instruction ALTER TABLE permet d'enlever une contrainte d'une table.
- La syntaxe générale est la suivante :

ALTER TABLE [*schéma.]nomTable* ***DROP CONSTRAINT nomContrainte [CASCADE];***

avec la directive CASCADE supprime les contraintes référentielles des tables « pères ».

31

Suppression de contraintes (exemple)

- Supprimons la contrainte NOT NULL qui porte sur la colonne proprio de la table Avion :
  - ALTER TABLE Avion **DROP CONSTRAINT nn_proprio;**
- Supprimons la clé primaire de la table Avion. Il faut préciser CASCADE, car cette table est référencée par une clé étrangère dans une autre table. Cette commande supprime à la fois la clé primaire de la table Avion mais aussi les contraintes clés étrangères des tables dépendantes
  - ALTER TABLE Avion **DROP CONSTRAINT pk_Avion CASCADE;**

32

Suppression d'une table

- Il est possible de supprimer une table grâce à la clause *DROP*
- La syntaxe :
  - **DROP TABLE "nom de table"**
- Ainsi, pour supprimer la table « Customer » créée dans la section **CREATE TABLE**, il suffit de saisir :
  - **DROP TABLE customer**.

33

Les Vues en SQL

- Définition

Une vue est une table virtuelle, c'est-à-dire dont les données ne sont pas stockées dans une table de la base de données, et dans laquelle il est possible de rassembler des informations provenant de plusieurs tables. On parle de "vue" car il s'agit simplement d'une représentation des données dans le but d'une exploitation visuelle. Les données présentes dans une vue sont définies grâce à une clause *SELECT*

34

Création d'une vue en SQL

- La création d'une vue se fait grâce à la clause *CREATE VIEW* suivie du nom que l'on donne à la vue, puis du nom des colonnes dont on désire agrémenter cette vue, puis enfin d'une clause *AS* précédant la sélection. La syntaxe d'une vue ressemble donc à ceci :

CREATE VIEW Nom_de_la_Vue (Colonnes) AS SELECT ...

- Exemple:

```
CREATE VIEW Vue (colonneA,colonneB,colonneC,colonneD)
AS SELECT colonne1, colonne2, colonneI, colonneII
FROM Nom_table1 Alias1,Nom_tableII AliasII
WHERE Alias1.colonne1 = AliasII.colonneI
AND Alias1.colonne2 = AliasII.colonneII
```

35

Suppression d'une vue

- Toute vue peut être supprimée.
- Exemple : **DROP VIEW** nom_vue

Intérêts des vues : La vue représente de cette façon une sorte d'intermédiaire entre la base de données et l'utilisateur. Cela a de nombreuses conséquences :

- une sélection des données à afficher
- une restriction d'accès à la table pour l'utilisateur, c'est-à-dire une sécurité des données accrue
- un regroupement d'informations au sein d'une entité

36

L'index : définition et utilité

- Un index est un objet complémentaire (mais non indispensable) à la base de données permettant d'"indexer" certaines colonnes dans le but d'améliorer l'accès aux données par le SGBDR
- Toutefois la création d'index utilise de l'espace mémoire dans la base de données, et, étant donné qu'il est mis à jour à chaque modification de la table à laquelle il est rattaché, peut alourdir le temps de traitement du SGBDR lors de la saisie de données. Par conséquent il faut que la création d'index soit justifiée et que les colonnes sur lesquelles il porte soient judicieusement choisies (de telle façon à minimiser les doublons).
- De cette façon certains SGBDR créent automatiquement un index lorsqu'une clé primaire est définie.

37

Création d'index en SQL

- La création d'index en SQL se fait grâce à la clause *INDEX* précédée de la clause *CREATE*. Elle permet de définir un index désigné par son nom, portant sur certains champs d'une table. La syntaxe est la suivante :

CREATE [UNIQUE] INDEX Nom_de_l_index

ON Nom_de_la_table

(Nom_de_champ [ASC/DESC], ...)

- L'option UNIQUE permet de définir la présence ou non de doublons pour les valeurs de la colonne
- Les options ASC/DESC permettent de définir un ordre de classement des valeurs présentes dans la colonne

38

Suppression d'un index

- La suppression d'un index se fait logiquement avec DROP et la syntaxe suivante :

DROP INDEX Nom_index

39