Bases de Données

SQL – Langage de Description

Composants

- SQL-LMD (langage de manipulation de données)
  - Interrogation : select
  - Manipulation : insert, update, delete
- SQL-LDD (langage de description de données)
  - Create, alter, drop
- SQL-LCD : langage de contrôle de données)
  - Grant, revoke
- + commit, rollback

SQL-LDD

CREATE – ALTER - DROP

Contraintes d'intégrité

Les contraintes d’intégrité sont des règles qui doivent être vérifiées en permanence par le SGBD, quelque soit l'opération effectuée sur la base.

Parmi ces contraintes d'intégrité :

- la définition des clés primaires et des clés étrangères.
- l'indication des valeurs possibles pour les attributs.

Contraintes d'intégrité

C'est lors de la création d’une table, qu'il est possible de spécifier une contrainte associée à une colonne. Les contraintes sont de natures différentes :

- La colonne ne peut pas contenir de valeur nulle (NOT NULL)
- La valeur d'un attribut doit être unique (une même colonne ne doit pas contenir deux valeurs identiques dans des tuples différents)
- La condition générale (clause CHECK) qui permet de spécifier un intervalle ou une zone de validité de la valeur d'un attribut.
- La clé primaire
- La clé étrangère

Contraintes d'intégrité

Principales contraintes

- Clause NOT NULL : permet donc d'indiquer qu'un attribut ne peut pas prendre la valeur NULL, sinon (dans le cas d'une tentative d'insertion de valeur nulle), le SGBD renvoie un message d'erreur.

- Exemple : Création de la table «matiere» avec une contrainte sur le libellé, qui doit toujours être renseigné !

- CREATE TABLE matiere

```
( ….
  libelle-m VARCHAR(20) NOT NULL
 ) ;
```

Contraintes d'intégrité

Clause UNIQUE : permet de ne pas stocker deux données identiques pour des tuples différents. Par exemple, ce type de clause peut être appliqué pour un numéro INSEE.

Exemple : Numéro de sécurité sociale unique.

```
 => create table etudiant
(num_etu integer not null primary key,
 nom_etu varchar(20) not null,
 num_secu char(15) unique
);

NOTICE:  CREATE TABLE / PRIMARY KEY créera un
index implicite « etudiant_pkey » pour la table « etudiant »
NOTICE:  CREATE TABLE / UNIQUE créera un index
implicite « etudiant_num_secu_key » pour la table «
etudiant »
CREATE TABLE
=>
```

=> insert into etudiant values (105, 'MARC', '11111111111111');

INSERT 0 1

=> insert into etudiant values (101, null, '293541135487651');

ERREUR: une valeur NULL viole la contrainte NOT NULL de la colonne « nom_etu »

=> insert into etudiant values (1011, 'JACQUES', '111111111111111');

ERREUR: la valeur d'une clé dupliquée rompt la contrainte unique « etudiant_pkey »

=>

Contraintes d'intégrité

Clause CHECK : permet, lors de la création d’une table, de spécifier une contrainte associée à une colonne, pour valider les informations insérées dans cette colonne.
Exemple : Un champ sexe-et ne peut contenir que 'M' ou 'F'.

Exemple : Un champ sexe ne peut contenir que 'M' ou 'F'.

```
Drop table etudiant cascade;
=> CREATE TABLE etudiant (
    nom varchar(20) not null,
    prenom varchar (20),
    sexe char constraint cont_s CHECK (sexe = 'M' or sexe = 'F') );
CREATE TABLE

=> insert into etudiant values ('MARVI', 'loic','M');
INSERT 0 1

=> insert into etudiant values ('BERGET', 'lea','Z');
ERREUR:  la nouvelle ligne viole la contrainte de vérification «
etudiant » de la relation « etudiant_sexe_check »
sadegb=>
```

Rmq : La colonne Sexe ne peut prendre que l'une des 2 valeurs ‘M’ ou ‘F’. Si un utilisateur tente d’insérer une autre valeur, le système retourne un message d’erreur.

CONSTRAINT cont_s permet de donner un nom à la contrainte, dans le but de pouvoir utiliser ce nom pour une éventuelle suppression de cette contrainte

Contraintes d'intégrité

Spécification d’une clé primaire : Une clé primaire est un ensemble de colonnes (ou une seule colonne) dont les valeurs (la valeur) identifie de manière unique chaque tuple (ligne) de la table.

Syntaxe :

```
CREATE TABLE nom_table
  (
    Nom_colonne1 TYPE_DE_DONNEES [NOT NULL], ...
    Nom_colonneN TYPE_DE_DONNEES [NOT NULL],
    [CONSTRAINT nom_contrainte_clé_primaire]
    PRIMARY KEY (nom_colonneA, ..., nom_colonneX)
  ) ;
```

Contraintes d'intégrité

Exemple :

```
=> Drop table etudiant cascade;
=> CREATE TABLE etudiant (
    code integer not null,
    nom varchar(20) not null,
    prenom varchar (20),
    sexe char CHECK (sexe = 'M' or sexe = 'F'),
  CONSTRAINT PK_Code_et PRIMARY KEY (Code)
);
=>
NOTICE:  CREATE TABLE / PRIMARY KEY créera un index implicite
« pk_code_et » pour la table « etudiant »
CREATE TABLE
=>
```

Contraintes d'intégrité

Rmq : Si la clé primaire est composée d'une seule colonne, il est possible d'indiquer la clause PRIMARY KEY au niveau de sa définition.

Exemple :

```
CREATE TABLE etudiant
        ( Code_et INTEGER PRIMARY KEY,
          Nom_et VARCHAR(20) NOT NULL,
          Prenom_et VARCHAR(20),
          CodeSexe_et CHAR CONSTRAINT CK_Sexe
              CHECK  (CodeSexe_et IN ('M', 'F'))
        ) ;
```

Contraintes d'intégrité

Spécification d’une clé étrangère : Pour créer une clé étrangère, on utilise la clause REFERENCES suivie de la table référencée et du champ concerné.

Syntaxe :

=> CREATE TABLE nom_table
(Nom_col_1 TYPE_DE_DONNEES [NOT NULL], ...
Nom_col_n TYPE_DE_DONNEES [NOT NULL],
[CONSTRAINT nom_contrainte_clé_étrangère]
FOREIGN KEY (nom_colonneF1, ...,
nom_colonneFX)
REFERENCES table_referencée (nom_colonneP1, ...,
nom_colonnePX),
[CONSTRAINT nom_contrainte_clé_primaire]
PRIMARY KEY (nom_colonneA, ..., nom_colonneX)
) ;

Contraintes d'intégrité

Exemple : création d’une table représentant la notion d'atelier, puis création d'une table employe qui fait référence à la table atelier dans lequel travaille l'employé.

Commençons par créer la table atelier :

```
=> drop table if exists atelier cascade;
NOTICE:  DROP cascade sur contrainte fk_at_code sur table employe_bis
DROP TABLE
=> CREATE TABLE atelier
      (at_code CHAR default 'A' check (at_code in ('A', 'B')),
        at_intitule VARCHAR(12),
        CONSTRAINT PK_at_Code
PRIMARY KEY (at_code) ) ;
```

Insertions

=> insert into atelier values('A', 'operation');

INSERT 0 1

=> insert into atelier values ('B', 'marketing');

INSERT 0 1

=> insert into atelier values ('X', 'march');

ERREUR: la nouvelle ligne viole la contrainte de vérification « atelier » de la relation « atelier_at_code_check »

=>

- ## Contraintes d'intégrité

Il est maintenant possible de créer la table employe qui fait référence à la table atelier.

Note importante : L’ordre de création des tables est important : créer d'abord la table référencée (ici atelier, qui a été déjà créée), puis on crée employe :

```
=> drop table if exists employe cascade;
NOTICE:  la table « employe » n'existe pas, poursuite du traitement
DROP TABLE
=>CREATE TABLE employe
   (Code_e INTEGER primary key,
     Nom_e VARCHAR(20) NOT NULL,
     Prenom_e VARCHAR(20),
     at_code CHAR REFERENCES atelier (at_code) );


--[ou bien : at_code char references atelier (at_code), ]
```

Rmq : Contraintes d'intégrité

Supposons que atelier a 2 tuples :

{('A', 'operation'), ('B', 'marketing')}

D’après la définition de l’intégrité référentielle, il n'est plus possible d’insérer un employé ayant *at_code* différent de ‘A’ et de ‘B’.

Cependant, à l‘inverse, il est impossible de supprimer ou de modifier un *at_code* qui est déjà mis en relation avec la table employe.

Complément : Contraintes d'intégrité

Pour rendre possible ce type d'opérations (supprimer, modifier), il faut préciser une clause qui permet au système de savoir comment réagir en face de ce type d'action. Pour cela, on utilise les clauses suivantes :

- **ON DELETE *option*** : action à effectuer suite à une tentative de suppression dans la table référencée.

- **ON UPDATE *option*** : action à effectuer suite à une tentative de modification dans la table référencée.

où *option* peut correspondre à une des actions suivantes :

• NO ACTION : équivaut à ne pas spécifier de clause ON DELETE ou ON UPDATE (comportement par défaut).

• CASCADE : associé à ON UPDATE, cette action entraîne la modification des enregistrements présents dans les tables liées. Associé à ON DELETE, les enregistrements liés sont supprimés.

• SET NULL : les changements sont autorisés et la clé étrangère mise en œuvre est mise à NULL.

• SET DEFAULT : les changements sont autorisés et la clé étrangère est mise à la valeur par défaut spécifiée pour ce champ.

Contraintes d'intégrité

Valeur par défaut : La clause DEFAULT permet, lors de la définition d’une colonne, de déclarer une valeur par défaut. Lors de l’insertion d’un tuple, si aucune information n’est spécifiée pour cette colonne, c’est la valeur par défaut qui sera enregistrée.

Exemple

```
Drop table if exists employe cascade;
 CREATE TABLE employe (
    Code_e INTEGER,
    Nom_e VARCHAR(20) NOT NULL,
    Prenom_e VARCHAR(20),
    atelier_e CHAR DEFAULT 'A',
    CONSTRAINT FK_atelier_e FOREIGN KEY (atelier_e)
        REFERENCES atelier (at_code)
            ON UPDATE CASCADE
     ON DELETE SET DEFAULT,
    CONSTRAINT PK_Code1_e PRIMARY KEY (Code_e)) ;

- - atelier = table précédente déjà créée
```

Insertions :

```
=> Insert into employe values (1, 'TOTO', null, 'B');
INSERT 0 1
=> Insert into employe (code_e, nom_e) values (2, 'MALON');
INSERT 0 1
=> select * from employe;
 code_e | nom_e | prenom_e | atelier_e
--------+-------+----------+-----------
      1 | TOTO  |          | B
      2 | MALON |          | A
(2 lignes)
```

```
=>sadegb=> select * from atelier;
 at_code | at_intitule
---------+-------------
 A       | operation
 B       | marketing
(2 lignes)
=> delete from atelier where at_code = 'B';
DELETE 1
sadegb=> select * from employe;
 code_e | nom_e | prenom_e | atelier_e
--------+-------+----------+-----------
      2 | MALON |          | A
      1 | TOTO  |          | A
(2 lignes)
```

=> – TOTO a l'atelier 'A' maintenant (car set default !)

Contraintes d'intégrité : Résumé

Principales contraintes

- **PRIMARY KEY :** définit une clé primaire (la valeur est différente de NULL et unique dans la table)
=> toutes les lignes sont différentes

- **FOREIGN KEY :** définit une clé étrangère. Attribut (ou groupe) qui fait référence à une clé primaire dans une autre table
=> «contrainte d'intégrité référentielle».

- **NOT NULL :** la valeur de l'attribut ne doit pas être à NULL (doit être renseignée).

- **UNIQUE :** chaque tuple de la table doit avoir une valeur différente de celle des autres ou NULL, pour l'attribut qui a cette option.

- **CHECK :** définit un ensemble de valeurs possibles pour l'attribut.

- **DEFAULT :** donne une valeur par défaut (A la création du tuple, c'est la valeur par défaut qui sera fournie, sauf saisie d'une valeur).

Les contraintes PRIMARY KEY, NOT NULL, UNIQUE et CHECK ont le même type de conséquence :

- Si on cherche à donner une valeur à un attribut qui n'est pas conforme à ce qui est précisé dans la définition de l'attribut :
  - une valeur NULL s'il est défini NOT NULL ou PRIMARY KEY,
  - une valeur existant déjà s'il est défini UNIQUE ou PRIMARY KEY,
  - une valeur n'appartenant pas au domaine spécifié par CHECK),
- ALORS le SGBD renvoie un message d'erreur et ne modifie pas la base de données.

FOREIGN KEY : contrainte d'intégrité référentielle : ordre de création/suppression

1. Si on crée une table avec une clé étrangère, il faut que la table à laquelle on fait référence soit déjà créée => ordre logique de création des tables (l'ordre INVERSE pour la suppression).

2. Même chose pour la création et la suppression de tuples (lignes) : même ordre que création/suppression.

3. Rmq : on doit supprimer d'abord le tuple qui référence, ensuite le tuple référencé, sinon IMPOSSIBLE de supprimer : c'est le CAS par défaut, SINON : => les cas qu'on a vu : **{on delete|on update}** ***option***

CREATION D'UNE TABLE

CREATE TABLE nom_tab (<definition_colonne>
[, <definition_colonne>] …);

où <definition_colonne> est :

nom-col type [DEFAULT val_defaut] [NOT NULL] [UNIQUE] [autre]

où autre peut être :

… REFERENCES nom_tab [(nom_col [, nom_col] …)] …;

… CHECK condition;

Rmq : On peut nommer les contraintes

Exemple

Avec le nom de la contrainte de clé primaire :

```
drop table if exists auteur cascade;
CREATE TABLE auteur (
    num_aut INTEGER,
    nom_aut VARCHAR(30),
    CONSTRAINT cle_auteur PRIMARY KEY (num_aut)
);
```

- Drop table if exists livre cascade;

```
CREATE TABLE livre (
    code_l INTEGER PRIMARY KEY,
    titre_l VARCHAR(30) NOT NULL,
    num_aut INTEGER references auteur(num_aut)
) ;
```

```
Insert into auteur values(10, 'Gison');
Insert into auteur values(20, 'lamartine');
Insert into auteur values (30,'hugo');
=> select * from auteur;
 num_aut |  nom_aut
---------+-----------
      10 | Gison
      20 | lamartine
      30 | hugo
(3 lignes)

=> insert into livre values (100, 'godel', 10);
INSERT 0 1
=> insert into livre values (200, 'le printemps', 20);
INSERT 0 1

=> insert into livre values (300, 'guerre mondiale 1', 50);
ERREUR:  une instruction insert ou update sur la table « livre » viole la
contrainte de clé
étrangère « livre_num_aut_fkey »
DÉTAIL : La clé (num_aut)=(50) n'est pas présente dans la table « auteur ».
sadegb=
```

CREATE TABLE – Contraintes : Rappel

NOT NULL : si on ne spécifie rien pour la valeur d ’un attribut, il prend la valeur NULL (indéterminée). Si la clause NOT NULL est présente, on doit spécifier une valeur non nulle pour cet attribut.

DEFAULT : on peut spécifier une valeur par défaut pour un attribut à la création de la table (cette valeur doit être du type de l’attribut).

UNIQUE : spécifie qu’un attribut (ou groupe) est une clé candidate. Un seul tuple correspond à une valeur de cette clé. Il est conseillé de spécifier NOT NULL pour un attribut déclaré UNIQUE (qui n ’est pas clé primaire).

Exemple

Create table fournisseurs
(… nom_fou char(25) NOT NULL UNIQUE
…
);

CHECK : spécifie une contrainte qui doit être vérifiée à tout moment par les tuples concernés.

Exemple : Create table …
(cli_type char(16) DEFAULT ‘PARTICULIER’
CHECK (cli_type IN (‘PARTICULIER’, ‘ PME ’, ‘PMI’)
) ;

La quantité livrée doit être inférieure ou égale à la quantité commandée.

… qte_liv integer default 0 CHECK (qte_liv <= qtç_com….)

NOT NULL : ... CHECK (nom_fou IS NOT NULL) …

Intervalle : . CHECK val_prix BETWEEN 10 AND 40

Contrainte d ’entité : **PRIMARY KEY**

1- Create table client
(numcli char(5) not null primary key,
…
);

2- Si la clé est composée (ici 2 attributs) :
Create table lign_comm
(ref char(5) not null,
design char(10) not null,
primary key (ref, design),
…
);

CI référentielle

=> gestion correcte des modifications de la clé étrangère dans la table qui référence et de la clé primaire dans la table référencée.

Soient les relations :

client(numcl, ….) et commande(numcom, …numcl,..)

si on ajoute une commande => il faut que le client associé existe.

Vérification des CI référentielles

Pour gérer les Contraintes d'intégrité référentielles :vérification assurée par le SGBD automatiquement : la contrainte est déclarée explicitement une fois pour toutes (valable pour tout programme et tout utilisateur).

=> adoptée par la norme SQL-92 ou SQL-2 (et donc dans beaucoup de SGBD)

Exemple

Create table client
(num_cl char(5) not null primary key, ...);

Create table commande
(numcomm integer not null primary key, .
num_com char(5) not null REFERENCES client, ...);

Si la clé étrangère = plusieurs attributs
=> utiliser FOREIGN KEY.
… FOREIGN KEY (atr1_cle, attr2_cle)
REFERENCES (cle_prim1).

On ajoute la clause suivante :

... Num_com references client

[**ON UPDATE {option}**]

[**ON DELETE {option}**]

où option peut être l ’une des options suivantes :

Si on modifie la valeur d ’une clé primaire référencée, SQL permet de réaliser automatiquement certaines opérations :

NO ACTION : équivalent à ne pas mettre de clause ON UPDATE ou ON DELETE => refus de modifier ou supprimer une clé primaire. Situation pas défaut.

ON UPDATE [ON DELETE] CASCADE : si UPDATE d ’une clé primaire référencée => les modif sont répercutées sur les relations qui référencent.
SI DELETE => suppression des clés qui référencent.

SET NULL : si une clé primaire référencée est modifiée => on les met à NULL là où elles sont référencées (condition : les parties de la clé étrangères doivent permettre les valeurs NULL).

SET DEFAULT : si la clé primaire est modifiée
=> la valeur de la clé étrangère qui référence est mise à une valeur par défaut (définie au préalable)

Exemple :

…. deptno REFERENCES dept
ON UPDATE CASCADE
ON DELETE SET NULL;

Modification de la structure d'une table : INSTRUCTION ALTER TABLE

Pour modifier la la structure d’une table, on utilise l’instruction ALTER TABLE :

- d’ajouter un champ (une colonne) à une table (ADD COLUMN)
- de modifier un champ (une colonne) (ALTER COLUMN)
- de supprimer une colonne (DROP COLUMN)
- d’ajouter une contrainte (ADD CONSTRAINT)
- de supprimer une contrainte (DROP CONSTRAINT)

ALTER TABLE nom_tab **ADD** nom_col type ;

ALTER TABLE nom_tab **DROP** COLUMN nom_col ;

ALTER TABLE nom_tab **ALTER** nom_tab type ;

=> ALTER TABLE EMP ALTER column sal type numeric (10,3);

- ALTER TABLE

=> alter table emp add column col varchar(10) ;

- alter table emp add column col numeric;
- ALTER TABLE
- =>

```
=> ALTER TABLE emp ADD COLUMN description text CHECK
(description <> '');
ALTER TABLE
sadegb=>
```

Suppression d ’une table : DROP

DROP TABLE nom-tab;

Ex : Drop table bonus;

Autres exemples

- Ajout d’un champ Age de type décimal

ALTER TABLE etudiant ADD COLUMN Age DECIMAL;

- Modification du type du champ

ALTER TABLE etudiant ALTER COLUMN Age TYPE INTEGER;

- Ajout d’une contrainte sur ce champ

```
ALTER TABLE etudiant ADD CONSTRAINT CK_age
    CHECK (age > 18);
```

- Suppression de la contrainte

```
ALTER TABLE etudiant DROP CONSTRAINT CK_age;
```

- Suppression de la colonne

```
ALTER TABLE etudiant DROP COLUMN Age;
```