Bases de données

Cours 4 – Langage SQL – Langage de Description des Données

Adrien Ugon

ESIEE-Paris

Vendredi 15 Février 2019

Plan

Deux types de langages

Langage de Définition de Données

Le **Langage de Définition de Données** permet de décrire la structure de la base de données.

Langage de Manipulation de Données

Le **Langage de Manipulation de Données** permet de gérer les contenus de la base de données.

Langage de Définition de Données

Le **Langage de Définition de Données** permet de :

- *créer* ou *supprimer* une base de données.
- *définir* un nouveau type de données (domaine).
- *créer* une liste d'utilisateurs.
- *gérer les droits* de ces utilisateurs
- *créer*, *modifier* ou *supprimer* une table.
- *ajouter*, *modifier* ou *supprimer* des contraintes sur une table.

Langage de Manipulation de Données

Le **Langage de Manipulation de Données** permet de :

- *ajouter*, *modifier* ou *supprimer* des données.
- écrire des requêtes pour afficher des données
  - *sélectionner* des données sur une table (requête simple)
  - *filtrer* des données
  - *sélectionner* les attributs
  - *joindre* plusieurs tables
  - *regrouper* des données
  - *calculer* des nouvelles données à partir des données existantes

Plan

Création d'une base de données

La requête permettant de **créer** une nouvelle base de données est la suivante :

CREATE DATABASE *nom_base_de_donnees* ;

Sélectionner une base de données

La requête permettant de **sélectionner** une base de données pour la rendre active dans la session courante est la suivante :

USE *nom_base_de_donnees* ;

Supprimer une base de données

La requête permettant de **supprimer** une base de données :

DROP *nom_base_de_donnees* ;

Plan

Utilisateurs et SGBD

- Tout SGBD gère les accès des utilisateurs aux bases de données.
- Ces accès sont contrôlés :
  - Les utilisateurs sont authentifiés.
  - Chaque utilisateur n'a accès qu'à certaines bases de données.
  - Les opérations autorisées sont totalement décrites.

Création d'un nouvel utilisateur

La requête permettant de créer un nouvel utilisateur est la suivante :

```
CREATE USER [IF NOT EXISTS] user@serveur
IDENTIFIED BY 'password' ;
```

Gestion des privilèges

Privilège	Droit
SELECT	affichage
DELETE	suppression
INSERT	ajout
UPDATE	modification
REFERENCES	faire référence à une autre table
USAGE	utiliser un nom de domaine
EXECUTE	exécuter une procédure

Octroi de privilèges

La requête qui permet d'octroyer des privilèges à une liste d'utilisateurs est la suivante :

```
GRANT liste_privileges
ON [TABLE] objet
TO liste_utilisateurs
[WITH GRANT OPTION];
```

N.B. L'option « WITH GRANT OPTION » permet d'autoriser les utilisateurs à octroyer les droits à leur tour. Par défaut, cela leur est interdit.

Révoquer des droits

La requête qui permet de révoquer des privilèges à une liste d'utilisateurs est la suivante :

```
REVOKE [GRANT OPTION FOR] liste_privileges
ON objet
FROM liste_utilisateurs
[RESTRICT|CASCADE];
```

- Les options RESTRICT et CASCADE ne s'appliquent que lors de la redistribution des privilèges (GRANT OPTION FOR)
- RESTRICT rejette l'instruction si les droits ont été redistribués.
- CASCADE étend l'annulation à toutes les redistributions.

Plan

Types de données en SQL

- Chaque attribut (= « colonne ») est défini avec un *type*.
- Ce type apporte des contraintes :
  - il contraint l'ensemble des valeurs autorisées pour une colonne.
  - il définit les opérations permises.
  - il définit la taille en mémoire pour le stockage des valeurs.
  - il définit le format par défaut pour l'affichage des valeurs

Plan

Quatre types natifs de données

SQL définit 4 « familles » de types de données :

- Types numériques ;
- Chaînes de caractères ;
- Données temporelles ;
- Booléens ;
- Chaînes de bits ;
- Énumérations et Ensembles ;
- Objets « larges ».

Types numériques

Types exacts et approchés

Les types numériques sont divisés en deux catégories :

- les types **exacts** ;
- les types **approchés**.

Types numériques pour les nombres réels

Type	Description	Taille	Minimum	Maximum
INTEGER ou INT	entier relatif	4 octets	-2^{31}	$2^{31}-1$
UNSIGNED INTEGER ou UNSIGNED INT	entier	4 octets	0	$2^{32}-1$
TINYINT	entier relatif	1 octet	-2^{7}	$2^{7}-1$
UNSIGNED TINYINT	entier	1 octet	0	$2^{8}-1$
SMALLINT	entier relatif	2 octets	-2^{15}	$2^{15}-1$
UNSIGNED SMALLINT	entier	2 octets	0	$2^{16}-1$
MEDIUMINT	entier relatif	3 octets	-2^{23}	$2^{23}-1$
UNSIGNED MEDIUMINT	entier	3 octets	0	$2^{24}-1$
BIGINT	entier relatif	8 octets	-2^{63}	$2^{63}-1$
UNSIGNED BIGINT	entier	8 octets	0	$2^{64}-1$

Types numériques exacts pour les décimaux

On note avec les types suivants un nombre décimal composé de p chiffres dont e situés après la virgule :

- NUMERIC(p,e)
- DECIMAL(p,e) ou DEC(p,e)
- FLOAT(p,e)
- REAL(p,e)

Types numériques approchés pour les décimaux

Type	Description	Taille
FLOAT	Flottant simple précision	4 octets
REAL	Flottant simple précision	4 octets
DOUBLE ou DOUBLE PRECISION	Flottant double précision	8 octets

Différents types de chaînes de caractères

Une chaîne de caractères est définie comme une suite quelconque de caractères, chacun pouvant être imprimable ou non (espace, tabulation, ...).

SQL différencie :

- chaînes de longueur fixe et chaînes de longueur variable ;
- encodage des caractères sur 1 octet (ASCII et EBCDIC) ou sur 2 octets (UNICODE).

Types de chaînes de caractères en SQL

Type	Description	Encodage	Nombre de caractères
CHARACTER(N) ou CHAR(N)	Chaîne de longueur fixe	1 octet	N
CHARACTER VARYING(N) ou VARCHAR(N)	Chaînes de longueur variable	1 octet	$L+1$ avec $L \leq N$
NATIONAL CHARACTER(N) ou NATIONAL CHAR(N)	Chaîne de longueur fixe	2 octets	N
NATIONAL CHARACTER VARYING(N) ou NATIONAL VARCHAR(N)	Chaînes de longueur variable	2 octets	$L+1$ avec$L \leq N$

Types de données temporelles

Il existe plusieurs types de données temporelles :

- les dates ;
- les durées.

Dates et Heures

Type	Description	Taille	format
DATE	Date(année, mois, jour)	3 octets	'YYYY-MM-DD'
TIME	Horaire(heure, minutes, secondes)	3 octets	'HH :MM :SS'
DATETIME	Date et Horaire	8 octets	'YYYY-MM-DD HH :MM :SS'
TIMESTAMP	Date et Horaire	4 octets	'YYYY-MM-DD HH :MM :SS'
YEAR	Année	1 octet	'YYYY'

Note

Le type TIMESTAMP a une plage de valeurs plus réduite que DATETIME.

- TIMESTAMP autorise des valeurs de '1970-01-01 00 :00 :01' UTC à '2038-01-19 03 :14 :07'
- DATETIME autorise des valeurs de '1000-01-01 00 :00 :00' à '9999-12-31 23 :59 :59'.

Durées

Pour les durées, on utilise le type INTERVAL qui est défini par le format suivant :

INTERVAL *unite_1* [TO *unite_2*]

- *unite_1* représente une unité de temps (YEAR, MONTH, DAY, HOUR, MINUTE, SECOND)
- *unite_2* représente une unité de temps inférieure à *unite_1*. Elle permet d'exprimer des durées sur plusieurs unités, y compris les unités intermédiaires (par exemple, en heures, minutes, secondes).

Le type booléen

- Le type booléen est défini par BOOLEAN, ou BOOL.
- Il est synonyme de TINYINT(1). 0 signifie FAUX, les autres valeurs signifient VRAI.

Les types de chaînes de bits

De la même façon que pour les chaînes de caractères, il existe deux types de chaînes de bits :

- les chaînes de **longueur fixe**.
- les chaînes de **longueur variable**.

Chaînes de bits de longueur fixe

Les chaînes de bits de longueur fixe sont définis par le type :
BIT(n)

où n définit la longueur de la chaîne.

Chaînes de bits de longueur variable

Les chaînes de bits de longueur variable sont définis par le type :
BIT VARYING(n)

où n définit la longueur maximale de la chaîne.

Utilisation des énumérations et ensembles

- Parfois, les valeurs autorisées pour un attribut appartiennent à une liste de valeurs prédéfinie.
- Dans ce cas, deux cas se distinguent :
  - L'attribut n'autorise qu'une seule valeur. On utilisera alors une énumération.
  - L'attribut autorise plusieurs valeurs. On utilisera alors un ensemble.

Le type « Énumérations »

- Les énumérations sont définies par le type ENUM(*valeur_1*,*valeur_2*, . . . , *valeur_n*)
- Les valeurs autorisées sont indexées, ce qui permet de gagner de la place en stockage mémoire.
- Les valeurs suivantes sont également autorisées :
  - « chaîne vide » : '' ;
  - NULL.

Le type « Ensembles »

- Les ensembles sont définis par le type SET(*valeur_1*,*valeur_2*, . . . , *valeur_n*)
- Les valeurs autorisées sont indexées, ce qui permet de gagner de la place en stockage mémoire.
- Les valeurs suivantes sont également autorisées :
  - « chaîne vide » '' ;
  - NULL.

Réflexion sur les types Énumérations et Ensembles

Commentaire

Il est toujours possible de créer des tables « *ancillaires* » pour les attributs de type Énumérations et Ensembles.
On préférera autant que possible profiter du modèle relationnel, plutôt que d'utiliser les types ENUM et SET.

Le type BLOB

- BLOB : Binary Large OJect
- Le type BLOB est dédié aux attributs binaires longs binaires.
- Il existe des dérivés :
  - TINYBLOB ;
  - MEDIUMBLOB ;
  - LONGBLOB ;

Type BLOB

Type	Taille	Maximum
TINYBLOB	L + 1 octets, avec $L < 2^8 = 256$	256 octets
BLOB	L + 2 octets, avec $L < 2^16 = 65536$	64 ko
MEDIUMBLOB	L + 3 octets, avec $L < 2^24 = 16777216$	16 Mo
LONGBLOB	L + 4 octets, avec $L < 2^32 = 4294967296$	4 Go

Le type TEXT

- Le type TEXT est dédié aux attributs de type texte long (par exemple, un roman).
- Parfois le type TEXT s'appelle CLOB (Character Large OBject).
- Il existe des dérivés :
  - TINYTEXT ;
  - MEDIUMTEXT ;
  - LONGTEXT ;

Type TEXT

Type	Taille	Maximum
TINYTEXT	L + 1 octets, avec $L < 2^8 = 256$	256 octets
TEXT	L + 2 octets, avec $L < 2^16 = 65536$	64 ko
MEDIUMTEXT	L + 3 octets, avec $L < 2^24 = 16777216$	16 Mo
LONGTEXT	L + 4 octets, avec $L < 2^32 = 4294967296$	4 Go

Plan

Intérêt de la création de domaines

- Un **domaine** est un type de données définissant un sous-ensemble de données, à partir d'un type prédéfini.
- Le nom de ce domaine peut être plus significatif dans le contexte d'utilisation.
- Il est possible de
  - spécifier la valeur par défaut
  - ajouter des contraintes

Création d'un domaine

La requête qui permet de définir un domaine est la suivante :

```
CREATE DOMAIN nom_domaine
AS type_predefini
[DEFAULT valeur_par_defaut]
[[CONSTRAINT nom_contrainte]CHECK contrainte]
```

Exemple

```
CREATE DOMAIN pression_arterielle
AS UNSIGNED TINYINT
DEFAULT 110
CHECK (VALUE BETWEEN 40 AND 220);
```

Suppression d'un domaine

La requête qui permet de supprimer un domaine est la suivante :

```
DROP DOMAIN nom_domaine
[RESTRICT|CASCADE];
```

- RESTRICT : L'opération est refusée si le domaine est utilisé dans la base de données ; c'est l'option par défaut.
- CASCADE : L'opération est toujours acceptée et les colonnes affectées passent au type prédéfini du domaine.

Plan

Plan

Requête pour créer une table

Une requête SQL de création de table contient les informations suivantes :

- la liste des attributs ;
- les contraintes sur les attributs ;
- les contraintes sur la table.

Requête SQL pour créer une table

La requête SQL qui permet de créer une table est la suivante :

```
CREATE TABLE nom_table
(nom_attribut_1 type_attribut_1 [NOT NULL] [DEFAULT valeur_defaut_attribut_1] [PRIMARY KEY],
nom_attribut_2 type_attribut_2 [NOT NULL] [DEFAULT valeur_defaut_attribut_1] [PRIMARY KEY],
...,
nom_attribut_n type_attribut_n [NOT NULL] [DEFAULT valeur_defaut_attribut_1] [PRIMARY KEY],
[contraintes_table]
);
```

Plan

Contrainte sur un attribut

Pour chaque attribut, on définit les contraintes suivantes :

- le type de l'attribut, son domaine de valeurs.
- l'attribut peut-il être nul ou non ?
- la valeur par défaut de l'attribut
- l'attribut est-il clé primaire ?

Contrainte sur un attribut

Type et domaine

- Le type spécifie la nature des valeurs d'un attribut (*numérique, textuel, date, booléen*).
- Il détermine l'ensemble des valeurs autorisées pour cet attribut (*minimum, maximum, longueur, . . .*).
- Il détermine la taille en mémoire.

Contrainte sur un attribut

La mention NOT NULL

- La mention NOT NULL permet de spécifier que la valeur de l'attribut ne peut pas avoir la valeur NULL.
- La valeur NULL est utilisée :
  - lorsque la donnée est inconnue (*p.ex. valeur non mesurée*).
  - lorsque la donnée est inexistante (*p.ex. ancienneté d'un interne en médecine*).
  - lorsque la donnée est absurde du point de vue sémantique (*p.ex. ménopause chez une homme*)

Contrainte sur un attribut

Valeur par défaut

- Il est possible de définir une valeur par défaut pour un attribut.
- Cette valeur sera utilisée lors de l'insertion d'un nouveau tuple, dont la valeur de cet attribut ne serait pas mentionnée.
- Par exemple, la date courante pour un champ date.

Contrainte sur un attribut

Clé primaire

- Si l'attribut appartient à la clé primaire, alors on peut le mentionner dans la définition de cet attribut.
- La clé primaire est l'ensemble des champs identifiant d'un tuple.
- Un attribut de la clé primaire ne peut pas être NULL.

Contrainte sur un attribut

Format général

Le format général de la ligne permettant d'ajouter des contraintes à un attribut est :

nom_attribut_1 type_attribut_1 [NOT NULL] [DEFAULT *valeur_defaut_attribut_1*] [PRIMARY KEY],

Contraintes sur la table

Ajouter une contrainte

Il est possible de définir des contraintes sur la table :

- contrainte d'entité (clé primaire) ;
- contrainte de clé candidate (unicité) ;
- contrainte d'intégrité référentielle ;
- contrainte sur les valeurs d'un attribut.

Contraintes sur la table

Format général pour exprimer une contrainte

en SQL, les contraintes s'expriment de la façon suivante :

[CONSTRAINT *nom_de_la_contrainte*]
MOT_CLE_CONTRAINTE options

Contraintes sur la table

Contrainte d'entité (clé primaire)

Il est possible (et recommandé) de définir la clé primaire dans les contraintes portant sur la table.

La contrainte s'exprime alors de la façon suivante :

[CONSTRAINT *nom_de_la_contrainte*]
PRIMARY KEY(*attribut_1*,*attribut_2*,...,*attribut_n*)

Contraintes sur la table

Clés candidates (unicité)

Les clés candidates se définissent par une contrainte d'unicité.

Elle s'exprime de la façon suivante :

[CONSTRAINT *nom_de_la_contrainte*]
UNIQUE(*attribut_1*,*attribut_2*,...,*attribut_n*)

Contraintes sur la table

Contrainte d'intégrité référentielle

Chaque clé étrangère se définit par une contrainte d'intégrité référentielle.

Elle s'exprime de la façon suivante :

[CONSTRAINT *nom_de_la_contrainte*]
FOREIGN KEY(*a_1*,*a_2*,...,*a_n*) REFERENCES *nom_table*(*a_1'*,*a_2'*,...,*a_n'*)
[MATCH {PARTIAL|FULL}]
[ON DELETE *action*]
[ON UPDATE *action*]

Contraintes sur la table

Contrainte d'intégrité référentielle

La clause MATCH permet de gérer le comportement face à l'insertion de valeurs NULL dans une clé étrangère :

- **FULL** relâche les contrôles uniquement si toutes les colonnes sont à NULL.
- **PARTIAL** relâche les contrôles pour les colonnes à NULL.

Contraintes sur la table

Contrainte d'intégrité référentielle

Les clauses ON DELETE et ON UPDATE permettent de définir le comportement lorsqu'on supprime ou modifie la clé candidate à laquelle on fait référence.

Les valeurs autorisées sont les suivantes :

NO ACTION La suppression ou la modification d'une valeur de la clé candidate est refusée s'il existe des valeurs identiques dans la clé étrangère.

CASCADE L'opération est autorisée et les lignes possédant des valeurs identiques aux valeurs de la clé candidate modifiée/supprimée sont elles aussi modifiées/supprimées.

SET NULL L'opération est autorisée et les valeurs des attributs identiques aux valeurs de la clé candidate modifiée/supprimée sont modifiées et mises à la valeur NULL.

SET DEFAULT L'opération est autorisée et les valeurs des attributs identiques aux valeurs de la clé candidate modifiée/supprimée sont modifiées et mises à la valeur par défaut.

Contraintes sur la table

Contrainte sur les valeurs d'un attribut

Il est possible de restreindre l'ensemble de valeurs d'un attribut.

La contrainte s'exprime de la façon suivante :

[CONSTRAINT *nom_de_la_contrainte*]
CHECK (*expression_logique*)

Contraintes sur la table

Exemples de contraintes sur les valeurs d'un attribut

Exemple sur des valeurs numériques

```
CHECK (pression_arterielle_diastolique BETWEEN 40 AND 150)
```

Exemple sur des valeurs textuelles

```
CHECK (specialite IN ('cardiologie','pneumologie','MIT','psychiatrie','ORL','obstétrique','urgences'))
```

Exemple sur des valeurs textuelles

```
CHECK (glycemie>0.3 AND glycemie<2.5)
```

Requête SQL pour créer une table

Exemple n°1

Création d'une table Medecin

```
CREATE TABLE Medecin(
num_RPPS UNSIGNED INT,
nom VARCHAR(100),
prenom VARCHAR(100),
date_naissance DATE,
sexe ENUM('H','F'),
adresse VARCHAR(255),
id_specialite UNSIGNED TINYINT,
PRIMARY KEY(IPP),
FOREIGN KEY (specialite) REFERENCES specialite(id_specialite),
CHECK nom in ('Dupont','Durand')
);
```

Requête SQL pour créer une table

Exemple n°2

Création d'une table Hopital

```
CREATE TABLE Hopital(
FINESS UNSIGNED INT, nom VARCHAR(100), id_directeur UNSIGNED
TINYINT, adresse VARCHAR(100), code_insee CHAR(5), PRIMARY
KEY(FINESS), FOREIGN KEY(id_directeur) REFERENCES
Personnel(id_personnel), FOREIGN KEY(code_insee) REFERENCES
Villes(code_insee) ) ;
```

Requête SQL pour créer une table

Exmple n°3

Création d'une table Consultation

```
CREATE TABLE consultation(
RPPS UNSIGNED SMALLINT,
IPP UNSIGNED INT,
date Date,
lieu VARCHAR(100),
compte_rendu TEXT
PRIMARY KEY()
);
```

Requête SQL pour créer une table

Exmple n°3

Création d'une table Consultation

```
CREATE TABLE consultation(
RPPS UNSIGNED SMALLINT,
IPP UNSIGNED INT,
date Date,
lieu VARCHAR(100),
compte_rendu TEXT
PRIMARY KEY(RPPS, IPP, date)
);
```

Plan

Modification d'une table

Il est possible de modifier une table existante avec les 5 opérations suivantes :

- Ajouter une colonne ;
- Modifier une colonne ;
- Supprimer une colonne ;
- Ajouter une contrainte ;
- Supprimer une contrainte.

Modification d'une table

Ajout d'une colonne

La requête qui permet d'ajouter une colonne est la suivante :

```
ALTER TABLE nom_table
ADD COLUMN nom_colonne type_colonne [contraintes_colonne] ;
```

Modification d'une table

Exemple d'ajout d'une colonne

Exemple

```
ALTER TABLE Ordonnance
ADD COLUMN signature BOOLEAN ;
```

Modification d'une table

Modification d'une colonne

La requête qui permet de modifier une colonne est la suivante :

ALTER TABLE *nom_table*
ALTER COLUMN *nom_colonne* {SET DEFAULT *valeur* | DROP DEFAULT} ;

Modification d'une table

Exemple de modification d'une colonne

Exemple

```
ALTER TABLE Admissions
ALTER COLUMN service SET DEFAULT 'urgences' ;
```

Modification d'une table

Suppression d'une colonne

La requête qui permet de supprimer une colonne est la suivante :

```
ALTER TABLE nom_table
DROM COLUMN nom_colonne [RESTRICT | CASCADE];
```

RESTRICT La suppression est refusée tant qu'il existe des colonnes qui font référence à la colonne qu'on souhaite supprimer

CASCADE La suppression est acceptée mais elle entraîne la suppression de toutes les autres colonnes lui faisant référence.

Modification d'une table

Exemple de suppression d'une colonne

Exemple

```
ALTER TABLE Service
DROP COLUMN chef_service CASCADE;
```

Modification d'une table

Ajout d'une contrainte

La requête qui permet d'ajouter une contrainte est la suivante :

```
ALTER TABLE nom_table
ADD CONSTRAINT [nom_contrainte] definition_contrainte ;
```

Modification d'une table

Exemple d'ajout d'une contrainte

Exemple

```
ALTER TABLE Resultat_examen
ADD CONSTRAINT (CHECK Medecin IS NOT NULL);
```

Modification d'une table

Suppression d'une contrainte

La requête qui permet de supprimer une contrainte est la suivante :

```
ALTER TABLE nom_table
DROP CONSTRAINT nom_contrainte ;
```

Modification d'une table

Exemple de suppression d'une contrainte

Exemple

```
ALTER TABLE Tension
DROP CONSTRAINT contrainte_hypertension ;
```

Plan

Suppression d'une table

La requête qui permet de supprimer une table est la suivante :

```
DROP TABLE nom_table [RESTRICT | CASCADE];
```

RESTRICT La suppression est refusée tant qu'il existe des colonnes qui font référence à la table qu'on souhaite supprimer

CASCADE La suppression est acceptée mais elle entraîne la suppression de toutes les autres colonnes lui faisant référence.