Module Base de données Licence 2

Le langage de requêtes SQL: Définition des données (1)

H.BELLEILI
Université Badji Mokhtar Annaba
Département Informatique
2019-2020

Plan

1. La commande CREATE TABLE
2. Les types de Données
3. Les contraintes
4. La commande ALTER TABLE
5. La commande DROP TABLE
6. Les procédures Stockées

CREATE TABLE

Permet de créer une table en définissant le nom de la table, les colonnes (attributs) et le type de chacune des colonnes.

La forme la plus simple:

CREATE TABLE <nom de table>(colonne$_1$ type$_1$, colonne$_2$ type $_2$,é ,colonne$_n$ type$_n$);

<nom de table>: est une chaine de caractère commençant impérativement par un caractère alphabétique (les chiffres et les caractères spéciaux ne doivent pas être au début du nom.

SQL est insensible à la casse

<colonne$_i$>: est un nom d'attribut

<type$_i$> : est le type de l'attribut

TYPES DE DONNEES SUPPORTÉS

Types Numériques

- ***INT*** ou ***INTEGER*****:** entier (un sous-ensemble fini d'entiers qui dépend de l'implémentation)
- ***SMALLINT*** : entier court (un sous-ensemble du type de domaine entier qui dépend de l'implémentation)
- ***DECIMAL(p, d)***, ***DEC(p, d)*** ou ***NUMERIC(p, d)***: nombre réel à point fixe , avec une précision (nombre total de chiffres décimaux) de *p* chiffres et une échelle (nombre de chiffres après la virgule) de *d* chiffres.
- ***REAL***, ***DOUBLE PRECISION***: nombres réels à virgule flottante avec simple et double précision dépendant de l'implémentation.
- ***FLOAT(n)*** : nombre réel avec une précision optionnelle d'au moins n chiffres décimaux, la valeur par défaut de n étant 0, et si n=0 on ***FLOAT*** ⇔***REAL***

Types catactères

- ***CHAR(n)*** ou ***CHARACTER(n)***: les chaînes de caractères de longueurs fixes, la valeur par défaut de la longueur étant 1.

- ***VARCHAR(n)***: les chaînes de caractères de longueurs variables, avec au maximum n caractères. Les chaînes de caractères constantes sont placées entre simples quotes (guillemets) (ex., 'Bonjour tout le monde ')

Types Temporels

- ***DATE*** : réserve 2 chiffres pour le mois et le jour et 4 pour l ’année : Sur 3 octets. L’affichage est au format ‘YYYY-MM-DD’.
- ***TIME*** : pour les heures, minutes et secondes l’heure au format ‘HHH:MM:SS’ sur 3 octets
- ***TIMESTAMP*** : indique un moment précis par une date avec heures, minutes et secondes (6 chiffres après la virgule) ‘YYYY-MM-DD HH:MM:SS’
- Les format de *DATE*, *TIME et TIMESTAMP* peuvent être considérés comme un type spécial de chaîne de caractères. Par conséquent, ils peuvent généralement être utilisés, après les avoir converties en chaînes de caractères équivalentes, dans les comparaisons de chaînes de caractères

Types Binaires

- Les types BLOB (*Binary Large OBject) permettent de stocker des données non structurées* comme le multimédia (images, sons, vidéo, etc.).
- la longueur maximum d'un BLOB peut être spécifiée en kilo-octets (K), méga-octets (M) ou en giga-octets (G). Par exemple, ***BLOB(20G)*** spécifie une chaine de bits de longueur maximum de 20 gigabits.

Types Enumération

- Le type ENUM définit une liste de valeurs permises (chaînes de caractères).

Type	Description
`ENUM('valeur1','valeur2',...)`	Liste de 65 535 valeurs au maximum.

Contraintes d’intégrité

- I. Les contraintes de domaine
- II. Les contraintes d’intégrité d’entité
- III. Les contraintes d ’intégrité référentielle

Contraintes de domaine

- Ensemble des valeurs que peut prendre un attribut.
- Ces contraintes sont décrites dans la définition d ’un attribut, directement après son type .
- **NOT NULL** : on impose que l ’attribut possède une valeur
- **DEFAULT** : on spécifie une valeur par défaut dont le type doit correspondre au type de l ’attribut
- **UNIQUE** : interdit qu’une colonne contienne deux valeurs identiques
  - Pour les colonnes UNIQUE le SGBD crée un index

```
CREATE TABLE  Client (NumCli….
                      nomCli VARCHAR(25) NOT NULL,
                      CaCli INTEGER DEFAULT 0,
                      TypeCli VARCHAR(16) DEFAULT 'Particulier ’ …);
```

CREATE TABLE fournisseurs (... NomFour CHAR(25) **NOT NULL UNIQUE**, ..);

NomFour et NumFour sont deux clés candidates. NumFour a été choisi comme clé primaire

Contrainte de domaine

- **CHECK**(condition)
- 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='MCA' OR grade='MCB')
- CHECK grade IN('MCA','MCB')
- CHECK date_depart<date_arrivee

Contrainte de clé primaire

- La clause PRIMARY KEY.
- Une clé primaire doit toujours avoir une valeur déterminée et unique pour la table.
- 2 cas possibles:
  - Si elle porte sur 1 seul attribut la clause PRIMARY KEY est situé après le type de l'attribut. Et l'attribut est NOT NULL implicitement
  - Si elle porte sur plusieurs attributs (clé composée) la clause PRIMARY KEY est placée avant les attributs qui font partie de la clé
  - Ils sont entre parenthèse séparés par des virgules.
  - Dans ce cas tous les attributs de la clé composée doivent être NOT NULL
- Le SGBD crée automatiquement un INDEX sur la clé primaire

Exemple Clé primaire

- **Clé primaire sur un seul attribut**:

  NumClient Number **PRIMARY KEY** ou bien

  NumClient Number NOT NULL,

  ……

  PRIMARY KEY NumClient

- **Clé primaire composée**:

  NP CHAR(4) NOT NULL,

  NF CHAR(2) NOT NULL,

  NU CHAR(3) NOT NULL

  PRIMARY KEY (NP,NF,NU)

Contraintes d'intégrité référentielles

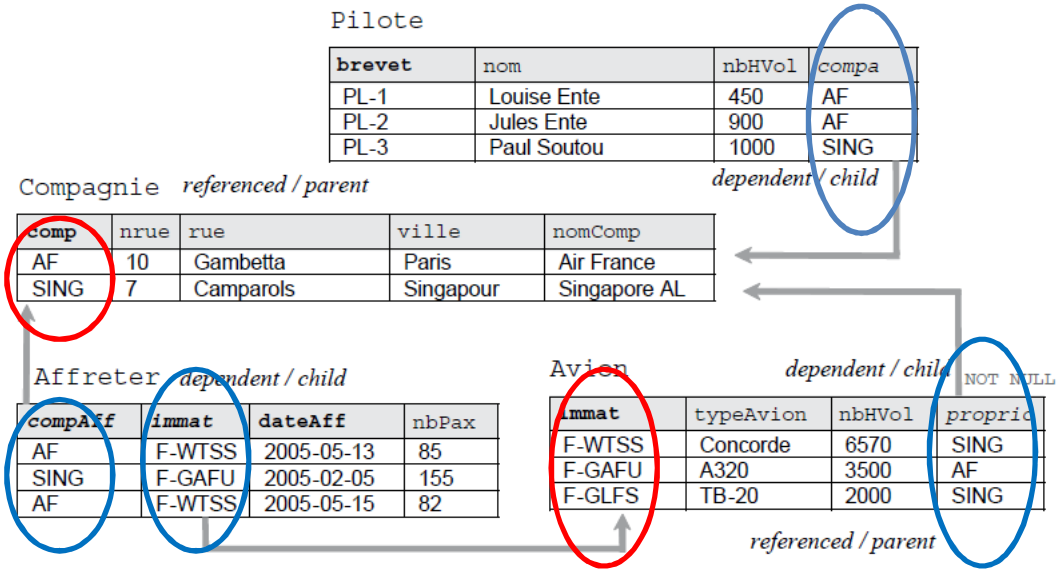
- **L'intégrité référentielle forme le coeur de la cohérence d'une base de données relationnelle**
- C'est une contrainte de clé étrangère
- Elle permet de lier une table « fils »(enfant) à une table « père » (maitre ou parent) possédant un ou plusieurs attributs en commun
- La clé étrangère de la table « fils » Référence la clé primaire (ou candidate) de la table « père »
- On dit que table « fils » est la table qui référence et
- La table « père » est la table référencée
- Deux syntaxes possibles:

<Attribut_clé_étrangère> <type> **REFERENCES** <Table père> (<Attribut_clé_candidate>)

FOREIGN KEY <Attribut_clé_étrangère> **REFERENCES** <Table père> (<Attribut_clé_candidate>)

- Le SGBD crée un index pour les clés étrangères

Exemple



CREATE TABLE Compagnie (comp VARCHAR(4) **PRIMARY KEY**…….)

CREATE TABLE Pilote(….. Compa VARCHAR(4) **NOT NULL REFERENCES** Compagnie(comp)…)

CREATE TABLE Pilote(….. Compa VARCHAR(4) **NOT NULL**
FOREIGN KEY (Compa) **REFERENCES** Compagnie(comp)…)

Intégrité référentielle
problèmes résolus automatiquement par le SGBG

- **La cohérence du « fils » vers le « père »** :

On ne doit pas pouvoir insérer un enregistrement « fils » (ou modifier sa clé étrangère) rattaché à un enregistrement « père » inexistant.

- **La cohérence du « père » vers le « fils »** :

on ne doit pas pouvoir supprimer un enregistrement « père » si un enregistrement « fils » y est encore rattaché.

mysql> **DELETE FROM Compagnie WHERE comp = 'SING';**

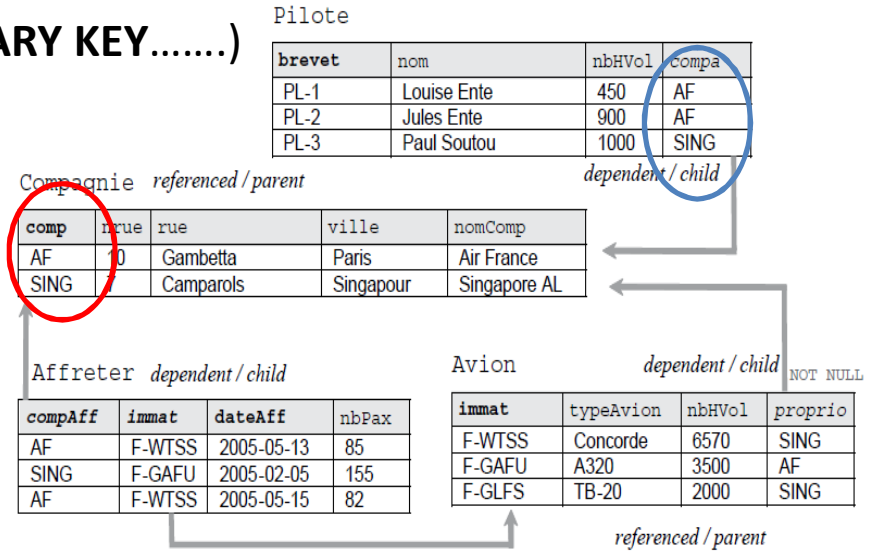
*ERROR **1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`bdsoutou/pilote`,** FOREIGN KEY (`compa`) REFERENCES `compagnie` (`comp`))*

Cohérence du père vers le fils

Il est possible:

- ✓de supprimer les « fils » associés **(ON DELETE CASCADE**),
- ✓ d'affecter la valeur nulle aux clés étrangères des « fils » associés **(ON DELETE SET NULL)** à ce moment là la clé étrangère ne doit pas être **NOT NULL** ou
- ✓de répercuter une modification de la clé primaire du père sur la clé étrangère des fils associés (**ON UPDATE CASCADE)**

CREATE TABLE Compagnie (comp VARCHAR(4) **PRIMARY KEY**…….)



CREATE TABLE Pilote(….. Compa VARCHAR(4) **NOT NULL REFERENCES** Compagnie(comp) **ON DELETE CASCADE**…)

CREATE TABLE Pilote(….. Compa VARCHAR(4) **NOT NULL FOREIGN KEY (**Compa) **REFERENCES** Compagnie(comp) **ON DELETE CASCADE …)**

CREATE TABLE Pilote(….. Compa VARCHAR(4) ~~**NOT NULL**~~ **REFERENCES** Compagnie(comp) **ON DELETE SET NULL**…)

CREATE TABLE Pilote(….. Compa VARCHAR(4) **NOT NULL FOREIGN KEY (**Compa) **REFERENCES** Compagnie(comp) **ON UPDATE CASCADE …)**

Contraintes d'intégrité référentielles

- Il faut ordonner les créations:
  - Les tables référencées (père) doivent être créées avant les tables qui référencent (fils)
- Dans le cas de suppression de table:
  - Les tables fils sont supprimer en premier
- Il faut **ordonner les MAJ :**
  - il faut insérer dans la relation Maitre (la relation référencée) avant d'insérer dans la relation qui la référence

Expression des contraintes d'intégrité
Contraintes colonnes

- Les ***contraintes de colonnes*** *appelées aussi* ***contraintes en ligne*** ne peuvent porter que sur un seul attribut
- Elles sont placées juste après le type de la colonne
- Les différentes variantes sont :
  - Valeur nulle impossible (é .***NOT NULL***),
  - Attribut clé primaire (é ..***PRIMARY KEY)***
  - Unicité de l'attribut (é .***UNIQUE***),
  - Contrainte référentielle (é .***REFERENCES*** *<table référencée> (clé candidate)*
  - Contrainte générale (de contrôle) (é .***CHECK*** *<condition>*);

Expression des contraintes d'intégrité
Contraintes Tables

- Les ***contraintes Table*** peuvent porter sur un ou plusieurs attributs.
- Elles sont obligatoires lorsque la contrainte porte sur plusieurs attributs
- Elles sont placée avant lâattribut
- Elle sont exprimées à l'aide des syntaxes suivantes:
  - *PRIMARY KEY (<attribut>$^{+}$) ,**UNIQUE (**<attribut>+)*
  - Contrainte référentielle, permettant de spécifier quelles colonnes référencent celles d'une autre table,

***FOREIGN KEY** (<Attribut clé étrangère>$^{+}$) **REFERENCES** <table référencée> (<Attribut clé candidate>$^{+}$),*

- Contrainte générale, ***CHECK** <condition>.*

.

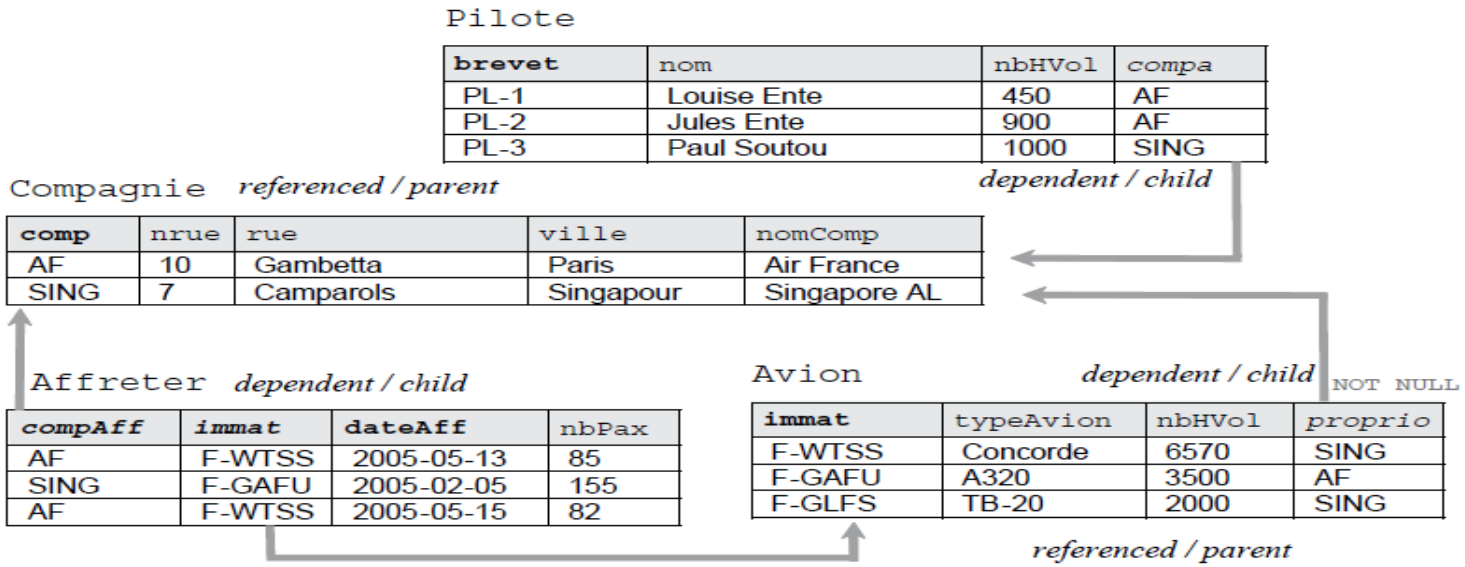
Nommage des Contraintes

- Il est conseillé de nommer la contrainte, sinon MySQL s'en charge.

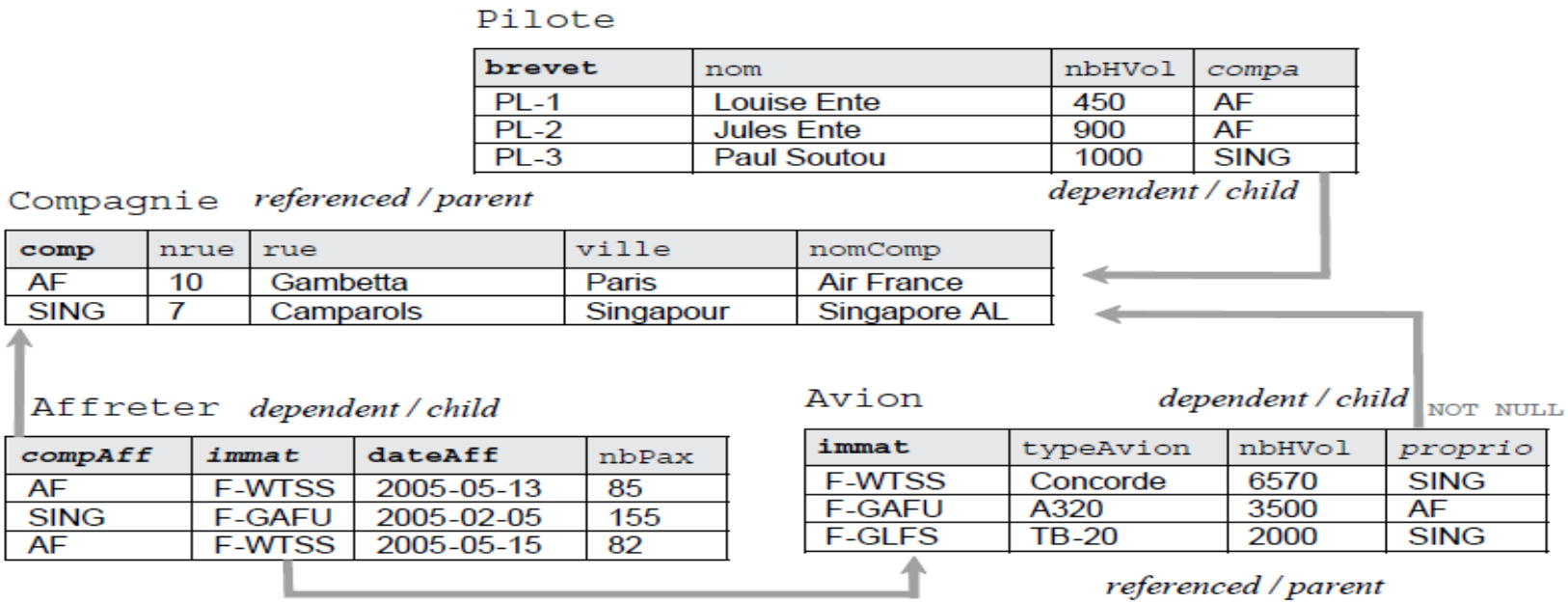
```
CONSTRAINT nomContrainte
UNIQUE (colonne1 [,colonne2]...)
PRIMARY KEY (colonne1 [,colonne2]...)
FOREIGN KEY (colonne1 [,colonne2]...)
REFERENCES nomTablePere [(colonne1 [,colonne2]...)]
            [ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
            [ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
CHECK (condition)
```

Conventions recommandées

- Préfixez par `pk_` le nom d'une contrainte clé primaire, `fk_` une clé étrangère, `ck_` une vérification, `un_` une unicité.
- Pour une contrainte clé primaire, suffixez du nom de la table la contrainte (exemple `pk_Avion`).
- Pour une contrainte clé étrangère, renseignez (ou abrégez) les noms de la table source, de la clé, et de la table cible (exemple `fk_Pil_compa_Comp`).



Tables	Contraintes
CREATE TABLE Compagnie (comp CHAR(4), nrue INTEGER(3), rue CHAR(20), ville CHAR(15) **DEFAULT** 'Paris' nomComp CHAR(15) **NOT NULL**, **CONSTRAINT** pk_Compagnie **PRIMARY KEY**(comp));	Deux contraintes en ligne et une contrainte nommée de clé primaire.
CREATE TABLE Pilote (brevet CHAR(6), nom CHAR(15) **NOT NULL**, nbHVol DECIMAL(7,2), compa CHAR(4), **CONSTRAINT** pk_Pilote **PRIMARY KEY**(brevet), **CONSTRAINT** ck_nbHVol CHECK(nbHVol BETWEEN 0 AND 20000), **CONSTRAINT** un_nom **UNIQUE** (nom), **CONSTRAINT** fk_Pil_compa_Comp **FOREIGN KEY** (compa) **REFERENCES** Compagnie(comp));	Une contrainte en ligne et quatre contraintes nommées : • Clé primaire • NOT NULL • CHECK (nombre d'heures de vol compris entre 0 et 20 000) • UNIQUE (homonymes interdits) • Clé étrangère




```
CREATE TABLE Affreter
 (compAff CHAR(4), immat CHAR(6), dateAff DATE, nbPax INTEGER(3),
  CONSTRAINT pk_Affreter PRIMARY KEY (compAff, immat, dateAff),

  CONSTRAINT fk_Aff_na_Avion
      FOREIGN KEY(immat) REFERENCES Avion(immat),

 CONSTRAINT fk_Aff_comp_Compag
     FOREIGN KEY(compAff) REFERENCES Compagnie(comp));
```

Contrôle des Contraintes d'intégrité

- Un SGBD doit garantir la cohérence des Données lors des MAJ
- Les Contraintes sont déclarées explicitement et sont mémorisées dans le catalogue (dictionnaire de Données)
- Les MAJ portant sur des données dépendantes doivent faire passer la base d'un état cohérent à un autre état cohérent,
- Pour garder la base de données cohérente, le SGBD doit vérifier que les CI sont satisfaites à chaque fin d'opération de MAJ,

Méthodes de vérification de cohérences

- Méthode de détection:
  - Après une opération de modification de données (INSERT, DELETE ou UPDATE) un test appelé **post-test** est lancé par le SGBD qui consiste à vérifier si une CI a été violée,
- Méthode de prévention
  - Elle consiste à empêcher les modifications de la base qui violeraient une quelconque CI. Pour cela un test avant MAJ appelé **pré-test** permet de garantir la non violation dûne CI. Ce test est lancé par le SGBD

Contrôle sur les contraintes simples

- Unicité de la clé
  - Tout SGBD gère en général un index (table dans laquelle il y a toutes les clés primaires) sur les clés primaires. **Un pré-test** simple consiste à vérifier que la nouvelle clé ne figure pas dans lôindex. Ce pré-test est effectué lors **de lôinsertion dôun tuple ou de la modification dôune clé dans la table.**
- Contrainte dôintégrité référentielle (clé étrangère)
  - Deux tables, la table référencée et la table dépendante sont mises en jeu par une contrainte référentielle. 4 types de modification nécessitent des **vérifications**:
- 1) Insertion dans la table fils**: pré-test** simple qui consiste à vérifier lôexistence de la valeur de la colonne clé étrangère dans lôindex (table dans laquelle il y a toutes les clés)
- 2) MAJ de la colonne clé étrangère dans la table fils. Le pré-test est identique au précédent
- 3) suppression dans la table maitre: le pré-test consiste à vérifier quôil nôexiste pas de tuple contenant la valeur de clé à supprimer dans la colonne de clé étrangère. Il y a plusieures possibilités:
  - soit rejeter la MAJ,
  - Soit supprimer les tuples de la table dépendantes (ON DELETE CASCADE)

Référence


Christian Soutou
Apprendre
SQL avec
MySQL
Avec 40 exercices corrigés

FIN Partie 1 LDD

Module Base de données Licence 2

Le langage de requêtes SQL: Définition des données (2)

H.BELLEILI
Université Badji Mokhtar Annaba
Département Informatique
2019-2020

Plan

- ALTER TABLE
- DROP TABLE
- Procédures stockées

Commande ALTER TABLE

- Permet des:
  - Modifications structurelles
  - Modifications comportementales

Modification structurelles (ADD)

- Ajout de colonne (**ADD**)
- ALTER TABLE <nom_table> ADD (<colonne> type_col [DEFAULT valeur [NOT NULL]]

```
ALTER TABLE Pilote ADD (nbHVol DECIMAL(7,2));
ALTER TABLE Pilote
      ADD (compa VARCHAR(4)  DEFAULT 'AF',
           ville VARCHAR(30) DEFAULT 'Paris' NOT NULL);
```

Modifications structurelles (CHANGE)

- Permet de renommer une colonne existante.
- Le type doit être **reprécisé**. (sans le changer)
- **La position de la colonne** peut aussi être modifiée en même temps.

```
ALTER TABLE [nomBase].nomTable CHANGE [COLUMN] ancienNom
        nouveauNom typeMySQL [NOT NULL | NULL] [DEFAULT valeur]
            [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
                                        [REFERENCES ...]
          [FIRST|AFTER nomColonne];
```

```
ALTER TABLE Pilote ADD (nbHVol DECIMAL(7,2));
ALTER TABLE Pilote
      ADD (compa VARCHAR(4)  DEFAULT 'AF',
           ville VARCHAR(30) DEFAULT 'Paris' NOT NULL);
```

```
ALTER TABLE Pilote CHANGE ville adresse VARCHAR(30) AFTER nbHVol;
```

Modifications structurelles (MODIFY)

- L'option **MODIFY** de l'instruction ALTER TABLE modifie le **type d'une colonne** existante sans la renommer.
- On peut augmenter la taille des types numériques/ chaine
- La diminution de la taille est possible (sauf pour les colonnes indexées) à condition que la taille des valeurs déjà existantes entre dans la nouvelle taille.
- **Les contraintes en ligne** peuvent être aussi modifiées par cette instruction. Les données présentes devront toutes vérifier cette nouvelle contrainte.

```
ALTER TABLE [nomBase].nomTable MODIFY [COLUMN] nomColonneAmodifier
      typeMySQL [NOT NULL | NULL] [DEFAULT valeur]
            [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
                                    [REFERENCES ...]
        [FIRST|AFTER nomColonne];
```

MODIFY (exemple)

```
CREATE TABLE Pilote
(brevet CHAR(6), nom CHAR(15) NOT NULL,
 nbHVol DECIMAL(7,2), compa CHAR(4),
 CONSTRAINT pk_Pilote PRIMARY KEY(brevet),
 CONSTRAINT ck_nbHVol
            CHECK(nbHVol BETWEEN 0 AND 20000),
 CONSTRAINT un_nom UNIQUE (nom),
 CONSTRAINT fk_Pil_compa_Comp
      FOREIGN KEY (compa)
             REFERENCES Compagnie(comp));
```

Une contrainte en ligne
et quatre contraintes nommées :

- Clé primaire
- NOT NULL
- CHECK (nombre d'heures de vol compris entre 0 et 20 000)
- UNIQUE (homonymes interdits)
- Clé étrangère

Instructions SQL	Commentaires

`ALTER TABLE Pilote`
`  MODIFY compa VARCHAR(6)`
`  DEFAULT 'SING';`
`INSERT INTO Pilote (brevet, nom)`
`  VALUES ('PL-2', 'Laurent Boutrand');`
	Augmente la taille de la colonne compa et change la contrainte de valeur par défaut puis insère un nouveau pilote.

`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 (possible car les données contenues dans la colonne ne dépassent pas quatre caractères).
`ALTER TABLE Pilote` `  MODIFY compa CHAR(4);`	Rend possible l'insertion de valeur nulle dans la colonne compa.

Modifications structurelles
valeurs par défaut

L'option **ALTER COLUMN** de l'instruction **ALTER TABLE** modifie la valeur par défaut d'une colonne existante

```
ALTER TABLE [nomBase].nomTable ALTER [COLUMN] nomColonneAmodifier
      {SET DEFAULT 'chaine' | DROP DEFAULT};
```

```
ALTER TABLE Pilote ADD (nbHVol DECIMAL(7,2));
ALTER TABLE Pilote
      ADD (compa VARCHAR(4)  DEFAULT 'AF',
           ville VARCHAR(30) DEFAULT 'Paris' NOT NULL);
```

```
ALTER TABLE Pilote CHANGE ville adresse VARCHAR(30) AFTER nbHVol;
```

```
ALTER TABLE Pilote ALTER COLUMN adresse SET DEFAULT 'Blagnac';
ALTER TABLE Pilote ALTER COLUMN compa DROP DEFAULT;
```

Modifications structurelles (DROP)

- Permet de supprimer une colonne
- Il n'est pas possible de supprimer avec cette instruction :
  - toutes les colonnes d'une table ;
  - les colonnes qui sont clés primaires (ou candidates par UNIQUE) référencées par des clés étrangères.

```
ALTER TABLE [nomBase].nomTable DROP
  { [COLUMN] nomColonne | PRIMARY KEY
    | INDEX nomIndex | FOREIGN KEY nomContrainte }
```

```
ALTER TABLE Pilote DROP COLUMN adresse;
```

Modifications comportementales

- Ce sont les mécanismes d'ajout, de suppression, d'activation et de désactivation de contraintes..
  - ADD CONSTRAINT
  - Suppression de contrainte

ADD CONSTRAINT

```
ALTER TABLE [nomBase].nomTable ADD
   { INDEX [nomIndex] [typeIndex] (nomColonne1,...)
   | CONSTRAINT nomContrainte typeContrainte }
```

Trois types de contraintes sont possibles :

- UNIQUE (*colonne1* [,*colonne2*]...)
- PRIMARY KEY (*colonne1* [,*colonne2*]...)
- FOREIGN KEY (*colonne1* [,*colonne2*]...)

```
      REFERENCES nomTablePère (col1 [,col2]...)
        [ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
        [ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
```

Compagnie

unique

comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
SING	7	Camparols	Singapour	Singapore AL

INDEX NOT NULL

Affreter

compAff	immat	dateAff	nbPax
AF	F-WTSS	2003-05-13	85
SING	F-GAFU	2003-02-05	155
AF	F-WTSS	2003-05-15	82

Avion

immat	typeAvion	nbHVol	proprio
F-WTSS	Concorde	6570	SING
F-GAFU	A320	3500	AF
F-GLFS	TB-20	2000	SING

```
ALTER TABLE Compagnie ADD CONSTRAINT un_nomC UNIQUE (nomComp);
```

```
ALTER TABLE Avion ADD INDEX (proprio);
ALTER TABLE Avion ADD CONSTRAINT fk_Avion_comp_Compag
                  FOREIGN KEY(proprio) REFERENCES Compagnie(comp);
ALTER TABLE Avion MODIFY proprio CHAR(4) NOT NULL;
```

```
ALTER TABLE Affrete ADD (
      CONSTRAINT pk_Affreter         PRIMARY KEY (compAff, immat, dateAff),
      CONSTRAINT fk_Aff_na_Avion     FOREIGN KEY(immat) REFERENCES
                                     Avion(immat),
      CONSTRAINT fk_Aff_comp_Compag  FOREIGN KEY(compAff)
                                     REFERENCES Compagnie(comp));
```

Suppression de CONTRAINTE

- NOT NULL: on supprime une contrainte NOT NULL en la modifiant (via MODIFY) à NULL
- UNIQUE:
  - DROP INDEX <nom de la contrainte UNIQUE>
- FOREIGN KEY:
  - DROP FOREIGN KEY <le nom de la contrainte>
- PRIMARY KEY:
  - DROP PRIMARY KEY
    - ***Si la colonne clé primaire à supprimer contient des clés étrangères, il faut d'abord retirer les contraintes de clé étrangère.***
    - ***Si la clé primaire à supprimer est référencée par des clés étrangères d'autres tables, il faut d'abord retirer les contraintes de clé étrangère de ces autres tables.***

Suppression de CONTRAINTE

Contrainte NOT NULL

```
ALTER TABLE Avion MODIFY proprio CHAR(4) NULL;
```

Contrainte UNIQUE

```
ALTER TABLE Compagnie DROP INDEX un_nomC;
```

Contrainte Clé Etrangère

```
ALTER TABLE Avion DROP FOREIGN KEY fk_Avion_comp_Compag;
```

Contrainte clé primaire

```
ALTER TABLE Affreter DROP FOREIGN KEY fk_Aff_na_Avion;
ALTER TABLE Affreter DROP FOREIGN KEY fk_Aff_comp_Compag;

ALTER TABLE Affreter DROP PRIMARY KEY;
```

Affreter *dependent / child*

compAff	immat	dateAff	nbPax
AF	F-WTSS	2005-05-13	85
SING	F-GAFU	2005-02-05	155
AF	F-WTSS	2005-05-15	82

Désactivation/réactivation des contraintes d'intégrité référentielles

L'instruction `SET FOREIGN_KEY_CHECKS=0` permet de désactiver temporairement (jusqu'à la réactivation) toutes les contraintes référentielles d'une base.

L'instruction `SET FOREIGN_KEY_CHECKS=1` permet de réactiver toutes les contraintes référentielles d'une base.

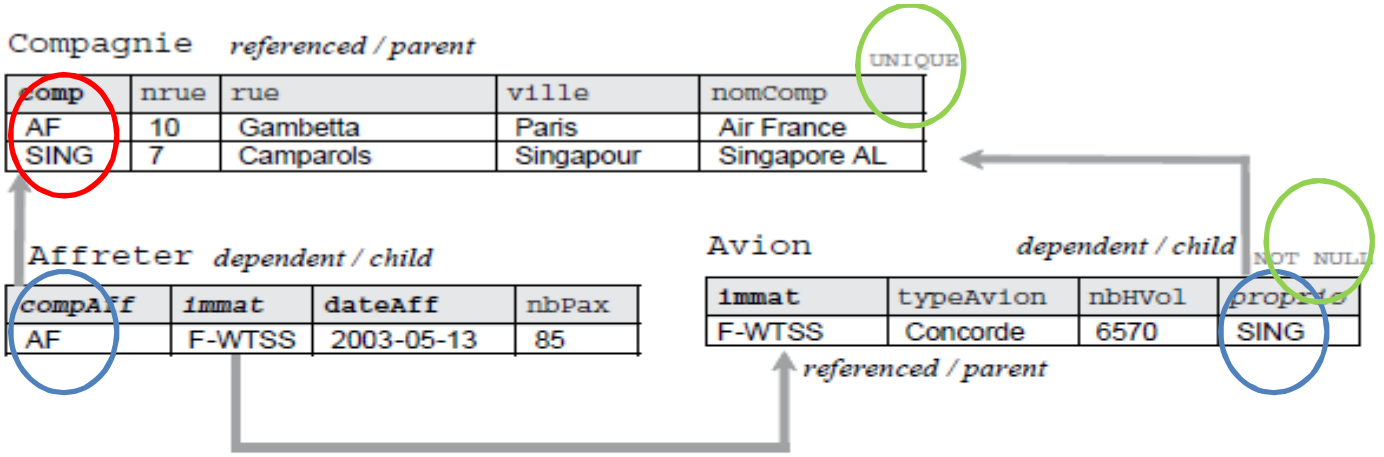


Tableau 3-3 Insertions après la désactivation de l'intégrité référentielle

Instructions valides	Instructions non valides
mysql> **SET FOREIGN_KEY_CHECKS**=0;	
mysql> INSERT INTO Avion VALUES ('F-GLFS', 'TB-22', 500, **'Toto'**); Query OK, 1 row affected (0.04 sec)	mysql> INSERT INTO Compagnie VALUES ('GTR', 1, 'Brassens', 'Blagnac', **'Air France'**); ERROR **1062** (23000): Duplicate entry 'Air France' for key 2
mysql> INSERT INTO Avion VALUES **('Bidon1'**, 'TB-21', 1000, 'AF'); Query OK, 1 row affected (0.10 sec)	mysql> INSERT INTO Avion VALUES ('Bidon1', 'TB-20', 2000, **NULL**); ERROR **1048** (23000): Column 'proprio' cannot be null
mysql> INSERT INTO Affreter VALUES ('AF', **'Toto'**, '2005-05-13', 0); Query OK, 1 row affected (0.10 sec)	mysql> INSERT INTO Avion VALUES ('F-GLFS', 'TB-21', 1000, 'AF'); ERROR **1062** (23000): Duplicate entry 'F-GLFS' for key 1
mysql> INSERT INTO Affreter VALUES **('GTI'**, 'F-WTSS', '2005-11-07',10); Query OK, 1 row affected (0.02 sec)	
mysql> INSERT INTO Affreter VALUES **('GTI'**, **'Toto'**, '2005-11-07',40); Query OK, 1 row affected (0.03 sec)	

DROP Table

- Cette commande permet de supprimer une table de la base de données.
- Les lignes de la table et la définition elle-même sont détruites.
- L'espace occupé par la table est libéré.

- **DROP Table <nom_table>**

Procédures stockées (PS)

- Une procédure stockée est un ensemble nommé d'instructions SQL, précompilée et stockée sur le serveur. Elle permet d'encapsuler les tâches répétitives afin de les exécuter efficacement.
- **Avantages des procédures**
  - **performance** ***: les PS sont plus performantes car le serveur vérifie la syntaxe à la création et compile la*** procédure une fois pour toute. Les exécutions ultérieures de ces procédures seront plus rapides ;
  - **réutilisabilité** ***: la PS peut être appelée à n'importe quel moment. Ceci permet une modularité et*** encourage la réutilisation du code ;
  - **simplification** ***: les règles et politiques de fonctionnement encapsulées dans les PS peuvent être*** modifiées au même endroit. Tous les clients peuvent utiliser les mêmes PS afin de garantir la cohérence des accès aux données et de leurs modifications (sécurité) ;
  - **Accès réseau** ***: elles contribuent à la réduction du trafic sur le réseau. Au lieu d'envoyer des centaines*** d'instructions SQL sur le réseau, les utilisateurs peuvent effectuer une opération complexe à l'aide d'une seule instruction, réduisant ainsi le nombre de demandes échangées entre le client et le serveur.

Gestion des procédures stockées

Création:

L'instruction **CREATE PROCEDURE permet de créer une procédure stockée**

```
CREATE PROCEDURE nom_procedure
([ [IN/OUT/INOUT] Parametre type_donnees [,..]])
Begin  Instructions ; End
```

Exécution:

```
CALL nom_procedure ( [{value| @variable}]);
```

Modification :

La seule façon de modifier une procédure existante en MySQL est de la supprimer puis de la recréer avec les modifications (**à la différence des autres SGBD).**

Suppression :

L'instruction **DROP PROCEDURE permet de supprimer une procédure stockée :**

```
DROP PROCEDURE psInfoDuClient
```

Référence


Christian Soutou
Apprendre
SQL avec
MySQL
Avec 40 exercices corrigés

FIN
SQL LDD