

Manipulation des données et évolution d'un schéma

Pierre Lefebvre

Thèmes abordés

- Rappels sur le langage de définition de données (LDD)
- La gestion des index
- La manipulation des données
- L'évolution d'un schéma
- L'intégrité référentielle
- Démo MySql

Rappels sur le langage de définition de données (DDL)

Rappels: qu'est-ce qu'une base de données ?

- C'est un ensemble de données organisées
- Une base de données contient des tables qui décrivent les types des données
- Les tables contiennent à leur tour des enregistrements qui sont les vraies données
- En utilisant un identifiant commun entre les tables il est possible de les « relier » entre elles
- Jargon
 - Une relation est une table comportant des colonnes (appelées aussi attributs) dont le nom et le type caractérise le contenu qui sera inséré dans la table.
 - Les enregistrements sont également appelés des tuples

Exemple

- Supposons que l'on veuille stocker dans une base de données la liste des participants à un colloque. On va donc créer une relation (table) Personne qui aura pour attributs (colonnes) : nom, prénom, e-mail, laboratoire.

Personne

Nom	Prénom	E-Mail	Laboratoire
Charnay	Daniel	charnay@in2p3.fr	Centre de Calcul
Macchi	Pierre-Etienne	macchi@in2p3.fr	Centre de Calcul

Les opérateurs relationnels

- **la sélection** (ou restriction) :
 - obtenir les lignes de répondant à certains critères ;
- **la projection**
 - obtenir une partie des attributs des lignes de la relation ;
- **l'union**
 - obtenir tout ce qui se trouve dans la relation A ou dans la relation B ;
- **l'intersection**
 - obtenir tout ce qui se trouve à la fois dans la relation A et dans la relation B ;
- **la différence**
 - obtenir ce qui se trouve dans la relation A mais pas dans la relation B ;
- **la jointure**
 - obtenir l'ensemble des lignes provenant de la liaison de la relation A et de la relation B à l'aide d'une information commune.

Les opérations du DDL

Les opérations du DDL :

Création d'un schéma

Création d'une table

- colonnes (obligatoire)

- domaine

- identifiants primaire et secondaire

- clé étrangère

Suppression d'une table

Ajout, suppression, modification d'une colonne

Ajout, suppression d'une contrainte

Ajout, suppression d'un index

Création d'un schéma (database avec MySql)

Création d'un schéma

```
create schema CLICOM;
```

Connexion à une base de données

```
connect to ADMIN\SERV01 as CON_21Dec2009_002  
user 'jlh' password 'jlhclicom';
```

Déconnexion

```
disconnect CON_21Dec2009_002;
```

Création d'une table et de ses colonnes

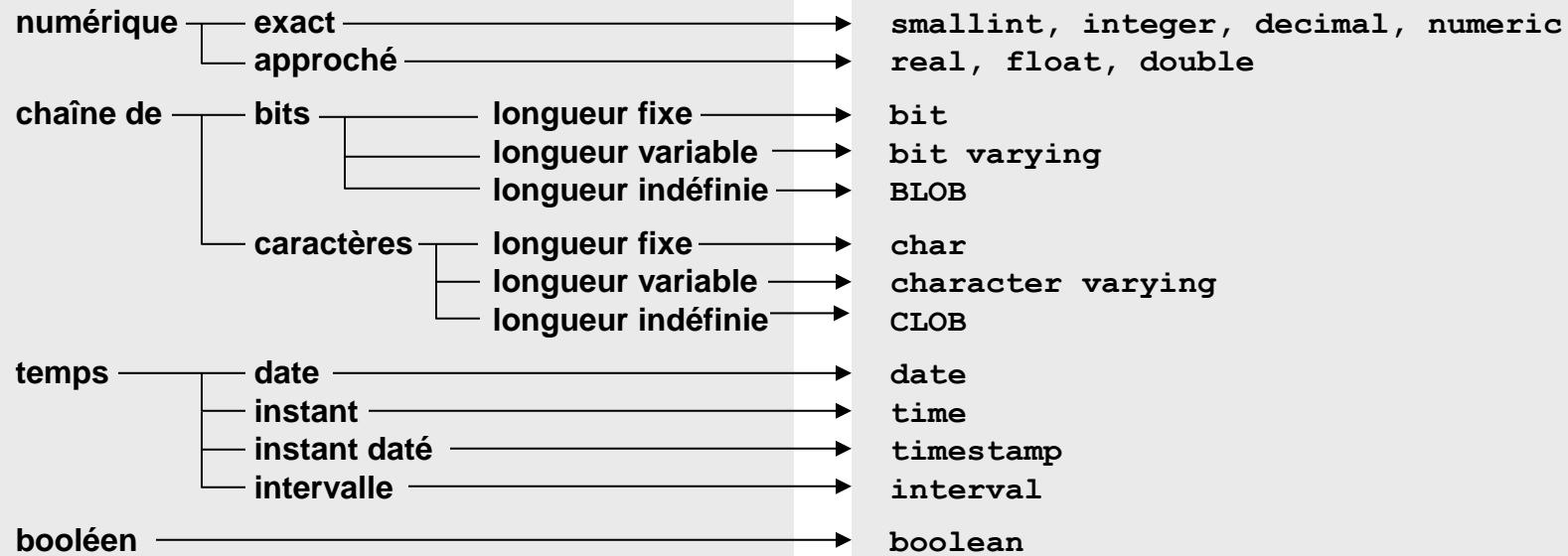
CLIENT
NCLI: char (10)
NOM: char (32)
ADRESSE: char (60)
LOCALITE: char (30)
CAT[0-1]: char (2)
COMPTE: num (9,2)

```
create table CLIENT ( NCLI      char(10) ,  
                      NOM       char(32) ,  
                      ADRESSE   char(60) ,  
                      LOCALITE  char(30) ,  
                      CAT       char(2) ,  
                      COMPTE    decimal(9,2) );
```

Types des attributs

- Les attributs peuvent avoir des **types** très différents :
 - Nombre entier signé ou non (numéro d'ordre, nombre d'objets)
 - Nombre à virgule (prix, température)
 - Date et heure (date de naissance, heure de l'insertion)
 - Enumération (un laboratoire parmi une liste)
 - Ensemble (une ou des compétences parmi une liste)
- Il s'agit de **choisir le type le plus adapté aux besoins**
- Les types requièrent une plus ou moins grande quantité de données à stocker : il vaut mieux choisir un type varchar pour stocker un nom qu'un type longtext.
- Les types de MySQL sont conformes avec la norme SQL ANSI mais des ajouts ont été pratiqués

Les types de base



Les entiers

nom	taille en octets	borne inférieure	borne supérieure
TINYINT	1	-128	127
TINYINT UNSIGNED	1	0	255
SMALLINT	2	-32768	32767
SMALLINT UNSIGNED	2	0	65535
MEDIUMINT	3	-8388608	8388607
MEDIUMINT UNSIGNED	3	0	16777215
INT*	4	-2147483648	2147483647
INT* UNSIGNED	4	0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
BIGINT UNSIGNED	8	0	18446744073709551615

* : **INTEGER** est un synonyme de **INT**

Les flottants

Les flottants (ou nombres réels) sont des nombres à virgule. Contrairement aux entiers

nom	taille en octets
FLOAT	4
DOUBLE	8
REAL	8
NUMERIC(M,D)*	M+2 si D>0 M+1 si D=0 D+2 si M<D
DECIMAL(M,D)*	M+2 si D>0 M+1 si D=0 D+2 si M<D

- *: ce sont des types numériques exacts : ils représentent une valeur numérique exacte.
- A contrario DOUBLE, FLOAT sont des types numériques flottants, ils ne représentent une valeur qu'avec une précision limitée.
- M=précision D=échelle

Les chaînes

nom	longueur
CHAR(M)	Chaîne de taille fixée à M, où $1 < M < 255$, complétée avec des espaces si nécessaire.
VARCHAR(M)	Chaîne de taille variable, de taille maximum M, où $1 < M < 255$, complété avec des espaces si nécessaire.
TINYTEXT	Longueur maximale de 255 caractères.
TEXT	Longueur maximale de 65535 (2^8) caractères.
MEDIUMTEXT	Longueur maximale de 16777215 (2^{16}) caractères.
LONGTEXT	Longueur maximale de 4294967295 (2^{32}) caractères.

- Les types TINYTEXT, TEXT , MEDIUMTEXT et LONGTEXT peuvent être remplacés TINYBLOB, BLOB, MEDIUMBLOB et LONGBLOB
- Ils ne diffèrent que par la sensibilité à la casse caractéristique des BLOB alors que les TEXT sont insensibles à la casse lors des tris et recherches.
- Les BLOB peuvent être utilisés pour stocker des données binaires (images, ...)
- Les VARCHAR, TEXT et BLOB sont de taille variable alors que les CHAR sont de taille fixe

Dates et heures

nom	Taille en octets	description
DATE	3	Date au format anglophone AAAA-MM-JJ.
DATETIME	8	Date et heure au format anglophone AAAA-MM-JJ HH:MM:SS.
TIMESTAMP	4	Affiche la date et l'heure sans séparateur : AAAAMMJJHHMMSS.
TIME	3	Heure au format HH:MM:SS.
YEAR	1	Année au format AAAA.

nom	description
TIMESTAMP(2)	AA
TIMESTAMP(4)	AAMM
TIMESTAMP(6)	AAMMJJ
TIMESTAMP(8)	AAAAMMJJ
TIMESTAMP(10)	AAMMJJHHMM
TIMESTAMP(12)	AAMMJJHHMMSS
TIMESTAMP(14)	AAAAMMJJHHMMSS

En cas d'insertion d'un enregistrement en laissant vide un attribut de type TIMESTAMP, celui-ci prendra automatiquement la date et heure de l'insertion. Contrairement à Unix (où le timestamp est le nombre de secondes écoulées depuis le 1er janvier 1970), en MySQL, il est une chaîne de format comme indiqué ci-contre

Les domaines

Définition d'un domaine :

```
create domain MONTANT decimal(9,2);
create domain MATRICULE char(10);
create domain LIBELLE char(32);
```

Utilisation d'un domaine :

```
create table CLIENT ( NCLI      MATRICULE ,
                      NOM       LIBELLE ,
                      ADRESSE   char(60) ,
                      LOCALITE  LIBELLE ,
                      CAT       char(2) ,
                      COMPTE    MONTANT );
```

Les colonnes obligatoires

Une colonne est **facultative** par défaut. Il faut déclarer explicitement les colonnes **obligatoires**

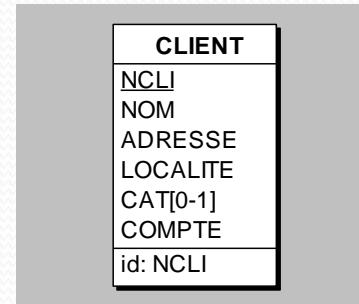
```
create table CLIENT ( NCLI      char(10) not null,  
                      NOM       char(32) not null,  
                      ADRESSE   char(60) not null,  
                      LOCALITE  char(30) not null,  
                      CAT       char(2),  
                      COMPTE    decimal(9,2) not null  
                    ) ;
```

Valeur par défaut d'une colonne

Sera assignée à la colonne si on ne spécifie pas de valeur lors de la création d'une ligne

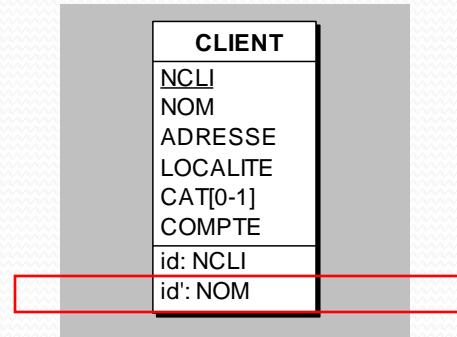
```
create table CLIENT (
    NCLI      char(10) not null,
    NOM       char(32) not null,
    ADRESSE   char(60) not null,
    LOCALITE  char(30) not null default 'Paris',
    CAT       char(2)  default 'B1',
    COMPTE    decimal(9,2) not null default 0.0
);
```

Les identifiants primaires (*primary key*)



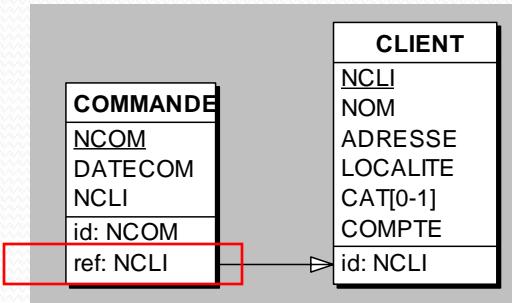
```
create table CLIENT ( NCLI      char(10) not null,
                      NOM       char(32) not null,
                      ADRESSE   char(60) not null,
                      LOCALITE  char(30) not null,
                      CAT        char(2),
                      COMPTE    decimal(9,2) not null,
                     primary key (NCLI) );
```

Les identifiants secondaires (*candidate key*)



```
create table CLIENT ( NCLI      char(10) not null,  
                      NOM       char(32) not null,  
                      ADRESSE   char(60) not null,  
                      LOCALITE  char(30) not null,  
                      CAT        char(2),  
                      COMPTE    decimal(9,2) not null,  
                     primary key (NCLI),  
                     unique (NOM) );
```

Les clés étrangères (*foreign key*)



```
create table COMMANDÉ (NCOM          char(12) not null,  
                      NCLI           char(10) not null,  
                      DATECOM       date not null,  
                      primary key (NCOM),  
                      foreign key (NCLI) references CLIENT);
```

Variante : contraintes de colonne

```
create table CLIENT (
    NCLI char(10) not null primary key,
    . . .
);
```

```
create table CLIENT (
    NCLI char(10) not null,
    NOM char(32) not null unique,
    . . .
);
```

```
create table COMMANDE (
    NCOM char(12) not null,
    NCLI char(10) not null references CLIENT,
    . . .
);
```

Variante : contraintes nommées

```
create table COMMANDE (NCOM      char(12) not null,
                      NCLI      char(10) not null,
                      DATECOM   date not null,
                      constraint COMPK primary key (NCOM),
                      constraint COMFK foreign key (NCLI)
                      references CLIENT);
```

```
create table COMMANDE (
    NCOM  char(12) constraint COMPK not null,
    NCLI  char(10) not null constraint COMFK
    references CLIENT,
    . . . );
```

Types de relation

- Un à un
- Un à plusieurs
- Plusieurs à plusieurs

Relation un à un

```
/* Create "person" table */
CREATE TABLE person (
    person_id INT NOT NULL AUTO_INCREMENT,
    pname varchar(255) NOT NULL,
    PRIMARY KEY (person_id)
) ENGINE=InnoDB;
```

```
/* Create "primary_address" table with FOREIGN KEY */
CREATE TABLE primary_address (
    primary_address_id INT NOT NULL,
    address varchar(255) NOT NULL,
    p_id INT NOT NULL,
    PRIMARY KEY (primary_address_id),
    FOREIGN KEY (p_id) REFERENCES person (person_id)
) ENGINE=InnoDB;
```

person_id	pname
1	Sang Shin
2	Casey Jones
3	Bull Fighter
4	Passion You

primary_address_id	address	p_id
11	11 dreamland	1
12	5 king road	2
13	67 nichole st	3
14	32 Washington st	4

Relation un à plusieurs

```
/* Create departments table */
CREATE TABLE departments (
    department_id int(11) NOT NULL AUTO_INCREMENT,
    dname varchar(255) NOT NULL,
    PRIMARY KEY (department_id)
) ENGINE=InnoDB;
```

```
/* Create "employees" table with FOREIGN KEY */
CREATE TABLE employees (
    employee_id int(11) NOT NULL AUTO_INCREMENT,
    ename varchar(255) NOT NULL,
    d_id int(11) NOT NULL,
    salary decimal(7,2) NOT NULL,
    PRIMARY KEY (employee_id),
    FOREIGN KEY (d_id) REFERENCES departments (department_id)
) ENGINE=InnoDB;
```

department_id	dname
1	Engineering
2	Sales
3	Marketing
4	HR

employee_id	ename	d_id	salary
1	jack	1	3000.00
2	mary	2	2500.00
3	nichole	1	4000.00
4	angie	2	5000.00
5	jones	3	5000.00

Relation plusieurs à plusieurs

```
/* Create student table */  
CREATE TABLE student (  
    student_id INT NOT NULL AUTO_INCREMENT,  
    sname varchar(255) NOT NULL,  
    PRIMARY KEY (student_id)  
) ENGINE=InnoDB;
```

```
/* Create course table */  
CREATE TABLE course (  
    course_id INT NOT NULL AUTO_INCREMENT,  
    cname varchar(255) NOT NULL,  
    PRIMARY KEY (course_id)  
) ENGINE=InnoDB;
```

```
/* Create "student_course" join table with FOREIGN KEY to
 * both student and course tables. */
CREATE TABLE student_course (
    student_course_id INT NOT NULL AUTO_INCREMENT,
    s_id INT NOT NULL,
    c_id INT NOT NULL,
    PRIMARY KEY (student_course_id),
    FOREIGN KEY (s_id) REFERENCES student (student_id),
    FOREIGN KEY (c_id) REFERENCES course (course_id)
) ENGINE=InnoDB;
```

course_id	cname
11	Computer Science 101
22	MySQL
33	Java programming

3 rows in set (0.00 sec)

student_id	sname
1	jack
2	mary
3	nichole
4	mike

4 rows in set (0.00 sec)

student_course_id	s_id	c_id
1	1	11
2	1	22
3	3	22
4	4	22

Suppression d'une table

```
drop table COMMANDE;
```

- la table ne doit plus être référencée par une clé étrangère; la table DETAIL (ou sa clé étrangère vers COMMANDE) doit avoir été supprimée

La gestion des index

Les index

- Lors de la recherche d'informations dans une relation, MySQL parcours la table correspondante dans n'importe quel ordre. Dans le cas d'un grand nombre de lignes, cette recherche est très longue du fait du parcours de TOUTE la table.
- Pour y remédier, une optimisation possible et FORTEMENT recommandée, est d'utiliser des **index**.
- La création d'un index associé à un attribut ou à un ensemble ordonné d'attributs va créer une liste ordonnée des valeurs de ces attributs et de l'adresse de la ligne associée. C'est sur les valeurs de cette liste que se fera les recherches et les tris. Les algorithmes de recherche et de tri sur des ensembles ordonnées sont énormément plus rapides !
- Ainsi, d'une recherche à coût prohibitif, on passe à une recherche sur un ensemble déjà trié. On gagne donc énormément en temps d'accès aux informations. Bien que cela ralentisse les mises à jour (insertion, suppression, modification de clé).
- On choisira de créer des index sur les attributs qui seront les plus sollicités par les recherches ou utilisés comme critère de jointure. Par contre, on épargnera les attributs qui contiennent peu de valeurs différentes les unes des autres et ceux dont les valeurs sont très fréquemment modifiées.

Exemple

Id	Id	Espèce	Sexe	Date de naissance	Nom	Commentaires
1	2	chat	NULL	2010-03-24 02:23:00	Roucky	NULL
2	1	chien	male	2010-04-05 13:43:00	Rox	Mordille beaucoup
3	3	chat	femelle	2010-09-13 15:02:00	Schtroumpfette	NULL
4	6	tortue	femelle	2009-06-13 08:17:00	Bobosse	Carapace bizarre
5	9	tortue	NULL	2010-08-23 05:18:00	NULL	NULL
6	4	tortue	femelle	2009-08-03 05:12:00	NULL	NULL
7	7	chien	femelle	2008-12-06 05:18:00	Caroline	NULL
8	8	chat	male	2008-09-11 15:38:00	Bagherra	NULL
9	5	chat	NULL	2010-10-03 16:44:00	Choupi	Né sans oreille gauche

Exemple avec deux index

The diagram illustrates two indexing strategies on a table of pet information. The table has columns: Id, Id, Espèce, Sexe, Date de naissance, Nom, and Commentaires.

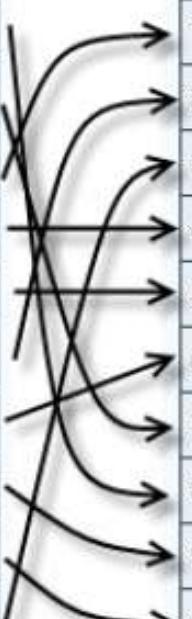
Index 1 (Left): Maps row ID to animal ID. Arrows point from row 1 to 2, 2 to 1, 3 to 3, 4 to 6, 5 to 9, 6 to 4, 7 to 7, 8 to 8, and 9 to 5.

Index 2 (Right): Maps birth date to row ID. Arrows point from 2008-09-11 to 8, 2008-12-06 to 2, 2009-06-13 to 3, 2009-08-03 to 4, 2010-03-24 to 1, 2010-04-05 to 2, 2010-08-23 to 5, 2010-08-23 to 9, 2010-09-13 to 3, 2010-10-03 to 5, and 2010-10-03 to 9.

Id	Id	Espèce	Sexe	Date de naissance	Nom	Commentaires	Date de naissance
1	2	chat	NULL	2010-03-24 02:23:00	Rouky	NULL	2008-09-11 15:38:00
2	1	chien	male	2010-04-05 13:43:00	Rox	Mordille beaucoup	2008-12-06 05:18:00
3	3	chat	femelle	2010-09-13 15:02:00	Schtroumpfette	NULL	2009-06-13 08:17:00
4	6	tortue	femelle	2009-06-13 08:17:00	Bobosse	Carapace bizarre	2009-08-03 05:12:00
5	9	tortue	NULL	2010-08-23 05:18:00	NULL	NULL	2010-03-24 02:23:00
6	4	tortue	femelle	2009-08-03 05:12:00	NULL	NULL	2010-04-05 13:43:00
7	7	chien	femelle	2008-12-06 05:18:00	Caroline	NULL	2010-08-23 05:18:00
8	8	chat	male	2008-09-11 15:38:00	Bagherra	NULL	2010-09-13 15:02:00
9	5	chat	NULL	2010-10-03 16:44:00	Choupi	Né sans oreille gauche	2010-10-03 16:44:00

Exemple avec un index sur plusieurs colonnes

nom	prenom	init_2e_prenom		id	nom	prenom	init_2e_prenom	email
Boulian	Gérard	M		1	Dupont	Charles	T	charles.dupont@email.com
Caramou	Arthur	B		2	François	Damien	V	fdamien@email.com
Dupont	Charles	T		3	Vandenbush	Guillaume	A	guillaumevdb@email.com
Dupont	Valérie	C		4	Dupont	Valérie	C	valdup@email.com
Dupont	Valérie	G		5	Dupont	Valérie	G	dupont.valerie@email.com
François	Damien	V		6	François	Martin	D	mdmartin@email.com
François	Martin	D		7	Caramou	Arthur	B	leroiarthur@email.com
Loupiot	Laura	F		8	Boulian	Gérard	M	gebou@email.com
Sunna	Christine	I		9	Loupiot	Laura	F	loulau@email.com
Vandenbush	Guillaume	A		10	Sunna	Christine	I	chrichrisun@email.com



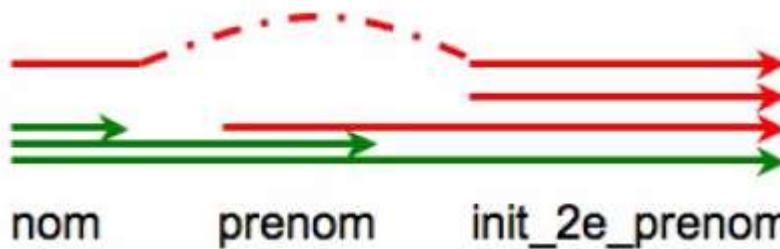
Tirer parti des index par la gauche

nom
Arthur
Charles
Christine
Damien
Gérard
Guillaume
Laura
Martin
Valérie
Valérie

nom
Boulian
Caramou
Dupont
Dupont
Dupont
François
François
Loupiot
Sunna
Vandenbush

nom	prenom
Boulian	Gérard
Caramou	Arthur
Dupont	Charles
Dupont	Valérie
Dupont	Valérie
François	Damien
François	Martin
Loupiot	Laura
Sunna	Christine
Vandenbush	Guillaume

nom	prenom	init_2e_prenom
Boulian	Gérard	M
Caramou	Arthur	B
Dupont	Charles	T
Dupont	Valérie	C
Dupont	Valérie	G
François	Damien	V
François	Martin	D
Loupiot	Laura	F
Sunna	Christine	I
Vandenbush	Guillaume	A



Les index

CLIENT
NCLI
NOM
ADRESSE
LOCALITE
CAT[0-1]
COMPTE
id: NCLI
acc
acc: LOCALITE

```
create unique index XCLI_NCLI on CLIENT(NCLI) ;
```

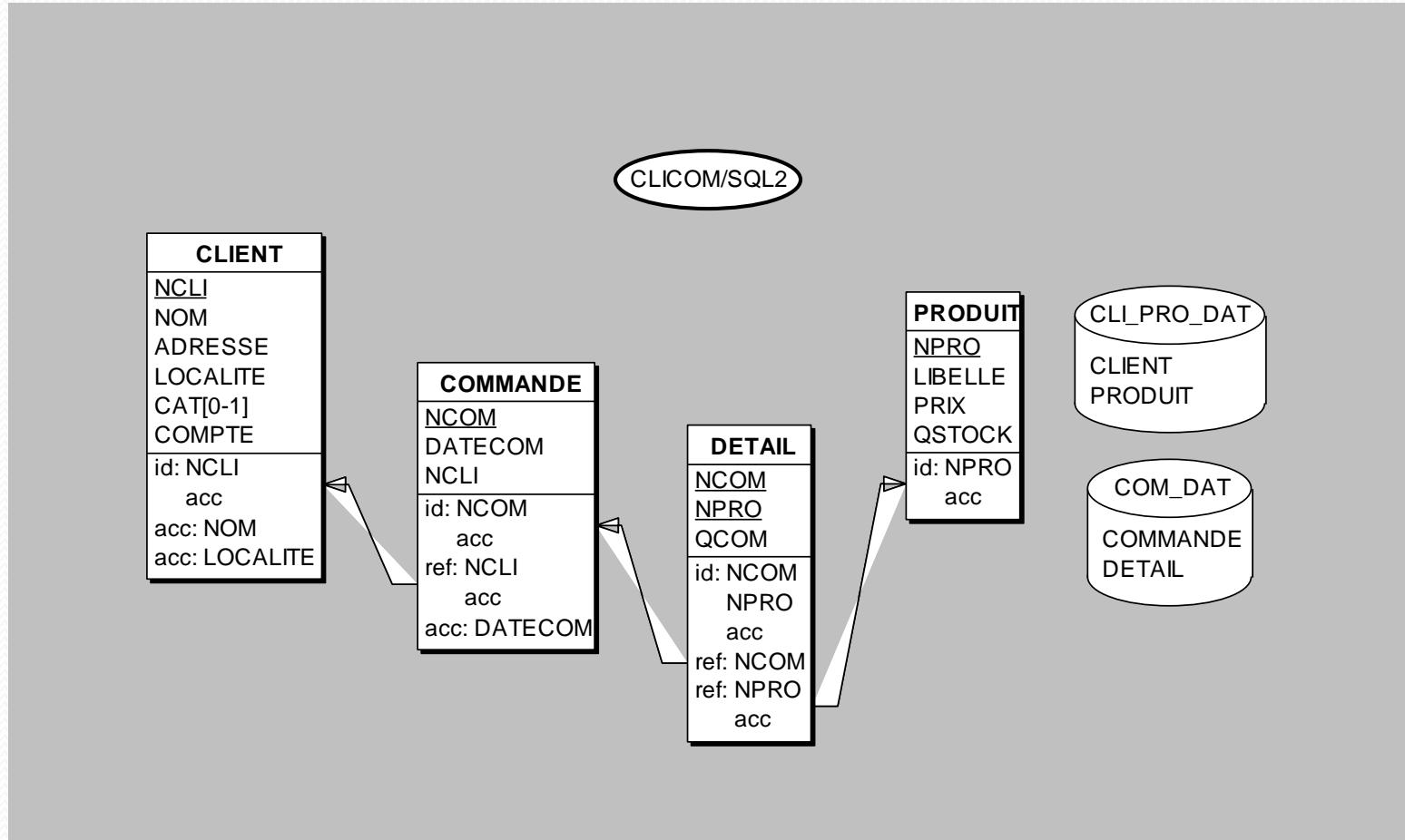
```
create index XCLI_LOC on CLIENT (LOCALITE) ;
```

```
create index XCLI_LOC on CLIENT (LOCALITE asc) ;
```

si index ordonné

```
drop index XCLI_LOC;
```

Exemple



```
create table CLIENT ( NCLI      char(10) not null,
                      NOM       char(32) not null,
                      ADRESSE   char(60) not null,
                      LOCALITE  char(30) not null,
                      CAT       char(2),
                      COMpte    decimal(9,2) not null,
                      primary key (NCLI) ) in CLI_PRO_DAT;

create table PRODUIT ( NPRO      char(15) not null,
                        LIBELLE   char(60) not null,
                        PRIX      decimal(6) not null,
                        QSTOCK    decimal(8) not null,
                        primary key (NPRO) ) in CLI_PRO_DAT;

create table COMMANDE (NCOM      char(12) not null,
                        NCLI      char(10) not null,
                        DATECOM   date not null,
                        primary key (NCOM),
                        foreign key (NCLI) references CLIENT) in
COM_DAT;

create table DETAIL (  NCOM      char(12) not null,
                        NPRO      char(15) not null,
                        QCOM      decimal(8) not null,
                        primary key (NCOM,NPRO),
                        foreign key (NCOM) references COMMANDE,
                        foreign key (NPRO) references PRODUIT) in
COM_DAT;
```

```
create index CLINCLI on CLIENT (NCLI) ;  
  
create index CLINOM on CLIENT (NOM) ;  
  
create index CLILOC on CLIENT (LOCALITE) ;  
  
create index COMNCOM on COMMANDE (NCOM) ;  
  
create index COMNOM on COMMANDE (NCLI) ;  
  
create index PRONPRO on PRODUIT (NPRO) ;  
  
create index DETCOMPRO on DETAILE (NCOM,NPRO) ;  
  
create index DETPRO on DETAILE (NPRO) ;
```

Manipulation des données

Les instructions

- l'insertion d'enregistrements : INSERT
- la modification de données : UPDATE
- la suppression d'enregistrements : DELETE (et TRUNCATE)

Insertions d'enregistrements (insert)

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] [nomBase.] { nomTable | nomVue } [(nomColonne,...)]
VALUES ({expression | DEFAULT},...),(...),...
[ON DUPLICATE KEY UPDATE nomColonne = expression,...]
```

Deux tables reliées à créer

Pilote

brevet	nom	nbHVol	compa



Compagnie

comp	nrue	rue	ville	nomComp

Création des tables

Tables

```
CREATE TABLE Compagnie
(comp CHAR(4), nrue INTEGER(3),
rue CHAR(20), ville CHAR(15) DEFAULT 'Paris'
COMMENT 'Par defaut : Paris',
nomComp CHAR(15) NOT NULL,
CONSTRAINT pk_Compagnie PRIMARY KEY(comp));
```

```
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));
```

Contraintes

Deux contraintes en ligne
et une contrainte nommée
de clé primaire.

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

Exemples d'insertions

- Renseigner toutes les colonnes

Instruction SQL	Commentaires
<pre>INSERT INTO Compagnie VALUES ('SING', 7, 'Camparols', 'Singapour', 'Singapore AL');</pre>	Toutes les valeurs sont renseignées dans l'ordre de la structure de la table.
<pre>INSERT INTO Compagnie VALUES ('AC', 10, 'Gambetta', DEFAULT, 'Air France');</pre>	DEFAULT explicite.
<pre>INSERT INTO Compagnie VALUES ('AN1', NULL, 'Hoche', 'Blagnac', 'Air Null');</pre>	NULL explicite.

Exemples d'insertions

- Renseigner certaines colonnes

Instruction SQL	Commentaires
INSERT INTO Compagnie(comp, nrule, rue, nomComp) VALUES ('AF', 8, 'Champs Elysées', 'Castanet Air');	DEFAULT implicite.
INSERT INTO Compagnie(comp, rue, ville, nomComp) VALUES ('AN2', 'Foch', 'Blagnac', 'Air Nul2');	NULL sur nrule implicite.

Compagnie	Valeur NULL	Valeur par défaut
SING	7	Camparols
AF	10	Gambetta
AN1		Hoche
AC	8	Champs Elysées
AN2		Foch

Insertions de plusieurs enregistrements

```
INSERT INTO Compagnie VALUES  
('LUFT', 9, 'Salas', 'Munich', 'Luftansa'),  
('QUAN', 1, 'Kangouroo', 'Sydney', 'Quantas'),  
('SNCM', 3, 'P. Paoli', 'Bastia', 'Corse Air');
```

Erreurs d'insertions

Insertions vérifiant les contraintes

```
INSERT INTO Pilote VALUES  
('PL-1', 'Louise Ente', 450, 'AF');
```

```
INSERT INTO Pilote VALUES  
('PL-2', 'Jules Ente', 900, 'AF');
```

```
INSERT INTO Pilote VALUES  
('PL-3', 'Paul Soutou', 1000, 'SING');
```

Insertions ne vérifiant pas les contraintes

```
mysql> INSERT INTO Pilote VALUES('PL-1',  
'Amélie Sulpice', 100, 'AF');  
ERROR 1062 (23000): Duplicate entry 'PL-1' for  
key 1
```

```
mysql> INSERT INTO Pilote VALUES ('PL-4',  
'Louise Ente', 450, 'AF');  
ERROR 1062 (23000): Duplicate entry 'Louise  
Ente' for key 2
```

```
mysql> INSERT INTO Pilote VALUES ('PL-5',  
'Thomas Sulpice', 500, 'TOTO');  
ERROR 1452 (23000): Cannot add or update a  
child row: a foreign key constraint fails  
(`bdscoutou/pilote`, CONSTRAINT  
`fk_Pil_compa_Comp` FOREIGN KEY (`compa`)  
REFERENCES `compagnie` (`comp`))  
mysql> INSERT INTO Pilote VALUES ('PL-6',  
NULL, 100, 'AF');  
ERROR 1048 (23000): Column 'nom' cannot be null
```

Le type ENUM

- Un attribut de type ENUM (non ANSI) peut prendre des valeurs parmi celles définies lors de la création de la table. Ces valeurs sont exclusivement des chaînes de caractères (insensibles à la casse) et peuvent être au maximum au nombre de 65535.
 - Nom_attr ENUM ("valeur 1", "valeur 2", ...)
- A chaque valeur est associée un index allant de 0 pour "" à N si N valeurs ont été définies.
- Si une sélection est faite dans un contexte numérique c'est l'index qui est renvoyé sinon c'est la valeur

Le type ENUM

Création

```
CREATE TABLE UnCursus  
  (num CHAR(4), nom CHAR(15), diplome  
  ENUM ('BTS', 'DUT', 'Licence', 'INSA'),  
  CONSTRAINT pk_Cusus PRIMARY KEY (num));
```

Insertions (deux bonnes une illicite)

```
mysql> INSERT INTO UnCursus VALUES  
  ('E1', 'F. Brouard', ('BTS'));  
mysql> INSERT INTO UnCursus VALUES  
  ('E2', 'F. Degrelle', 'Licence');  
  
mysql> INSERT INTO UnCursus VALUES  
  ('E3', 'Bug', ('MathSup'));  
ERROR 1265 (01000): Data truncated  
for column 'diplome' at row 1
```

UnCursus

num	nom	{diplome }
E1	F. Brouard	BTS
E2	F. Degrelle	Licence



ENUM
BTS, DUT, Licence INSA

Le type SET

- Un attribut de type SET (non ANSI) peut prendre pour valeur une chaîne vide, NULL ou une chaîne contenant une liste de valeurs qui doivent être déclarées au moment de la définition de l'attribut.
- Par exemple, un attribut déclaré comme ici :
 - SET ("voiture","avion","train") NOT NULLPeut prendre les valeurs suivantes :
""
"voiture,avion"
"train"
Mais pas NULL ni "TGV"

On ne peut définir que 64 éléments au maximum

Le type SET

Création

```
CREATE TABLE Cursus  
(num CHAR(4), nom CHAR(15), diplomes  
SET ('BTS', 'DUT', 'Licence', 'INSA'),  
CONSTRAINT pk_Cursus PRIMARY KEY(num));
```

Insertions (deux bonnes une illicite)

```
mysql> INSERT INTO Cursus VALUES ('E1',  
'F. Brouard', ('BTS', 'Licence'));  
mysql> INSERT INTO Cursus VALUES ('E2',  
'F. Degrelle', 'Licence, INSA, DUT');  
  
mysql> INSERT INTO Cursus VALUES ('E3',  
'Bug', ('BTS, INSA, ENAC'));  
ERROR 1265 (01000): Data truncated for  
column 'diplomes' at row 1
```

Cursus

num	nom	{diplomes}
E1	F. Brouard	BTS, Licence
E2	F. Degrelle	Licence, INSA, DUT



SET
BTS, DUT, Licence INSA

Gestion d'une séquence

Affreter

numAff	comp	immat	dateAff	nbPax
1	AF	F-WTSS	2005-05-13	85
2	SING	F-GAFU	2005-02-05	155
3	AF	F-WTSS	2005-09-11	90
4	AF	F-GLFS	2005-09-11	75

LAST_INSERT_ID() → 4

AUTO_INCREMENT

Table

Insertions

```
CREATE TABLE Affreter
(numAff SMALLINT AUTO_INCREMENT,
comp CHAR(4), immat CHAR(6),
dateAff DATE, nbPax SMALLINT(3),
CONSTRAINT pk_Affreter
PRIMARY KEY (numAff));
```

```
INSERT INTO Affreter
(comp, immat, dateAff, nbPax)
VALUES ('AF', 'F-WTSS', '2005-05-13', 85);
```

```
INSERT INTO Affreter
(comp, immat, dateAff, nbPax)
VALUES ('SING', 'F-GAFU', '2005-02-05', 155);
```

```
INSERT INTO Affreter
VALUES (NULL, 'AF', 'F-WTSS', '2005-09-11', 90);
```

```
INSERT INTO Affreter
VALUES (0, 'AF', 'F-GLFS', '2005-09-11', 75);
```

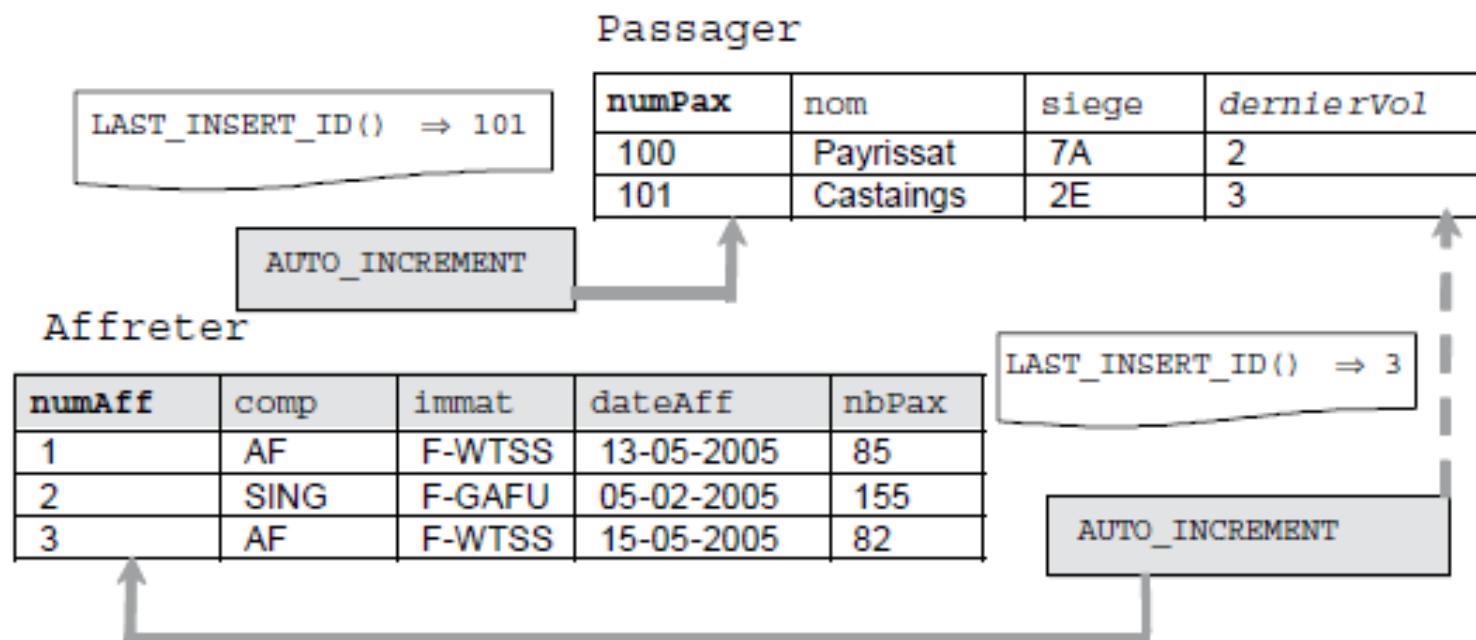
Modification d'une séquence

- La seule modification possible d'une séquence est celle qui consiste à changer la valeur de départ de la séquence (avec ALTER TABLE)

```
ALTER TABLE Affreter AUTO_INCREMENT = 100;
INSERT INTO Affreter (comp, immat, dateAff, nbPax)
VALUES ('SING', 'F-NEW', SYSDATE(), 77);
mysql> SELECT * FROM Affreter ;
```

numAff	comp	immat	dateAff	nbPax
1	AF	F-WTSS	2005-05-13	85
2	SING	F-GAFU	2005-02-05	155
3	AF	F-WTSS	2005-09-11	90
4	AF	F-GLFS	2005-09-11	75
100	SING	F-NEW	2005-11-01	77

Utilisation en tant que clé étrangère



Séquence pour une clé étrangère

Tables

```
CREATE TABLE Affreter  
(numAff SMALLINT AUTO_INCREMENT,  
comp CHAR(4), immat CHAR(6),  
dateAff DATE, nbPax SMALLINT(3),  
CONSTRAINT pk_Affreter  
    PRIMARY KEY (numAff));
```

```
CREATE TABLE Passager  
(numPax SMALLINT AUTO_INCREMENT,  
nom CHAR(15), siege CHAR(4),  
dernierVol SMALLINT,  
CONSTRAINT pk_Passager  
    PRIMARY KEY(numPax),  
CONSTRAINT fk_Pax_vol_Affreter  
    FOREIGN KEY (dernierVol)  
        REFERENCES Affreter(numAff))  
AUTO_INCREMENT = 100;
```

Insertions

```
INSERT INTO Affreter  
(comp, immat, dateAff, nbPax) VALUES  
('AF', 'F-WTSS', '2005-05-13', 85);
```

```
INSERT INTO Affreter  
(comp, immat, dateAff, nbPax) VALUES  
('SING', 'F-GAFU', '2005-02-05', 155);
```

```
INSERT INTO Passager VALUES  
(NULL, 'Payrissat', '7A', LAST_INSERT_ID());
```

```
INSERT INTO Affreter VALUES  
(NULL, 'AF', 'F-WTSS', '2005-05-15', 82);
```

```
INSERT INTO Passager VALUES  
(NULL, 'Castaings', '2E', LAST_INSERT_ID());
```

Remplacement d'un enregistrement

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] [nomBase.] nomTable [(colonne1,...)]
VALUES ({expression1 | DEFAULT},...) [,(...),...]
```



```
REPLACE INTO Compagnie VALUES ('AN1', 24, 'Salas', 'Ramonville',
'Air RENATO');
```

Supprimer un enregistrement

- Attention, la suppression est définitive !
- Syntaxe :
 - **DELETE [LOW_PRIORITY] FROM *relation* [WHERE *condition*] [LIMIT *a*]**
- Exemple :
 - **DELETE FROM *Personnes* WHERE nom='Martin' AND prénom='Marc'**
- Pour vider une table de tous ces éléments, ne pas mettre de clause WHERE. Cela efface et recrée la table, au lieu de supprimer un à un chacun des tuples de la table (ce qui serait plus long).
- Exemple :
 - **DELETE FROM *Personnes***

Suppression de lignes

```
delete from CLIENT  
where NCLI = 'K111';
```

```
delete from DETAIL  
where NPRO in (select NPRO  
from PRODUIT  
where QSTOCK <= 0);
```

```
delete from CLIENT  
where CAT is null;
```

Modifier un enregistrement

- Pour modifier un ou des enregistrement(s) d'une relation, il faut préciser un critère de sélection des enregistrement à modifier (clause **WHERE**), il faut aussi dire quels sont les attributs dont on va modifier la valeur et quelles sont ces nouvelles valeurs (clause **SET**).
- Syntaxe :
- **UPDATE [LOW_PRORITY] relation SET attribut=valeur, ... [WHERE condition] [LIMIT a]**
- Exemple :
- **UPDATE TABLE Personnes SET téléphone='0156281469' WHERE nom='Martin' AND prénom = 'Pierre'**
- Cet exemple modifie le numéro de téléphone de Martin Pierre.
- **LOW_PRORITY** est une option un peu spéciale qui permet de n'appliquer la ou les modification(s) qu'une fois que plus personne n'est en train de lire dans la relation.

Modifier un enregistrement

- Il est possible de modifier les valeurs d'autant d'attributs que la relation en contient.
- Exemple pour modifier plusieurs attributs :
UPDATE TABLE Personnes SET téléphone='0156281469', fax='0156281812' WHERE id = 102
- Pour appliquer la modification à tous les enregistrements de la relation, il suffit de ne pas mettre de clause WHERE.
- **LIMIT a** permet de n'appliquer la commande qu'aux **a** premiers enregistrements satisfaisant la condition définie par WHERE.
- Autre exemple :
 - **UPDATE TABLE Enfants SET age=age+1**
- Il est donc possible de modifier la valeur d'un attribut relativement à sa valeur déjà existante.

Modification d'enregistrements

```
update CLIENT
set ADRESSE = '29, av. de la Magne',
      LOCALITE = 'Niort'
where NCLI = 'F011';
```

```
update PRODUIT
set PRIX = PRIX * 1.05
where LIBELLE like '%SAPIN%';
```

Ne pas respecter les contraintes

Données

Figure 2-7 Données Pilote

brevet	nom	nbHVol	compa
PL-1	Louise Ente	450	AF
PL-2	Jules Ente	900	AF
PL-3	Paul Soutou	1000	SING

Table et contraintes

```
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));
```

Ne pas respecter les contraintes

Vérifiant les contraintes (sauf une !)

```
--Modification d'une clé étrangère  
UPDATE Pilote SET compa = 'SING'  
WHERE brevet = 'PL-2';
```

```
-- Modification d'une clé primaire  
UPDATE Pilote SET brevet = 'PL3bis'  
WHERE brevet = 'PL-3';
```

```
--Passe outre la contrainte CHECK !  
UPDATE Pilote SET nbHVol= 30000  
WHERE brevet = 'PL-1';
```

Figure 2-8 Après modifications

Pilote

brevet	nom	nbHVol	compa
PL-1	Louise Ente	30000	AF
PL-2	Jules Ente	900	SING
PL3bis	Paul Soutou	1000	SING

Ne vérifiant pas les contraintes

```
mysql> UPDATE Pilote SET brevet='PL-2'  
WHERE brevet='PL-1';  
ERROR 1062 (23000): Duplicate entry  
'PL-2' for key 1
```

```
mysql> UPDATE Pilote SET nom = NULL  
WHERE brevet = 'PL-1';  
ERROR 1263 (22004): Column set to  
default value; NULL supplied to NOT  
NULL column 'nom' at row 1
```

```
mysql> UPDATE Pilote SET nom='Paul  
Soutou' WHERE brevet = 'PL-1';  
ERROR 1062 (23000): Duplicate entry  
'Paul Soutou' for key 2
```

```
mysql> UPDATE Pilote SET compa='TOTO'  
WHERE brevet = 'PL-1';  
ERROR 1452 (23000): Cannot add or  
update a child row: a foreign key  
constraint fails ('bdsoutou/pilote',  
CONSTRAINT `fk_Pil_compa_Comp` FOREIGN  
KEY (`compa`) REFERENCES `compagnie`  
(`comp`))
```

Evolution d'un schéma

Modifier une relation

- La création d'une relation par **CREATE TABLE** n'en rend pas définitives les spécifications. Il est possible d'en modifier la définition par la suite, à tout moment par la commande **ALTER TABLE**.
- Voici ce qu'il est possible de réaliser :
 - - ajouter/supprimer un attribut
 - - créer/supprimer une clé primaire
 - - ajouter une contrainte d'unicité (interdire les doublons)
 - - changer la valeur par défaut d'un attribut
 - - changer totalement la définition d'un attribut
 - - changer le nom de la relation
 - - ajouter/supprimer un index

Ajout, retrait, modification d'une colonne

```
alter table PRODUIT add column POIDS smallint;
```

ajouter

```
alter table PRODUIT drop column PRIX;
```

supprimer

```
alter table CLIENT modify column CAT set '00';
```

modifier valeur
par défaut

```
alter table CLIENT modify column CAT drop default;
```

supprimer valeur
par défaut

Activation/désactivation de contraintes

```
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}]
```

Ajout et retrait d'une contrainte

```
alter table PROSPECT add primary key (NCLI);
```

```
alter table CLIENT add unique (NOM,ADRESSE,LOCALITE);
```

```
alter table CLIENT modify CAT not null;  
alter table CLIENT modify ADRESSE null;
```

```
alter table CLIENT  
add foreign key (CAT) references CATEGORIE ;
```

Variantes : contraintes nommées

```
alter table PROSPECT  
add constraint PROPK primary key (NCLI);
```

```
alter table CLIENT  
add constraint CLIUN unique (NOM,ADRESSE,LOCALITE);
```

```
alter table CLIENT  
add constraint CLIFK1 foreign key (CAT)  
references CATEGORIE ;
```

```
alter table PROSPECT  
drop constraint PROPK;
```

Ajouter un attribut

- Syntaxe :
ALTER TABLE *relation* ADD *definition* [FIRST | AFTER *attribut*]
- Ajoutons l'attribut *fax* qui est une chaîne représentant un nombre de 10 chiffres:
ALTER TABLE *Personnes* ADD *fax* DECIMAL(10,0)
- Nous aurions pu forcer la place où doit apparaître cet attribut. Pour le mettre en tête de la liste des attributs de la relation, il faut ajouter l'option **FIRST** en fin de commande. Pour le mettre après l'attribut '*téléphone*', il aurait fallu ajouter **AFTER 'téléphone'**.
- Note : il ne doit pas déjà y avoir dans la relation un attribut du même nom !

Supprimer un attribut

- Attention, supprimer un attribut implique la suppression des valeurs qui se trouvent dans la colonne qui correspond à cet attribut

- Syntaxe :

ALTER TABLE *relation* DROP *attribut*

- Exemple :

ALTER TABLE *Personnes* DROP *prénom*

Supprimer un attribut

- La suppression d'un attribut peut incidemment provoquer des erreurs sur les contraintes clé primaire (**PRIMARY KEY**) et unique (**UNIQUE**).

`CREATE TABLE Personne (`

```
    id SMALLINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,  
    nom VARCHAR(40),  
    prenom VARCHAR(40),  
    adresse TINYTEXT,  
    telephone DECIMAL(10,0),  
    UNIQUE(nom,prenom )
```

`ALTER TABLE Personnes DROP prenom`

<i>nom</i>	<i>prénom</i>
Dupond	Marc
Martin	Marc
Martin	Pierre

<i>nom</i>
Dupond
Martin
Martin

Refus d'opérer la suppression, car cela contredirait la contrainte d'unicité qui resterait sur l'attribut *nom*.

Créer une clé primaire

- La création d'une clé primaire n'est possible qu'en l'absence de clé primaire dans la relation.
- Syntaxe :
**ALTER TABLE *relation* ADD PRIMARY KEY
(*attribut*)**
- Exemple :
**ALTER TABLE *Personnes* ADD PRIMARY KEY
(*nom*)**

Supprimer une clé primaire

- Comme une clé primaire est unique, il n'y a aucune ambiguïté lors de la suppression.
- Syntaxe :
ALTER TABLE *relation* DROP PRIMARY KEY
- Exemple :
ALTER TABLE *Personnes* ADD PRIMARY KEY
- S'il n'y a aucune clé primaire lorsque cette commande est exécutée, aucun message d'erreur ne sera généré, la commande sera simplement ignorée.

Ajout d'une contrainte d'unicité

- Il est possible (facultatif) de donner un nom à la contrainte.
- Cette contrainte peut s'appliquer à plusieurs attributs.
- Si les valeurs déjà présentes dans la relation sont en contradiction avec cette nouvelle contrainte, alors cette dernière ne sera pas appliquée et une erreur sera générée.
- Syntaxe :
ALTER TABLE *relation* ADD UNIQUE [*contrainte*] (*attributs*)
- Exemple pour interdire tout doublon sur l'attribut *fax* de la relation **Personnes** :
ALTER TABLE *Personnes* ADD UNIQUE *u_fax* (*fax*)
- Autre exemple fictif :
ALTER TABLE *Moto* ADD UNIQUE *u_coul_vitre* (*couleur,vitre*)

Changer la valeur par défaut d'un attribut

- Pour changer ou supprimer la valeur par défaut d'un attribut.
- Attention aux types qui n'acceptent pas de valeur par défaut (les familles BLOB et TEXT).
- Syntaxe :
ALTER TABLE *relation* ALTER *attribut* { SET DEFAULT *valeur* | DROP DEFAULT }
- Changer sa valeur par défaut :
ALTER TABLE *Personnes* ALTER *téléphone* SET DEFAULT '9999999999'
- Supprimer sa valeur par défaut :
ALTER TABLE *Personnes* ALTER *téléphone* DROP DEFAULT
- Le changement ou la suppression n'affecte en rien les enregistrements qui ont eu recours à cette valeur lors de leur insertion.

Changer la définition d'un attribut

- Pour changer la définition de l'attribut sans le renommer :

ALTER TABLE *relation* MODIFY *attribut definition_relative*

- Exemple 1 :

ALTER TABLE *Personnes* MODIFY *fax* VARCHAR(14)

- Pour changer sa définition en le renommant :

ALTER TABLE *relation* CHANGE *attribut definition_absolue*

- Exemple 2 :

ALTER TABLE *Personnes* CHANGE *fax num_fax* VARCHAR(14)

- Attention, si le nouveau type appliqué à l'attribut est incompatible avec les valeurs des enregistrements déjà présents dans la relation, alors elles risquent d'être modifiées ou remises à zéro !

Changer le nom de la relation

- Syntaxe :
 - **ALTER TABLE *relation* RENAME *nouveau_nom***
- Exemple :
 - **ALTER TABLE *Personnes* RENAME *Carnet***
- Cela consiste à renommer la table, et donc le fichier qui la stocke.

Ajouter un index

- Une table ne peut comporter que 32 index.
- Et un index ne peut porter que sur 16 attributs maximum à la fois.
- Syntaxe :
ALTER TABLE *relation* ADD INDEX *index* (*attributs*)
- Exemple :
ALTER TABLE *Personnes* ADD INDEX *nom_complet* (*nom,prénom*)
- Dans cet exemple, on a ajouté à la relation ***Personnes*** un index que l'on nomme *nom_complet* et qui s'applique aux deux attributs *nom* et *prénom*. Ainsi, les recherches et les tris sur les attributs *nom* et *prénom* seront grandement améliorés car un index apporte les changements sous-jacents permettant d'optimiser les performances du serveur de base de données.

Supprimer un index

- Syntaxe :
 - **ALTER TABLE *relation* DROP INDEX *index***
- Exemple :
 - **ALTER TABLE *Personnes* DROP INDEX *nom_complet***
- Cet exemple permet de supprimer l'index nommé *nom_complet* de la relation ***Personnes***.

L'intégrité référencielle

Intégrité référentielle

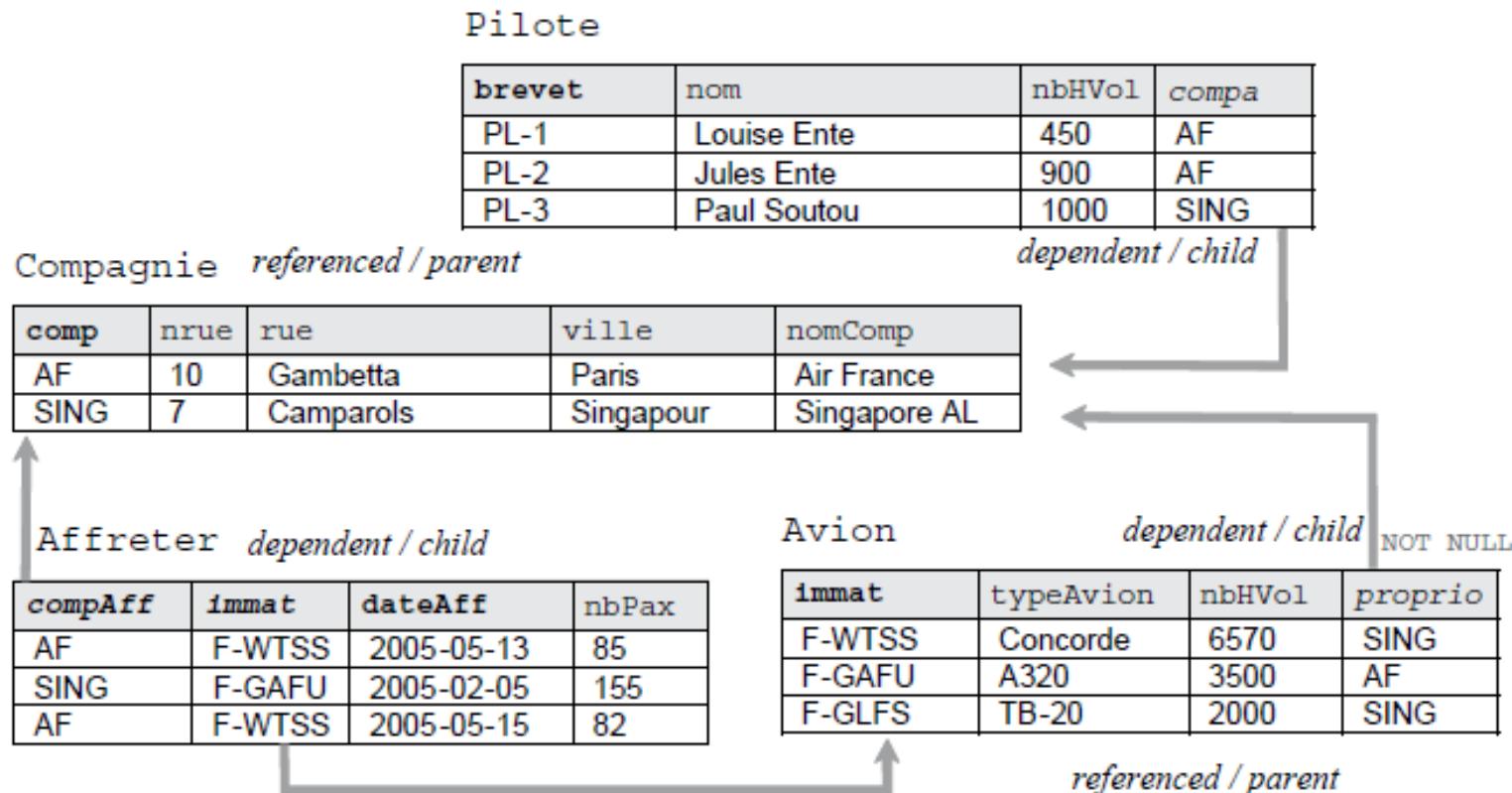
- L'intégrité référentielle forme le cœur de la cohérence d'une base de données relationnelle
- Cette intégrité est fondée sur la relation entre clés étrangères et clés primaires
- La contrainte référentielle concerne toujours deux tables possédant une ou plusieurs colonnes en commun.
 - une table « père » aussi dite « maître » (clé primaire)
 - une table « fils » (clé étrangère)

Intégrité référentielle (suite)

- L'intégrité référentielle se programme dans la table « fille »
- Deux types de problèmes sont résolus :
 - La cohérence du fils vers le père
 - La cohérence du père vers le fils

```
[CONSTRAINT nomContrainte] FOREIGN KEY [id] (listeColonneEnfant)
    REFERENCES nomTable (listeColonneParent)
    [ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}] ]
    [ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}] ]
```

Exemple



Ecriture des contraintes

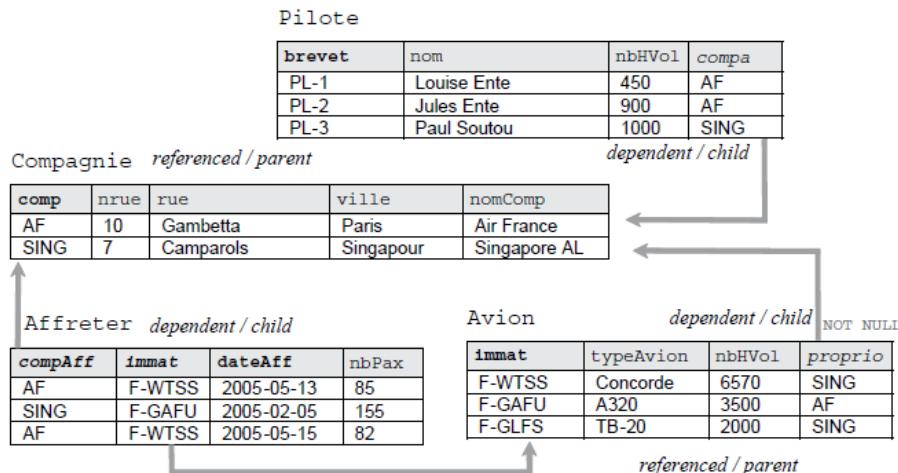
• Côté Père

Clé primaire	Clé candidate
<pre>CREATE TABLE Compagnie (comp CHAR(4), nrue INTEGER(3), rue CHAR(20), ville CHAR(15), nomComp CHAR(15), CONSTRAINT pk_Compagnie PRIMARY KEY(comp));</pre>	<pre>CREATE TABLE Compagnie (comp CHAR(4) NOT NULL, nrue INTEGER(3), rue CHAR(20), ville CHAR(15), nomComp CHAR(15), CONSTRAINT un_Compagnie UNIQUE(comp), CONSTRAINT pk_Compagnie PRIMARY KEY(nomComp));</pre>

• Côté Fils

Contrainte nommée sans index	Contrainte pas nommée et index
<pre>CREATE TABLE Pilote (brevet CHAR(6), nom CHAR(15), nbHVol DECIMAL(7,2), compa CHAR(4), CONSTRAINT pk_Pilote PRIMARY KEY(brevet), CONSTRAINT fk_Pil_compa_Comp FOREIGN KEY (compa) REFERENCES Compagnie(comp));</pre>	<pre>CREATE TABLE Pilote (brevet CHAR(6), nom CHAR(15), nbHVol DECIMAL(7,2), compa CHAR(4), CONSTRAINT pk_Pilote PRIMARY KEY(brevet), INDEX (compa), FOREIGN KEY (compa) REFERENCES Compagnie(comp));</pre>

Exemple



```
CREATE TABLE Avion
```

```

(immat CHAR(6), typeAvion CHAR(15), nbhVol DECIMAL(10,2),
proprio CHAR(4) NOT NULL, CONSTRAINT pk_Avion PRIMARY KEY(immat),
INDEX (proprio),
CONSTRAINT fk_Avion_comp_Compag
FOREIGN KEY(proprio) REFERENCES Compaqnie(comp));

```

```
CREATE TABLE Affreter
```

```

(compAff CHAR(4), immat CHAR(6), dateAff DATE, nbPax INTEGER(3),
CONSTRAINT pk_Affreter PRIMARY KEY (compAff, immat, dateAff),
INDEX (immat),
CONSTRAINT fk_Aff_na_Avion
    FOREIGN KEY(immat) REFERENCES Avion(immat),
INDEX (compAff),
CONSTRAINT fk_Aff_comp_Compag
    FOREIGN KEY(compAff) REFERENCES Compagnie(comp));

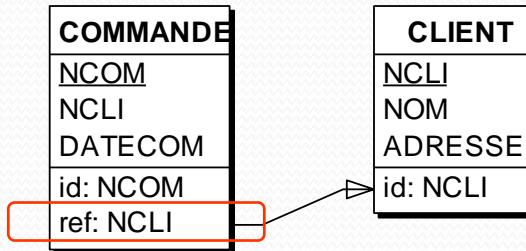
```

Cohérence du fils vers le père

- Si la clé étrangère est déclarée NOT NULL, l'insertion d'un enregistrement « fils » n'est possible que s'il est rattaché à un enregistrement « père » existant.
- Dans le cas inverse, l'insertion d'un enregistrement « fils » rattaché à aucun « père » est possible.

Insertions correctes	Insertion incorrecte
<pre>-- fils avec père INSERT INTO Pilote VALUES ('PL-3', 'Paul Soutou', 1000, 'SING'); -- fils sans père INSERT INTO Pilote VALUES ('PL-4', 'Un Connlu', 0, NULL); -- fils avec pères INSERT INTO Avion VALUES ('F-WTSS', 'Concorde', 6570, 'SING'); INSERT INTO Affreter VALUES ('AF', 'F-WTSS', '15-05-2003', 82)</pre>	<pre>-- avec père inconnu mysql> INSERT INTO Pilote VALUES ('PL-5', 'Pb de Compagnie', 0, '?');</pre> <p>ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`bdsoutou/pilote`, CONSTRAINT `fk_Pil_compa_Comp` FOREIGN KEY (`compa`) REFERENCES `compagnie` (`comp`))</p>

Modification des données - Intégrité référentielle



{NCLI} clé étrangère de **COMMANDÉ** vers **CLIENT**

insérer une ligne de **COMMANDÉ**

la valeur de **NCLI** doit être présente dans la colonne **NCLI** d'une ligne de **CLIENT**

modifier valeur de **NCLI** de **COMMANDÉ**

la nouvelle valeur de **NCLI** doit être présente dans la colonne **NCLI** d'une ligne de **CLIENT**

supprimer une ligne de **CLIENT**

l'intégrité référentielle doit être satisfaite après l'opération

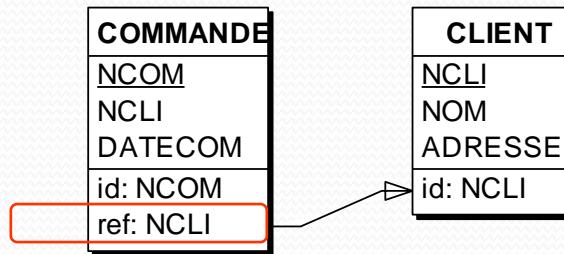
plusieurs comportements possibles

modifier valeur de **NCLI** de **CLIENT**

l'intégrité référentielle doit être satisfaite après l'opération

plusieurs comportements possibles

Modification des données - Intégrité référentielle



supprimer une ligne de CLIENT	Comportement	{NCLI} clé étrangère de COMMANDÉ vers CLIENT
	1. mode <i>no action</i>	opération refusée si lignes de COMMANDÉ dépendantes
	2. mode <i>cascade</i>	ligne supprimée mais aussi les lignes de COMMANDÉ dépendantes
	3. mode <i>set null</i>	(si NCLI de COMMANDÉ facultative) la colonne NCLI des lignes dépendantes de COMMANDÉ est mise à <i>null</i>
	4. mode <i>set default</i>	(si <i>default</i> pour NCLI de COMMANDÉ) la colonne NCLI des lignes dépendantes de COMMANDÉ est mises à <i>la valeur par défaut</i>

Mode propagation (*cascade*)

```
create table COMMANDE ( NCOM char(12) not null,  
                      NCLI char(10) not null,  
                      ...,  
                      primary key (NCOM),  
                      foreign key (NCLI) references CLIENT  
                        on delete cascade);
```

```
create table COMMANDE ( NCOM char(12) not null,  
                      NCLI char(10) not null,  
                      ...,  
                      primary key (NCOM),  
                      foreign key (NCLI) references CLIENT  
                        on update cascade);
```

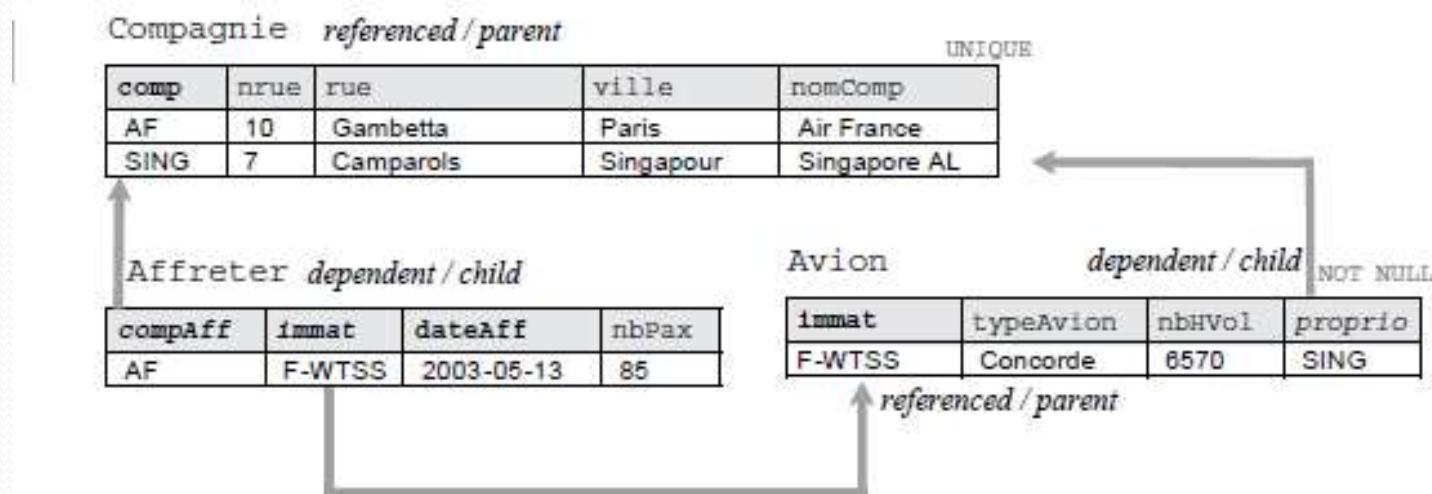
Mode bloquant (*no action*)

```
create table COMMANDE ( NCOM char(12) not null,  
                      NCLI char(10) not null,  
                      ...,  
                      primary key (NCOM),  
                      foreign key (NCLI) references CLIENT  
                            on delete no action);
```

```
create table COMMANDE ( NCOM char(12) not null,  
                      NCLI char(10) not null,  
                      ...,  
                      primary key (NCOM),  
                      foreign key (NCLI) references CLIENT  
                            on update no action);
```

Désactivation des contraintes

- La désactivation des contraintes référentielles peut être intéressante pour accélérer des procédures de chargement d'importation et d'exportation massives de données externes.
 - SET FOREIGN_KEY_CHECKS=0
- Avant la désactivation des contraintes



Après la désactivation

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)	

Synthèse de l'intégrité référentielle

Instructions	Table « père »	Table « fils »
INSERT	Correcte si la clé primaire (ou candidate) est unique.	Correcte si la clé étrangère est référencée dans la table « père » ou est nulle (partiellement ou en totalité).
UPDATE	Correcte si l'instruction ne laisse pas d'enregistrements dans la table « fils » ayant une clé étrangère non référencée.	Correcte si la nouvelle clé étrangère référence un enregistrement « père » existant.
DELETE	Correcte si aucun enregistrement de la table « fils » ne référence le ou les enregistrements détruits.	Correcte sans condition.
DELETE Cascade	Correcte sans condition.	Sans objet.
DELETE SET NULL	Correcte sans condition.	Sans objet.
UPDATE Cascade	Correcte sans condition.	Sans objet.
UPDATE SET NULL	Correcte s'il n'y a pas de NOT NULL dans la table « fils ».	Sans objet.

Exemple d'intégrité référentielle

department_id	dname
1	Engineering
2	Sales
3	Marketing
4	HR

employee_id	ename	d_id	salary
1	jack	1	3000.00
2	mary	2	2500.00
3	nichole	1	4000.00
4	angie	2	5000.00
5	jones	3	5000.00

```
mysql> INSERT INTO employees(employee_id, ename, salary, d_id)
-> VALUES (6, 'newperson', '5000.00', 10);
```

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
(`mydb`.`employees`, CONSTRAINT `employees_ibfk_1` FOREIGN KEY (`d_id`) REFERENCES
`departments` (`department id`))

Exemple pour delete et cascade

```
/* Create "employees" table with FOREIGN KEY */
CREATE TABLE employees (
    employee_id int(11) NOT NULL AUTO_INCREMENT,
    ename varchar(255) NOT NULL,
    d_id int(11) NOT NULL,
    salary decimal(7,2) NOT NULL,
    PRIMARY KEY (employee_id),
    FOREIGN KEY (d_id) REFERENCES departments (department_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
) ENGINE=InnoDB;
```

```
mysql> DELETE FROM departments WHERE department_id = 2;  
Query OK, 1 row affected (0.05 sec)
```

```
mysql> SELECT * FROM departments;
```

department_id	dname
3	Marketing
4	HR
11	Engineering

3 rows in set (0.00 sec)

// Observe that the employee record whose foreign key is 2
// are automatically deleted.

```
mysql> SELECT * FROM employees;
```

employee_id	ename	d_id	salary
1	jack	11	3000.00
3	nichole	11	4000.00
5	jones	3	5000.00

Démo Mysql

Un exemple

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| test |
| ..... |
```

Database: southwind
Table: products

productID INT	productCode CHAR(3)	name VARCHAR(30)	quantity INT	price DECIMAL(10,2)
1001	PEN	Pen Red	5000	1.23
1002	PEN	Pen Blue	8000	1.25
1003	PEN	Pen Black	2000	1.25
1004	PEC	Pencil 2B	10000	0.48
1005	PEC	Pencil 2H	8000	0.49

Création de la base de données

```
mysql> CREATE DATABASE southwind;  
Query OK, 1 row affected (0.03 sec)
```

```
mysql> DROP DATABASE southwind;  
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> CREATE DATABASE IF NOT EXISTS southwind;  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> DROP DATABASE IF EXISTS southwind;  
Query OK, 0 rows affected (0.00 sec)
```

Utilisation de la base de données

```
-- Remove the database "southwind", if it exists.  
-- Beware that DROP (and DELETE) actions are irreversible and not recoverable!  
mysql> DROP DATABASE IF EXISTS southwind;  
Query OK, 1 rows affected (0.31 sec)  
  
-- Create the database "southwind"  
mysql> CREATE DATABASE southwind;  
Query OK, 1 row affected (0.01 sec)  
  
-- Show all the databases in the server  
-- to confirm that "southwind" database has been created.  
mysql> SHOW DATABASES;  
+-----+  
| Database |  
+-----+  
| southwind |  
| ..... |  
  
-- Set "southwind" as the default database so as to reference its table directly.  
mysql> USE southwind;  
Database changed  
  
-- Show the current (default) database  
mysql> SELECT DATABASE();  
+-----+  
| DATABASE() |  
+-----+  
| southwind |  
+-----+
```

Création de la table

```
-- Show all the tables in the current database.  
-- "southwind" has no table (empty set).  
mysql> SHOW TABLES;  
Empty set (0.00 sec)  
  
-- Create the table "products". Read "explanations" below for the column definitions  
mysql> CREATE TABLE IF NOT EXISTS products (  
    productID      INT UNSIGNED  NOT NULL AUTO_INCREMENT,  
    productCode    CHAR(3)        NOT NULL DEFAULT '',  
    name           VARCHAR(30)    NOT NULL DEFAULT '',  
    quantity       INT UNSIGNED  NOT NULL DEFAULT 0,  
    price          DECIMAL(7,2)   NOT NULL DEFAULT 99999.99,  
    PRIMARY KEY    (productID)  
)  
Query OK, 0 rows affected (0.08 sec)
```

Description de la table

```
-- Show all the tables to confirm that the "products" table has been created
mysql> SHOW TABLES;
+-----+
| Tables_in_southwind |
+-----+
| products           |
+-----+


-- Describe the fields (columns) of the "products" table
mysql> DESCRIBE products;
+-----+-----+-----+-----+-----+-----+
| Field      | Type           | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| productID  | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| productCode | char(3)         | NO   |     |          |                |
| name        | varchar(30)      | NO   |     |          |                |
| quantity    | int(10) unsigned | NO   |     | 0       |                |
| price       | decimal(7,2)      | NO   |     | 99999.99 |                |
+-----+-----+-----+-----+-----+-----+
```

```
-- Show the complete CREATE TABLE statement used by MySQL to create this table
mysql> SHOW CREATE TABLE products \G
***** 1. row *****
      Table: products
Create Table:
CREATE TABLE `products` (
  `productID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `productCode` char(3) NOT NULL DEFAULT '',
  `name` varchar(30) NOT NULL DEFAULT '',
  `quantity` int(10) unsigned NOT NULL DEFAULT '0',
  `price` decimal(7,2) NOT NULL DEFAULT '99999.99',
  PRIMARY KEY (`productID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Insertions de données

```
-- Insert a row with all the column values
mysql> INSERT INTO products VALUES (1001, 'PEN', 'Pen Red', 5000, 1.23);
Query OK, 1 row affected (0.04 sec)

-- Insert multiple rows in one command
-- Inserting NULL to the auto_increment column results in max_value + 1
mysql> INSERT INTO products VALUES
    (NULL, 'PEN', 'Pen Blue', 8000, 1.25),
    (NULL, 'PEN', 'Pen Black', 2000, 1.25);
Query OK, 2 rows affected (0.03 sec)
Records: 2  Duplicates: 0  Warnings: 0

-- Insert value to selected columns
-- Missing value for the auto_increment column also results in max_value + 1
mysql> INSERT INTO products (productCode, name, quantity, price) VALUES
    ('PEC', 'Pencil 2B', 10000, 0.48),
    ('PEC', 'Pencil 2H', 8000, 0.49);
Query OK, 2 row affected (0.03 sec)

-- Missing columns get their default values
mysql> INSERT INTO products (productCode, name) VALUES ('PEC', 'Pencil HB');
Query OK, 1 row affected (0.04 sec)

-- 2nd column (productCode) is defined to be NOT NULL
mysql> INSERT INTO products values (NULL, NULL, NULL, NULL, NULL);
ERROR 1048 (23000): Column 'productCode' cannot be null
```

Contenu de la table

```
-- Query the table
```

```
mysql> SELECT * FROM products;
```

productID	productCode	name	quantity	price
1001	PEN	Pen Red	5000	1.23
1002	PEN	Pen Blue	8000	1.25
1003	PEN	Pen Black	2000	1.25
1004	PEC	Pencil 2B	10000	0.48
1005	PEC	Pencil 2H	8000	0.49
1006	PEC	Pencil HB	0	9999999.99

```
6 rows in set (0.02 sec)
```

```
-- Remove the last row
```

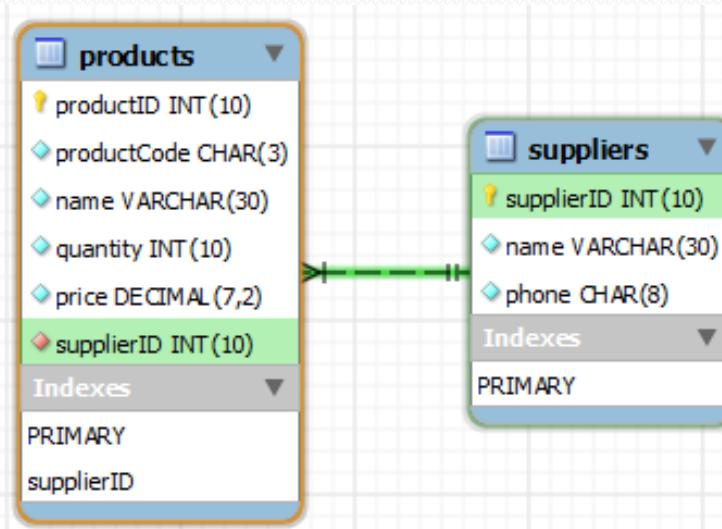
```
mysql> DELETE FROM products WHERE productID = 1006;
```

Relation un à plusieurs

Database: southwind

Table: suppliers

supplierID INT	name VARCHAR(3)	phone CHAR(8)
501	ABC Traders	88881111
502	XYZ Company	88882222
503	QQ Corp	88883333



```
mysql> USE southwind;

mysql> DROP TABLE IF EXISTS suppliers;

mysql> CREATE TABLE suppliers (
    supplierID INT UNSIGNED NOT NULL AUTO_INCREMENT,
    name        VARCHAR(30)  NOT NULL DEFAULT '',
    phone       CHAR(8)      NOT NULL DEFAULT '',
    PRIMARY KEY (supplierID)
);
```

```
mysql> DESCRIBE suppliers;
+-----+-----+-----+-----+-----+
| Field | Type           | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+
| supplierID | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| name        | varchar(30)      | NO   |     |          |                 |
| phone       | char(8)          | NO   |     |          |                 |
+-----+-----+-----+-----+-----+
```

```
mysql> INSERT INTO suppliers VALUE
        (501, 'ABC Traders', '88881111'),
        (502, 'XYZ Company', '88882222'),
        (503, 'QQ Corp', '88883333');
```

```
mysql> SELECT * FROM suppliers;
+-----+-----+-----+
| supplierID | name        | phone      |
+-----+-----+-----+
|      501 | ABC Traders | 88881111 |
|      502 | XYZ Company | 88882222 |
|      503 | QQ Corp     | 88883333 |
+-----+-----+-----+
```

Alter table

```
mysql> ALTER TABLE products  
        ADD COLUMN supplierID INT UNSIGNED NOT NULL;  
Query OK, 4 rows affected (0.13 sec)  
Records: 4  Duplicates: 0  Warnings: 0
```

```
mysql> DESCRIBE products;  
+-----+-----+-----+-----+-----+  
| Field      | Type           | Null | Key | Default | Extra          |  
+-----+-----+-----+-----+-----+  
| productID  | int(10) unsigned | NO   | PRI | NULL    | auto_increment |  
| productCode | char(3)         | NO   |     |          |                |  
| name        | varchar(30)     | NO   |     |          |                |  
| quantity    | int(10) unsigned | NO   |     | 0        |                |  
| price       | decimal(10,2)    | NO   |     | 99999999.99 |                |  
| supplierID  | int(10) unsigned | NO   |     | NULL    |                |  
+-----+-----+-----+-----+-----+
```

```
-- Set the supplierID of the existing records in "products" table to a VALID supplierID
--   of "suppliers" table
mysql> UPDATE products SET supplierID = 501;

-- Add a foreign key constrain
mysql> ALTER TABLE products
    ADD FOREIGN KEY (supplierID) REFERENCES suppliers (supplierID);

mysql> DESCRIBE products;
+-----+-----+-----+-----+-----+
| Field      | Type           | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
.....  
| supplierID | int(10) unsigned | NO   | MUL |          |             |
+-----+-----+-----+-----+-----+
```



```
mysql> UPDATE products SET supplierID = 502 WHERE productID  = 2004;
-- Choose a valid productID

mysql> SELECT * FROM products;
+-----+-----+-----+-----+-----+
| productID | productCode | name      | quantity | price   | supplierID |
+-----+-----+-----+-----+-----+
| 2001 | PEC        | Pencil 3B |      500 | 0.52   |      501 |
| 2002 | PEC        | Pencil 4B |      200 | 0.62   |      501 |
| 2003 | PEC        | Pencil 5B |      100 | 0.73   |      501 |
| 2004 | PEC        | Pencil 6B |      500 | 0.47   |      502 |
+-----+-----+-----+-----+-----+
```

Select with join

```
-- ANSI style: JOIN ... ON ...
mysql> SELECT products.name, price, suppliers.name
      FROM products
            JOIN suppliers ON products.supplierID = suppliers.supplierID
      WHERE price < 0.6;
+-----+-----+
| name    | price | name        |
+-----+-----+
| Pencil 3B |  0.52 | ABC Traders |
| Pencil 6B |  0.47 | XYZ Company |
+-----+
-- Need to use products.name and suppliers.name to differentiate the two "names"
```

-- Join via WHERE clause (lagacy and not recommended)

```
mysql> SELECT products.name, price, suppliers.name
      FROM products, suppliers
      WHERE products.supplierID = suppliers.supplierID
            AND price < 0.6;
```

```
+-----+-----+
| name    | price | name        |
+-----+-----+
| Pencil 3B |  0.52 | ABC Traders |
| Pencil 6B |  0.47 | XYZ Company |
+-----+
```

```
-- Use aliases for column names for display
mysql> SELECT products.name AS `Product Name`, price, suppliers.name AS `Supplier Name`
   FROM products
        JOIN suppliers ON products.supplierID = suppliers.supplierID
      WHERE price < 0.6;
+-----+-----+
| Product Name | price | Supplier Name |
+-----+-----+
| Pencil 3B    |  0.52 | ABC Traders  |
| Pencil 6B    |  0.47 | XYZ Company  |
+-----+-----+
-- Use aliases for table names too
mysql> SELECT p.name AS `Product Name`, p.price, s.name AS `Supplier Name`
   FROM products AS p
        JOIN suppliers AS s ON p.supplierID = s.supplierID
      WHERE p.price < 0.6;
```

Relation plusieurs à plusieurs

Database: southwind Table: products_suppliers	
productID INT (Foreign Key)	supplierID INT (Foreign Key)
2001	501
2002	501
2003	501
2004	502
2001	503

Database: southwind Table: suppliers		
supplierID INT	name VARCHAR(30)	phone CHAR(8)
501	ABC Traders	88881111
502	XYZ Company	88882222
503	QQ Corp	88883333

Database: southwind

Table: products

productID INT	productCode CHAR(3)	name VARCHAR(30)	quantity INT	price DECIMAL(10,2)
2001	PEC	Pencil 3B	500	0.52
2002	PEC	Pencil 4B	200	0.62
2003	PEC	Pencil 5B	100	0.73
2004	PEC	Pencil 6B	500	0.47

```
mysql> CREATE TABLE products_suppliers (
    productID  INT UNSIGNED NOT NULL,
    supplierID INT UNSIGNED NOT NULL,
        -- Same data types as the parent tables
    PRIMARY KEY (productID, supplierID),
        -- uniqueness
    FOREIGN KEY (productID) REFERENCES products (productID),
    FOREIGN KEY (supplierID) REFERENCES suppliers (supplierID)
);

mysql> DESCRIBE products_suppliers;
+-----+-----+-----+-----+-----+
| Field      | Type           | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| productID | int(10) unsigned | NO   | PRI | NULL    |       |
| supplierID | int(10) unsigned | NO   | PRI | NULL    |       |
+-----+-----+-----+-----+-----+

mysql> INSERT INTO products_suppliers VALUES (2001, 501), (2002, 501),
        (2003, 501), (2004, 502), (2001, 503);
-- Values in the foreign-key columns (of the child table) must match
-- valid values in the columns they reference (of the parent table)

mysql> SELECT * FROM products_suppliers;
+-----+-----+
| productID | supplierID |
+-----+-----+
|     2001 |         501 |
|     2002 |         501 |
|     2003 |         501 |
|     2004 |         502 |
|     2001 |         503 |
+-----+-----+
```

```
mysql> SHOW CREATE TABLE products \G
Create Table: CREATE TABLE `products` (
  `productID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `productCode` char(3)          NOT NULL DEFAULT '',
  `name`        varchar(30)       NOT NULL DEFAULT '',
  `quantity`    int(10) unsigned NOT NULL DEFAULT '0',
  `price`       decimal(7,2)      NOT NULL DEFAULT '99999.99',
  `supplierID` int(10) unsigned NOT NULL DEFAULT '501',
  PRIMARY KEY (`productID`),
  KEY `supplierID` (`supplierID`),
  CONSTRAINT `products_ibfk_1` FOREIGN KEY (`supplierID`)
    REFERENCES `suppliers` (`supplierID`)
) ENGINE=InnoDB AUTO_INCREMENT=1006 DEFAULT CHARSET=latin1

mysql> ALTER TABLE products DROP FOREIGN KEY products_ibfk_1;

mysql> SHOW CREATE TABLE products \G

mysql> ALTER TABLE products DROP supplierID;

mysql> DESC products;
```

```
mysql> SELECT products.name AS `Product Name`, price, suppliers.name AS `Supplier Name`
      FROM products_suppliers
        JOIN products ON products_suppliers.productID = products.productID
        JOIN suppliers ON products_suppliers.supplierID = suppliers.supplierID
      WHERE price < 0.6;
```

Product Name	price	Supplier Name
Pencil 3B	0.52	ABC Traders
Pencil 3B	0.52	QQ Corp
Pencil 6B	0.47	XYZ Company

-- Define aliases for tablenames too

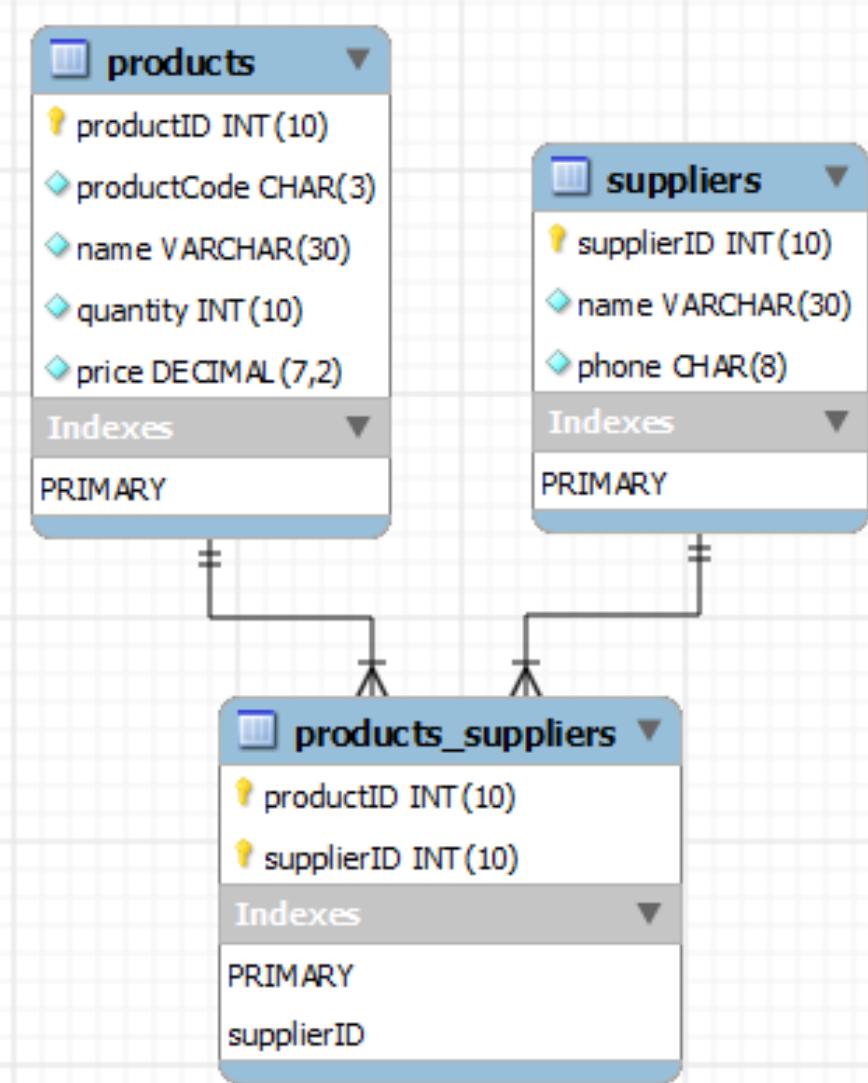
```
mysql> SELECT p.name AS `Product Name`, s.name AS `Supplier Name`
      FROM products_suppliers AS ps
        JOIN products AS p ON ps.productID = p.productID
        JOIN suppliers AS s ON ps.supplierID = s.supplierID
      WHERE p.name = 'Pencil 3B';
```

Product Name	Supplier Name
Pencil 3B	ABC Traders
Pencil 3B	QQ Corp

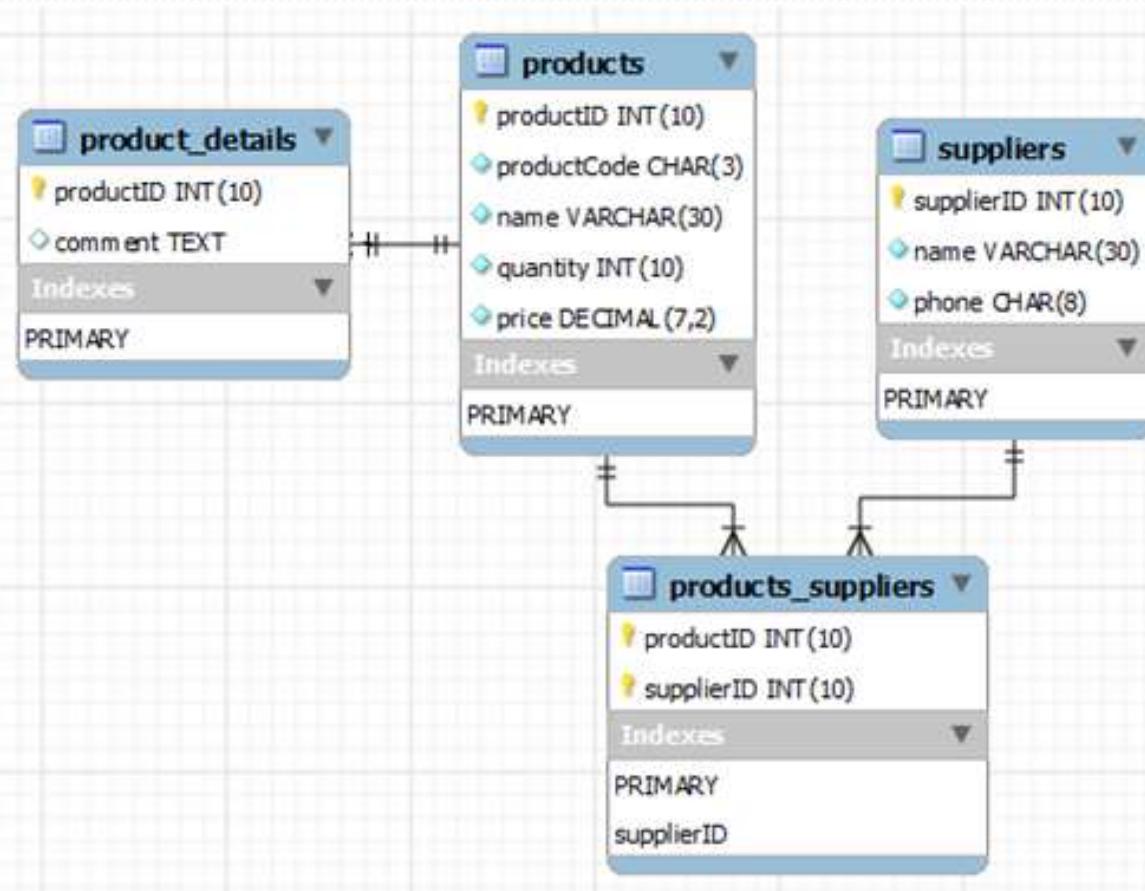
-- Using WHERE clause to join (legacy and not recommended)

```
mysql> SELECT p.name AS `Product Name`, s.name AS `Supplier Name`
      FROM products AS p, products_suppliers AS ps, suppliers AS s
      WHERE p.productID = ps.productID
        AND ps.supplierID = s.supplierID
        AND s.name = 'ABC Traders';
```

Product Name	Supplier Name
Pencil 3B	ABC Traders
Pencil 4B	ABC Traders
Pencil 5B	ABC Traders



Relation un à un



```
mysql> CREATE TABLE product_details (
    productID  INT UNSIGNED  NOT NULL,
        -- same data type as the parent table
    comment     TEXT  NULL,
        -- up to 64KB
    PRIMARY KEY (productID),
    FOREIGN KEY (productID) REFERENCES products (productID)
);
```

```
mysql> DESCRIBE product_details;
```

Field	Type	Null	Key	Default	Extra
productID	int(10) unsigned	NO	PRI	NULL	
comment	text	YES		NULL	

```
mysql> SHOW CREATE TABLE product_details \G
```

```
***** 1. row *****
```

```
Table: product_details
```

```
Create Table: CREATE TABLE `product_details` (
    `productID` int(10) unsigned NOT NULL,
    `comment` text,
    PRIMARY KEY (`productID`),
    CONSTRAINT `product_details_ibfk_1` FOREIGN KEY (`productID`) REFERENCES `products` (`productID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```