


## I) BASES DE DONNEES

### I-1) Présentation



Deux approches sont envisageables pour **STOCKER** et **GERER** (ajouter, consulter, modifier et supprimer) des données :

Développer une application informatique	Mettre en place une base de données
<p>L'application va exploiter directement des <b>FICHIERS</b> (texte, bureautiques, HTM, XML, etc...).</p> <p>Une fois chargées en mémoire, les données seront exploitables par des traitements spécifiques autour de structures de données : <i>variables, tableaux, classes d'objet et collections, etc...</i></p>	<p> <b>Séparation des tâches :</b></p> <p><b>Un logiciel SGBD</b> (système de gestion des bases de données) chargé :</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Du STOCKAGE et la SECURISATION (sauvegarde, reprise après panne,...) des données centralisées dans la base.</li> <li><input type="checkbox"/> D'assurer la CONFIDENTIALITE (droits d'accès).</li> <li><input type="checkbox"/> De gérer les ACCES CONCURRENTS (transactions).</li> <li><input type="checkbox"/> Des tâches d'INTERROGATION et MISES A JOUR rapides.</li> <li><input type="checkbox"/> ...</li> </ul> <p><b>Une application</b> (PHP, C#, etc...) chargée d'exploiter les données : envoi de REQUETES, puis traitement spécifique sur le résultat (données) obtenu.</p>

### I-2) Objectifs fondamentaux d'une base de données



Une base de données est une entité dans laquelle il est possible de **STOCKER** des données de façon **STRUCTUREE** et avec le moins de **REDONDANCE** possible. Ces données doivent pouvoir être utilisées par des programmes et par des utilisateurs différents. Les objectifs sont les suivants :

☐ **Centraliser l'information pour permettre :**

- la suppression de la redondance,
- l'unicité de la saisie,
- la centralisation des contrôles.

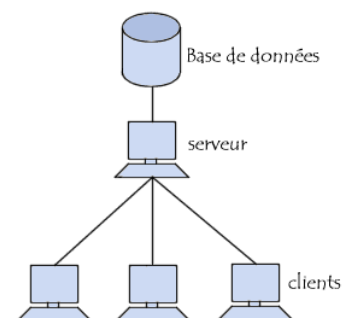
☐ **Assurer l'indépendance données - traitements:**

La BD décrit les données d'une organisation et, par conséquent, elle doit s'adapter à l'évolution de l'entreprise. Pour permettre cette adaptation, il est nécessaire de rendre les traitements (programmes d'application) indépendants des données. Il faut donc :

- permettre à différents programmes d'application d'avoir différentes vues d'une même donnée,
- qu'une modification de l'image d'une donnée ou d'une stratégie d'accès soit sans impact sur les traitements existants,
- etc ...

☐ **Assurer l'intégrité de l'information.**

☐ **Assurer la diffusion et le partage de l'information au sein de la communauté d'utilisateurs.**



### I-3) Système de gestion de Base de données (SGBD)

Le logiciel permettant à un utilisateur d'interagir avec une base de données est un Système de Gestion de Bases de Données (SGBD). Il permet principalement d'organiser les données sur les supports périphériques et fournit les procédures de manipulation de ces données.

#### Quelques SGBD du marché :



- ☐ Oracle Database, créé en 1979 par Oracle Corporation.
- ☐ SQL Server créé en 1989 par Microsoft.
- ☐ MySQL créé en 1995 par MySQL AB et détenu par Oracle Corporation depuis 2005.
- ☐ PostgreSQL développé en 1985 par Michael Stonebraker.
- ☐ Etc...

## Les objectifs sont les suivants :

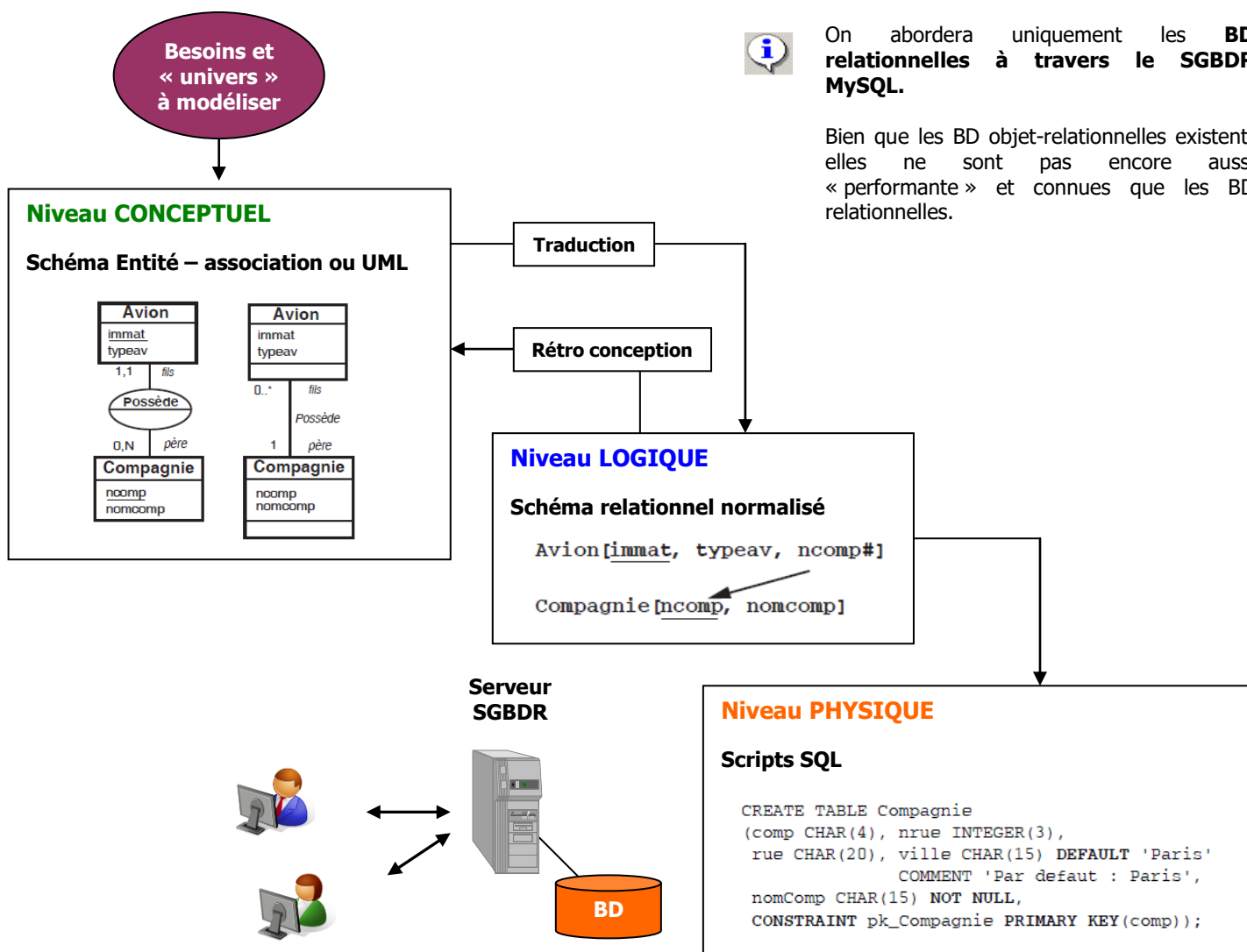
- ❑ **DESCRIPTION** Le SGBD doit fournir à l'utilisateur un outil de description des données qui seront stockées dans la BD.
- ❑ **UTILISATION**  
La fonction utilisation est celle qui a pour but d'offrir à l'utilisateur une interaction avec la BD, sous forme d'un dialogue, pour rechercher, sélectionner et modifier les données.
- ❑ **INTEGRITE**  
Le SGBD doit offrir la possibilité à l'usager de définir des règles qui permettront de maintenir l'intégrité de la BD. Ces règles correspondent à des propriétés qui devront toujours être vérifiées dans la BD, quelles que soient les valeurs enregistrées.
- ❑ **CONFIDENTIALITE**  
Si une BD est partagée entre plusieurs utilisateurs, certains sous-ensembles ne doivent être accessibles que par des personnes réellement autorisées. Pour cela, le SGBD doit offrir des mécanismes permettant de vérifier les droits d'accès des utilisateurs.
- ❑ **CONCURRENCE D'ACCES**  
Le SGBD doit détecter et traiter les conflits d'accès aux informations partagées.
- ❑ **SECURITE DE FONCTIONNEMENT**  
En cas d'incident (matériel ou logiciel), la BD peut prendre un état incohérent. Le SGBD permet, au redémarrage, de remettre la BD dans un état satisfaisant.

## I-4) Conception



Quels que soient les Systèmes de Gestion de Bases de Données, il y a toujours une **phase préalable** à leur utilisation qui concerne la définition du contenu de la base de données qu'ils auront à gérer.

La détermination de ce contenu s'effectue à partir de l'énoncé des besoins des futurs utilisateurs de la base. Leur analyse permet de déterminer les données à enregistrer au sein de la base : on élabore ainsi le schéma (ou structure) de la base de données.



On abordera uniquement les **BD relationnelles à travers le SGBDR MySQL**.

Bien que les BD objet-relationnelles existent, elles ne sont pas encore aussi « performante » et connues que les BD relationnelles.

## II) Le système de gestion de bases de données (SGBD) MySQL

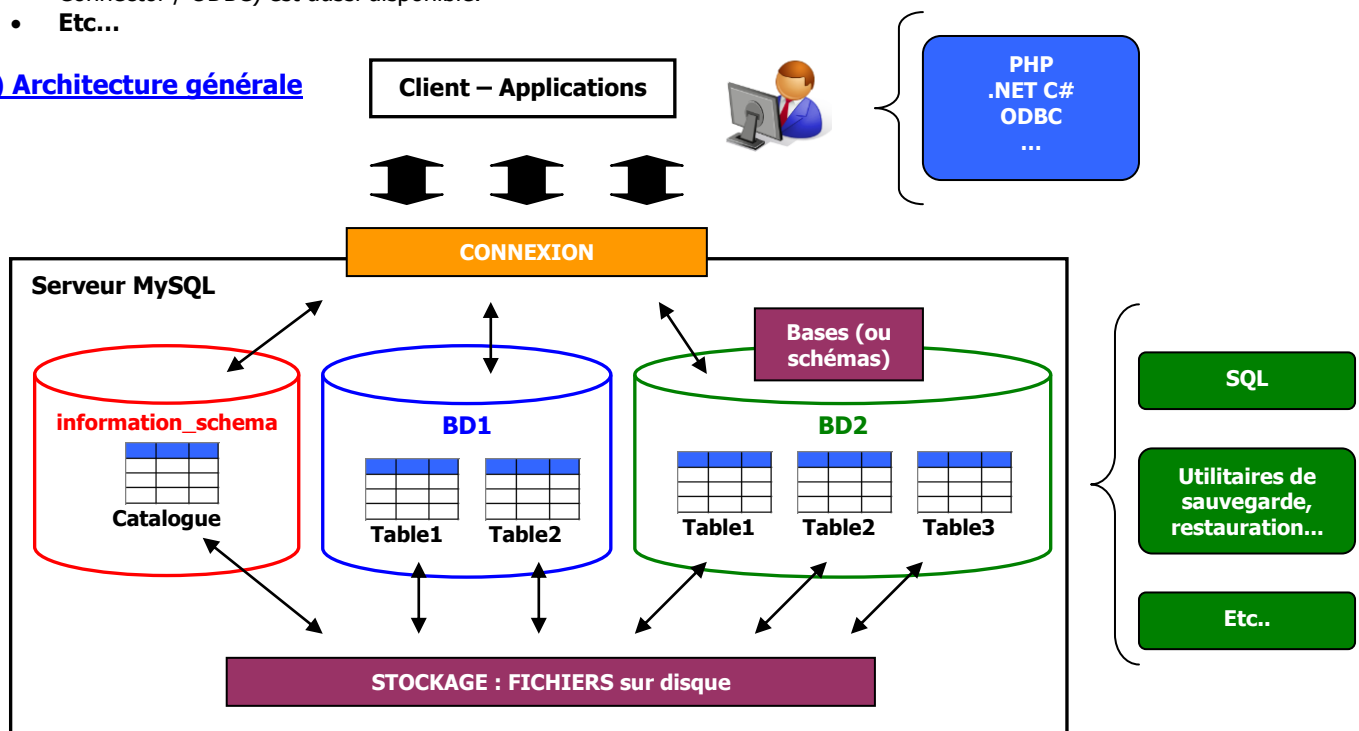
### II-1) Présentation



MySQL est l'œuvre d'une société suédoise, MySQL AB, achetée en 2008 par Sun. En 2009, Sun Microsystems a été acquis par Oracle Corporation, mettant entre les mains d'une même société les deux produits concurrents que sont Oracle Database et MySQL.

- **MySQL est un serveur de bases de données relationnelles SQL** orienté plutôt vers le service de données déjà en place (LECTURE) que vers celui de MISES A JOUR fréquentes et fortement sécurisées, pour lequel d'autres SGBD tels qu'ORACLE sont plus adaptés.
- **MySQL est un logiciel libre** développé sous double licence en fonction de l'utilisation qui en est faite : dans un produit libre ou dans un produit propriétaire. Dans ce dernier cas, la licence est payante, sinon c'est la licence publique générale GPL qui s'applique.
- **MySQL fonctionne sur de nombreux systèmes d'exploitation** : Linux, Mac OS, NetWare, Windows (2000, XP, Vista et 7)...
- **Les données sont accessibles en utilisant des langages de programmation** C, C++, VB, VB .NET, C#, Java, PHP... Une API (interface de programmation) spécifique est disponible pour chacun d'entre eux. Une interface ODBC (MySQL Connector / ODBC) est aussi disponible.
- **Etc...**

### II-2) Architecture générale



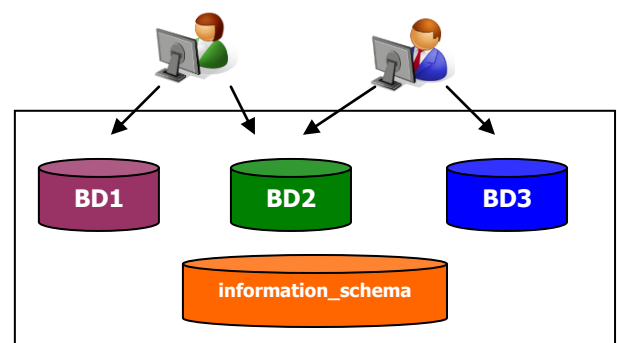
#### REMARQUES :

- MySQL fait la distinction entre les **BASES** (ou schémas) et le **STOCKAGE**. Le stockage des données est assuré par des **moteurs de tables**. MySQL dispose de plusieurs moteurs de tables (MyISAM, InnoDB...) qui définissent l'accès à certaines fonctionnalités. Par exemple le moteur MYISAM ne supporte pas les clés étrangères et les transactions, le moteur InnoDB les supporte, etc...
- Le schéma (base) **INFORMATION\_SCHEMA** contient le **CATALOGUE** système (= tables où est enregistrée la définition des tables « utilisateur », index, procédures, déclencheurs ...).

### Notion de schéma (database)

MySQL appelle « database » (base de données) un regroupement logique d'objets (tables, index, vues, déclencheurs, procédures cataloguées, etc.) pouvant être stockés à différents endroits de l'espace disque.

- Un utilisateur sera associé à un mot de passe pour pouvoir se connecter et manipuler des tables (s'il en a le droit, bien sûr).
- Pour MySQL, il n'y a pas de notion d'appartenance d'un objet (table, index, etc.) à un utilisateur. Un objet appartient à son schéma (database). Ainsi, deux utilisateurs distincts se connectant sur la même base (BD2) ne pourront pas créer chacun une table portant le même nom. S'ils doivent le faire, ce sera dans deux bases différentes



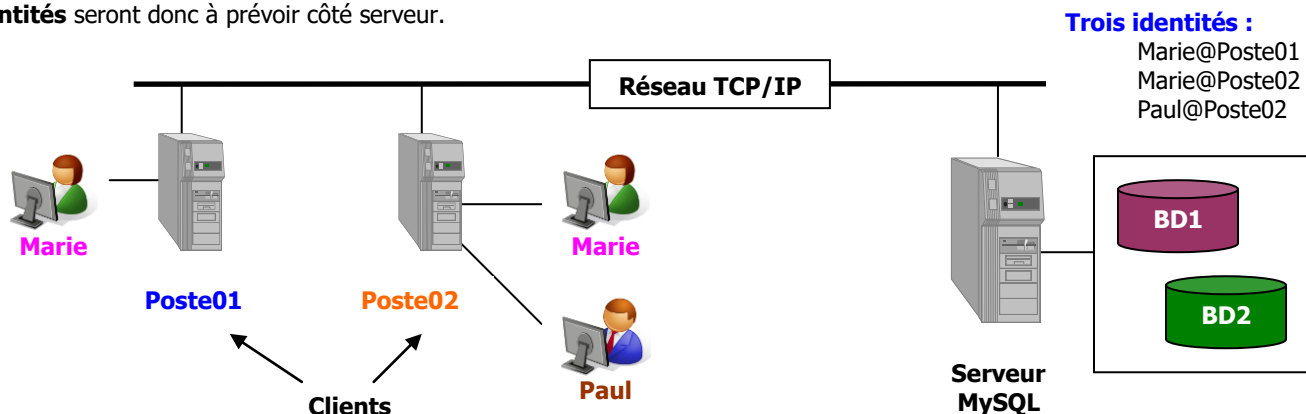
## Notion d'hôte et d'identité

MySQL dénomme « host » (hôte) la machine hébergeant le SGBD : il est alors possible de distinguer des accès d'un même utilisateur suivant qu'il se connecte à partir d'une machine ou d'une autre. La notion d'identité est basée sur le couple :

- nom d'utilisateur MySQL (user) côté serveur,
- machine cliente.

La figure suivante illustre le fait que **deux utilisateurs** peuvent se connecter par deux accès différents.

**Trois identités** seront donc à prévoir côté serveur.



## III) Le langage SQL

### III-1) Historique et présentation

En 1977, IBM a implanté le modèle relationnel au travers du langage SEQUEL (« Structured English as QUery Language »), rebaptisé par la suite SQL (« Structured Query Language » – langage structuré de requêtes).



**SQL est un pseudo-langage informatique (de type requête) standard et normalisé, destiné à interroger ou à manipuler une base de données relationnelle.**

- La première norme (SQL1) date de 1987. Elle était le résultat de compromis entre constructeurs, mais elle était fortement influencée par le dialecte d'IBM.
- SQL2 a été normalisée en 1992. Elle définit quatre niveaux de conformité, et les langages SQL des principaux éditeurs sont tous conformes au premier niveau et ont beaucoup de caractéristiques relevant des niveaux supérieurs.
- Depuis 1999, la norme est appelée SQL3. Elle comporte de nombreuses parties (concepts objets, entrepôts de données, accès à des sources non SQL, réplication des données, etc.).

### Classification des ordres SQL :

Ordres SQL	Aspect du langage
CREATE – ALTER – DROP – RENAME...	Définition des données (LDD)
INSERT – UPDATE – DELETE...	Manipulation des données (LMD)
SELECT	Interrogation des données (LID)
GRANT – REVOKE – COMMIT – ROLLBACK...	Contrôle des données (LCD)

### III-2) Définition des données

#### a) CREATION DE TABLES (CREATE TABLE)

Une table est créée en SQL par l'instruction CREATE TABLE, modifiée au niveau de sa structure par l'instruction ALTER TABLE et supprimée par la commande DROP TABLE.

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] [nomBase.]nomTable
(colonne1 type1 [NOT NULL | NULL] [DEFAULT valeur1] [COMMENT 'chaîne1'] ,
colonne2 type2 [NOT NULL | NULL] [DEFAULT valeur2] [COMMENT 'chaîne2'] ,
...
[CONSTRAINT nomContrainte1 typeContrainte1] ...)
[ENGINE= InnoDB | MyISAM | ...];
```



## REMARQUES :

- ❑ **TEMPORARY** : pour créer une table qui n'existera que durant la session courante (la table sera supprimée à la déconnexion. Il faut posséder le privilège CREATE TEMPORARY TABLES.
- ❑ **IF NOT EXISTS** : permet d'éviter qu'une erreur se produise si la table existe déjà (si c'est le cas, elle n'est aucunement affectée par la tentative de création).
- ❑ **nomBase** : s'il est omis, il sera assimilé à la base connectée (cf. USE nom BD). S'il est précisé, il faut que l'utilisateur courant ait le droit de créer une table dans l'autre base).
- ❑ **Colonne « i » type « i »** : nom d'une colonne et son type (INTEGER, CHAR, DATE...). La directive DEFAULT fixe une valeur par défaut. La directive NOT NULL interdit que la valeur de la colonne soit nulle.
- ❑ **COMMENT** : (jusqu'à 60 caractères) permet de commenter une colonne. Ce texte sera ensuite automatiquement affiché à l'aide des commandes SHOW CREATE TABLE et SHOW FULL COLUMNS.
- ❑ **nomContrainte « i » typeContrainte « i »** : nom de la contrainte et son type (clé primaire, clé étrangère, etc.).
- ❑ **ENGINE** : définit le type de table (par défaut InnoDB, bien adapté à la programmation de transactions et adopté dans ce cours). MyISAM parfaitement robuste, mais ne supportant pas pour l'instant l'intégrité référentielle. D'autres types existent, citons MEMORY pour les tables temporaires, ARCHIVE, etc.

## EXEMPLE

Soit la table « Compagnie » récapitulant les données des différentes compagnies aérienne :

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

Après avoir lancé le serveur MySQL, dans une fenêtre de commande Windows, lancer l'interface en ligne en connectant l'utilisateur « root » avec le mot de passe que donné lors de l'installation.



```
C:\Program Files\EasyPHP-5.3.5.0\mysql\bin>mysql --user=root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.54-community-log MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```



Un **script SQL** va permettre, tout d'abord de CREER la BASE DE DONNEES, puis de créer la table « Compagnie » et d'y ajouter plusieurs tuples :

```
-- Création de la BD
CREATE DATABASE sio
DEFAULT CHARACTER SET ascii
COLLATE ascii_general_ci;

-- Activer la BD
USE sio;

-- Table COMPAGNIE

CREATE TABLE Compagnie
(comp CHAR(4),
nrue INTEGER(3),
rue CHAR(20),
ville CHAR(15) DEFAULT 'Paris'
COMMENT 'Par défaut : Paris',
nomComp CHAR(15) NOT NULL);

INSERT INTO Compagnie VALUES ('AF', 10, 'Gambetta', 'Paris', 'Air France');
INSERT INTO Compagnie VALUES ('SING', 7, 'Camparols', 'Singapour', 'Singapore AL');

SELECT * FROM Compagnie;
```

Pour exécuter un script SQL, saisir sur la ligne de commande : **source** nomsript.sql

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

La table contient cinq colonnes (quatre chaînes de caractères et un numérique de trois chiffres). La colonne ville est commentée. La table inclut en plus deux **CONTRAINTES** :

- ❑ **DEFAULT** qui fixe Paris comme valeur par défaut de la colonne « ville ».
- ❑ **NOT NULL** qui impose une valeur non nulle dans la colonne « nomComp ».



Pour obtenir la description de la **STRUCTURE de la table**, il suffit de taper directement :

```
mysql> desc compagnie;
```

Field	Type	Null	Key	Default	Extra
comp	char(4)	YES		NULL	
nrue	int(3)	YES		NULL	
rue	char(20)	YES		NULL	
ville	char(15)	YES		Paris	
nomComp	char(15)	NO		NULL	

## CONTRAINTES



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

Les contraintes peuvent être déclarées de deux manières :

- ☐ **En même temps que la colonne** : ces contraintes sont dites « inline constraints ». L'exemple précédent en déclare deux (DEFAULT et NOT NULL).
- ☐ **Après que la colonne ait été déclarée** et peuvent être personnalisées par un nom (out-of-line constraints).

```
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)
```



## REMARQUES :

- ☐ La contrainte UNIQUE impose une **VALEUR DISTINCTE** au niveau de la table (les valeurs nulles en font exception).
- ☐ La contrainte PRIMARY KEY déclare la **CLE PRIMAIRE** de la table. Un index est généré automatiquement sur la ou les colonnes concernées. Les colonnes clés primaires ne peuvent être ni nulles ni identiques.
- ☐ La contrainte FOREIGN KEY déclare une **CLE ETRANGERE** entre une table enfant et une table père. Ces contraintes définissent **l'intégrité référentielle**.  
Les directives ON UPDATE et ON DELETE disposent de quatre options que qui seront détaillée avec les directives MATCH dans le paragraphe III-3) Manipulation des données.

## EXEMPLE :

Soient les tables « Pilote » et « Compagnie » suivantes :

brevet	nom	nbHVol	compa
PL-1	Louise Ente	450.00	AF
PL-2	Jules Ente	900.00	AF
PL-3	Paul Durand	1000.00	SING

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

Le script SQL précédent doit être modifié (la commande DROP TABLE Compagnie ; permet de détruire la table).

```
-- Table COMPAGNIE

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

Deux contraintes en ligne et une contrainte nommée de clé primaire ont été définies. On pourra adopter les conventions suivantes :

- ☐ Préfixez par **pk\_** le nom d'une contrainte clé primaire, **fk\_** une clé étrangère, **ck\_** une vérification, **un\_** une unicité, **nn\_** une valeur non nulle.
- ☐ Pour une contrainte **clé primaire**, suffixez du nom de la table la contrainte (exemple pk\_Avion).
- ☐ Pour une contrainte **clé étrangère**, renseignez les noms de la table source, de la clé, et de la table cible (exemple fk\_Pil\_compa\_Comp).



## Cinq contraintes nommées :

Clé primaire « pk\_Pilote »

NOT NULL « nn\_nom »

CHECK (nombre d'heures de vol compris entre 0 et 20000) « ck\_nbHvol »

UNIQUE (homonymes interdits) « un\_nom »

Clé étrangère « fk\_Pil\_compa\_Comp »

```
-- Table PILOTE

CREATE TABLE Pilote
(brevet CHAR(6), nom CHAR(15),
nbHVol DECIMAL(7,2), compa CHAR(4),
CONSTRAINT pk_Pilote PRIMARY KEY(brevet),
CONSTRAINT nn_nom CHECK (nom IS NOT NULL),
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));
```



**La contrainte CHECK** est prise en charge au niveau de la déclaration mais n'est pas encore opérationnelle même dans la version 5.1.

```
--ça passe... contrainte hors ligne NOT NULL !...
INSERT INTO Pilote VALUES ('PL-4', NULL, 1000, 'SING');

--ça passe... aussi
INSERT INTO Pilote VALUES ('PL-5', 'PB sur Check', 30000, 'SING');
```

```
mysql> select * from pilote;
```

brevet	nom	nbHvol	compa
PL-1	Louise Ente	450.00	AF
PL-2	Jules Ente	900.00	AF
PL-3	Paul Durand	1000.00	SING
PL-4	NULL	1000.00	SING
PL-5	PB sur Check	30000.00	SING

Lorsque la contrainte NOT NULL est **déclarée en ligne**, la vérification est effectuée correctement :

```
mysql> INSERT INTO Compagnie VALUES ('BA', 5, 'Paix', 'Paris', NULL);
ERROR 1048 (23000): Le champ 'nomComp' ne peut être vide (null)
```

## TYPES DES COLONNES

Pour décrire les colonnes d'une table, MySQL fournit les types prédéfinis suivants :

### Caractères (Extrait)

Type	Description	Commentaire pour une colonne
CHAR(n)	Chaîne fixe de n octets ou caractères.	Taille fixe (maximum de 255 caractères).
VARCHAR(n)	Chaîne variable de n caractères ou octets.	Taille variable (maximum de 65 535 caractères).

### Valeurs numériques (Extrait)

Type	Description
SMALLINT[(n)] [UNSIGNED][ZEROFILL]	Entier (sur 2 octets) de - 32 768 à 32 767 signé, 0 à 65 535 non signé.
MEDIUMINT[(n)] [UNSIGNED][ZEROFILL]	Entier (sur 3 octets) de - 8 388 608 à 8 388 607 signé, 0 à 16 777 215 non signé.
INTEGER[(n)] [UNSIGNED][ZEROFILL]	Entier (sur 4 octets) de -2 147 483 648 à 2 147 483 647 signé, 0 à 4 294 967 295 non signé.
BIGINT[(n)] [UNSIGNED][ZEROFILL]	Entier (sur 8 octets) de - 9 223 372 036 854 775 808 à 9 223 372 036 854 775 807 signé, 0 à 18 446 744 073 709 551 615 non signé.
FLOAT[(n[,p])] [UNSIGNED][ZEROFILL]	Flottant (de 4 à 8 octets) p désigne la précision simple (jusqu'à 7 décimales) de -3.4 10+38 à -1.1 10-38, 0, signé, et de 1.1 10-38 à 3.4 10+38 non signé.
DOUBLE[(n[,p])] [UNSIGNED][ZEROFILL]	Flottant (sur 8 octets) p désigne la précision double (jusqu'à 15 décimales) de -1.7 10+308 à -2.2 10-308, 0, signé, et de 2.2 10-308 à 1.7 10+308 non signé.
DECIMAL[(n[,p])] [UNSIGNED][ZEROFILL]	Décimal à virgule fixe, p désigne la précision (nombre de chiffres après la virgule, maximum 30). Par défaut n vaut 10, p vaut 0.



### REMARQUES :

- ☐ La directive UNSIGNED permet de considérer seulement des valeurs positives.
- ☐ La directive ZEROFILL complète par des zéros à gauche une valeur (par exemple : soit un INTEGER(5) contenant 4, si ZEROFILL est appliqué, la valeur extraite sera « 00004 »).

## Dates et heures

Les types suivants permettent de stocker des moments ponctuels (dates, dates et heures, années, et heures). Les fonctions NOW() et SYSDATE() retournent la date et l'heure courantes.

Type	Description	Commentaire pour une colonne
DATE	Dates du 1er janvier de l'an 1000 au 31 décembre 9999 après J.-C.	Sur 3 octets. L'affichage est au format 'YYYY-MM-DD'.
DATETIME	Dates et heures (de 0 h de la première date à 23 h 59 minutes 59 secondes de la dernière date).	Sur 8 octets. L'affichage est au format 'YYYY-MM-DD HH:MM:SS'.
YEAR[(2 4)]	Sur 4 positions : de 1901 à 2155 (incluant 0000). Sur 2 positions : de 70 à 69 (désignant 1970 à 2069).	Sur 1 octet ; l'année est considérée sur 2 ou 4 positions (4 par défaut). Le format d'affichage est 'YYYY'.
TIME	Heures de -838 h 59 minutes 59 secondes à 838 h 59 minutes 59 secondes.	L'heure au format 'HHH:MM:SS' sur 3 octets.

## Données binaires

Les types BLOB (*Binary Large Object*) permettent de stocker des données non structurées comme le multimédia (images, sons, vidéo, etc.). Les quatre types de colonnes BLOB sont TINYBLOB, BLOB, MEDIUMBLOB et LONGBLOB. Ces types sont traités comme des flots d'octets sans jeu de caractère associé.

Type	Description	Commentaire pour une colonne
TINYBLOB( <i>n</i> )	Flot de <i>n</i> octets.	Taille fixe (maximum de 255 octets).
BLOB( <i>n</i> )	Flot de <i>n</i> octets.	Taille fixe (maximum de 65 535 octets).
MEDIUMBLOB( <i>n</i> )	Flot de <i>n</i> octets.	Taille fixe (maximum de 16 mégaoctets).
LONGBLOB( <i>n</i> )	Flot de <i>n</i> octets.	Taille fixe (maximum de 4,29 gigaoctets).

## b) CREATION D'INDEX (CREATE INDEX)

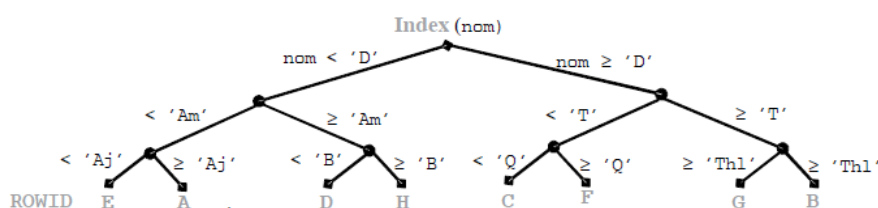


Un index permet d'**ACCELERER L'ACCES** aux données d'une table. Le but principal d'un index est d'éviter de parcourir une table séquentiellement du premier enregistrement jusqu'à celui visé. Le principe d'un index est l'association de l'adresse de chaque enregistrement avec la valeur des colonnes indexées.

Un index est associé à une table et peut être défini sur une ou plusieurs colonnes (dites « indexées »). Une table peut « héberger » plusieurs index. Ils sont **MIS A JOUR AUTOMATIQUEMENT** après rafraîchissement de la table (ajouts et suppressions d'enregistrements ou modification des colonnes indexées). Un index peut être déclaré unique si on sait que les valeurs des colonnes indexées seront toujours uniques.

```
CREATE [UNIQUE ] INDEX nomIndex
[USING BTREE | HASH]
ON nomTable (colonne1 [(taille1)] [ASC | DESC],...);
```

- ❑ **UNIQUE** permet de créer un index qui n'accepte pas les doublons.
- ❑ **ASC** et **DESC** précisent l'ordre (croissant ou décroissant).
- ❑ La plupart des index de MySQL sont stockés dans des arbres équilibrés (ou **BTREE**) :



## EXEMPLE :

Création d'index sur la table Pilote

```
-- Création d'index
USE sio;

-- Index B-tree, ordre décroissant sur les trois
-- premiers caractères du nom des pilotes.
CREATE UNIQUE INDEX idx_Pilote_nom3
USING BTREE
ON Pilote (nom(3) DESC);

-- Index B-tree, ordre croissant sur la colonne clé
-- étrangère compa.
CREATE INDEX idx_Pilote_compa
USING BTREE
ON Pilote (compa);

-- Suppression d'index
DROP INDEX idx_Pilote_nom3 ON Pilote;
DROP INDEX idx_Pilote_compa ON Pilote;
```



## REMARQUES :

- ❑ Un index ralentit les mises à jour de la base. En revanche il accélère les accès.
- ❑ Il est conseillé de créer des index sur des colonnes utilisées dans les clauses de jointures (voir chapitre 4)
- ❑ Il est possible de créer des index pour toutes les colonnes d'une table (jusqu'à concurrence de 16).
- ❑ Les index sont pénalisants lorsqu'ils sont définis sur des colonnes très souvent modifiées ou si la table contient peu de lignes.



### c) SUPPRESSION DE TABLES (DROP TABLE)

L'instruction DROP TABLE entraîne la suppression des données, de la structure, de la description dans le dictionnaire des données, des index, etc...

```
DROP [TEMPORARY] TABLE [IF EXISTS]  
[nomBase.] nomTable1 [, [nomBase.] nomTable2,...]
```



#### Ordre des suppressions

Il suffit de relire à l'envers le script de création des tables pour en déduire l'ordre de suppression

```
-- Les « fils » puis les « pères »  
DROP TABLE Pilote;  
DROP TABLE Compagnie;
```

## III-3) MANIPULATION DES DONNEES

### a) INSERTION d'enregistrements (INSERT)



Il existe plusieurs possibilités d'insertion d'enregistrements dans une table: **l'insertion monoligne** qui ajoute un enregistrement par instruction (traité ici) et **l'insertion multiligne** qui insère plusieurs enregistrements par une requête (traité plus loin).

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

- ❑ DELAYED indique que l'insertion est différée (si la table est modifiée par ailleurs, le serveur attend qu'elle se libère pour y insérer périodiquement de nouveaux enregistrements si elle redevient active entre-temps).
- ❑ LOW\_PRIORITY indique que l'insertion est différée à la libération complète de la table (option à ne pas utiliser sur des tables MyISAM).
- ❑ HIGH\_PRIORITY annule l'option *low priority* du serveur.
- ❑ IGNORE indique que les éventuelles erreurs déclenchées suite à l'insertion seront considérées en tant que *warnings*.
- ❑ **ON DUPLICATE KEY UPDATE** permet de mettre à jour l'enregistrement présent dans la table, qui a déclenché l'erreur de doublon (dans le cas d'un index UNIQUE ou d'une clé primaire). Dans ce cas le nouvel enregistrement n'est pas inséré, seul l'ancien est mis à jour.

#### Renseigner TOUTES les colonnes

```
mysql> select * from compagnie;
```

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

Avant

```
-- INSERT : renseigner TOUTES les colonnes  
  
-- TOUTES les colonnes sont renseignées dans l'ORDRE de leur creation  
INSERT INTO Compagnie VALUES ('BA01', 17, 'La Poste', 'Vichy', 'British Airways');  
  
-- DEFAULT sur 'ville' explicite  
INSERT INTO Compagnie VALUES ('BA02', 123, 'Gambetta', DEFAULT, 'British Airways');  
  
-- NULL sur 'nru' explicite  
INSERT INTO Compagnie VALUES ('BA03', NULL, 'Hoche', 'Lyon', 'British Airways');
```

comp	nru	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
BA01	17	La Poste	Vichy	British Airways
BA02	123	Gambetta	Paris	British Airways
BA03	NULL	Hoche	Lyon	British Airways
SING	7	Camparols	Singapour	Singapore AL

Après

## Renseigner CERTAINES les colonnes

```
-- Colonne 'ville' non spécifiée : utilise DEFAULT implicite
INSERT INTO Compagnie(comp, nrue, rue, nomComp) VALUES ('BA02', 123, 'Gambetta', 'British Airways');
-- Colonne 'nrue' non spécifiée : NULL sur nrue implicite
INSERT INTO Compagnie(comp, rue, ville, nomComp) VALUES ('BA03', 'Hoche', 'Lyon', 'British Airways');
```



comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
BA02	123	Gambetta	Paris	British Airways
BA03	NULL	Hoche	Lyon	British Airways
SING	7	Camparols	Singapour	Singapore AL

## PLUSIEURS enregistrements

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



comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
LUFT	9	Salas	Munich	Luftansa
QUAN	1	Kangourou	Sydney	Quantas
SING	7	Camparols	Singapour	Singapore AL
SNCM	3	P. Paoli	Bastia	Corse Air

## Ne pas respecter des CONTRAINTES

On va insérer des enregistrements dans la table Pilote, qui ne respectent pas des contraintes :

```
mysql> select * from pilote;
```

brevet	nom	nbHVol	compa
PL-1	Louise Ente	450.00	AF
PL-2	Jules Ente	900.00	AF
PL-3	Paul Durand	1000.00	SING



**ATTENTION** : pour que la contrainte de CLE ETRANGERE soit prise en compte, il faut s'assurer que les tables ont été créées avec le MOTEUR « InnoDB » :

```
CREATE TABLE Pilote
(brevet CHAR(6), nom CHAR(15) NOT NULL,
...
CONSTRAINT fk_Pil_compa_Comp
FOREIGN KEY (compa) REFERENCES Compagnie(comp)
)ENGINE=INNODB;
```

### Insertions ne vérifiant pas les contraintes

```
-- Clé PRIMAIRE : pk_Pilote
INSERT INTO Pilote VALUES
('PL-1', 'Henri Gimenez', 950, 'AF');
```

#### Erreur liée à la CLE PRIMAIRE

ERROR 1062 (23000): Duplicata du champ 'PL-1' pour la clef 'PRIMARY'

```
-- Clé ETRANGERE : fk_Pil_compa_Comp
INSERT INTO Pilote VALUES
('PL-5', 'Nicolas Dupond', 500, 'SFR');
```

#### Erreur liée à la CLE ETRANGERE

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`sio`.`pilote`, CONSTRAINT `fk\_Pil\_compa\_Comp` FOREIGN KEY (`compa`) REFERENCES `compagnie` (`comp`))

### Insertions ne vérifiant pas les contraintes

```
-- UNIQUE sur 'nom' : un_nom
INSERT INTO Pilote VALUES
('PL-4', 'Louise Ente', 150, 'LUFT');
```

#### ERREUR liée à l'UNICITE du nom

ERROR 1062 (23000): Duplicata du champ 'Louise Ente' pour la clef 'un\_nom'

```
-- NOT NULL sur 'nom'
INSERT INTO Pilote VALUES
('PL-6', NULL, 175, 'AF');
```

#### Erreur liée à la contrainte en ligne NOT NULL

ERROR 1048 (23000): Le champ 'nom' ne peut être vide (null)

## Données BINAIRES



Le type BIT permet de manipuler des suites variables de bits. Des fonctions sont disponibles pour programmer le « ET », le « OU » exclusif ou inclusif, etc.

```
-- Données BINAIRES
DROP TABLE IF EXISTS Registres;
CREATE TABLE Registres (nom CHAR(2), valeur BIT(16));

-- Prefixe 'b' pour initialiser un tel type
INSERT INTO Registres VALUES ('CO', b'0000010011110111');
INSERT INTO Registres VALUES ('AC', b'0000000011010011');

SELECT nom, BIN(valeur) FROM Registres;

-- ET
SELECT BIN(valeur & b'1010101010101010') AS ET
FROM Registres;

-- OU
SELECT BIN(valeur | b'1010101010101010') AS OU
FROM Registres;
```

nom	BIN(valeur)
CO	10011110111
AC	11010011



ET
10100010
10000010

OU
1010111011111111
101010101111011

## Type Énumération



Le type ENUM est considéré comme une liste de chaînes de caractères. Toute valeur d'une colonne de ce type devra appartenir à cette liste établie lors de la création de la table.

```
-- Avec ENUM
DROP TABLE IF EXISTS Cursus;
CREATE TABLE Cursus
(num CHAR(4), nom CHAR(15),
diplome ENUM ('BTS','DUT','Licence'),
CONSTRAINT pk_Cusus PRIMARY KEY(num));

INSERT INTO Cursus VALUES ('E1', 'Dupond', 'BTS');
INSERT INTO Cursus VALUES ('E2', 'Duval', 'Licence');
INSERT INTO Cursus VALUES ('E3', 'Durand', 'Master');

SHOW WARNINGS;

SELECT nom, diplome FROM Cursus ;

DROP TABLE Cursus;
```



nom	diplome
Dupond	BTS
Duval	Licence
Durand	

Level	Code	Message
Warning	1265	Data truncated for column 'diplome' at row 1

## ENSEMBLES



Le type SET permet de comparer une liste à une combinaison de valeurs permises à partir d'un ensemble de référence (chaînes de caractères).

```
DROP TABLE IF EXISTS Cursus;
CREATE TABLE Cursus
(num CHAR(4), nom CHAR(15),
diplomes SET ('BTS','DUT','Licence'),
CONSTRAINT pk_Cusus PRIMARY KEY(num));

INSERT INTO Cursus VALUES ('E1', 'Dupond', ('BTS'));
INSERT INTO Cursus VALUES ('E2', 'Duval', ('BTS,Licence'));
INSERT INTO Cursus VALUES ('E3', 'Durand', ('BTS,Licence,Master'));
```

nom	diplomes
Dupond	BTS
Duval	BTS,Licence
Durand	BTS,Licence

Level	Code	Message
Warning	1265	Data truncated for column 'diplomes' at row 1

## DATES et HEURES



Pour stocker des données de type DATE / HEURE, MySQL peut les considérer soit en tant que chaînes de caractères soit comme numériques.

### Exemple avec DATE et DATETIME

```
CREATE TABLE Personne
(nom VARCHAR(20), dateNaiss DATE, dateEmbauche DATETIME,
 CONSTRAINT pk_Personne PRIMARY KEY(nom));

INSERT INTO Personne VALUES ('Paul Dupond', '1985-03-05', SYSDATE());
INSERT INTO Personne VALUES ('Rose Davolio', '1982/02/07', '2000-02-05 08:30:00');
```

nom	dateNaiss	dateEmbauche
Paul Dupond	1985-03-05	2011-02-14 12:55:43
Rose Davolio	1982-02-07	2000-02-05 08:30:00

### Exemple avec YEAR et TIME

```
CREATE TABLE Personne
(nom VARCHAR(20), dateNaiss DATE,
 dateEmbauche YEAR, pasPromoDepuis TIME,
 CONSTRAINT pk_Personne PRIMARY KEY(nom));

-- 1 jour et 23 heures (47 heures)
INSERT INTO Personne VALUES ('Paul Dupond', '1985-03-05', '2002', '1 23:0:0');
-- 15 heures, 26 minutes 30 secondes
INSERT INTO Personne VALUES ('Rose Davolio', '1982/02/07', 2000, '15:26:30');
```

nom	dateNaiss	dateEmbauche	pasPromoDepuis
Paul Dupond	1985-03-05	2002	47:00:00
Rose Davolio	1982-02-07	2000	15:26:30

## SEQUENCES (« numéro automatiques »)

MySQL offre la possibilité de générer automatiquement des valeurs numériques. Ces valeurs sont bien utiles pour composer, par exemple, des **CLES PRIMAIRES** de tables

### Création d'une table AFFRETER (=avions affrétés par les compagnies)

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

INSERT INTO Affreter (comp, immat, dateAff) VALUES ('AF', 'F-WTSS', '2005-05-13');
INSERT INTO Affreter (comp, immat, dateAff) VALUES ('SING', 'F-GAFU', '2005-02-05');
INSERT INTO Affreter VALUES (NULL, 'AF', 'F-WTSS', '2005-09-11');
INSERT INTO Affreter VALUES (0, 'AF', 'F-GLFS', '2005-09-11');

SELECT * FROM Affreter ;

-- Dernier créé
SELECT LAST_INSERT_ID();
```

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

LAST_INSERT_ID()
4



### 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). Seules les valeurs à venir de la séquence modifiée seront changées.

Supposons qu'on désire continuer à insérer des nouveaux affrètements à partir de la valeur 100. Le prochain affrètement sera estimé à 100 et les insertions suivantes prendront en compte le nouveau point de départ tout en laissant intactes les données existantes des tables.

```
ALTER TABLE Affreter AUTO_INCREMENT = 100;

INSERT INTO Affreter (comp, immat, dateAff) VALUES ('SING', 'F-NEW', SYSDATE());
INSERT INTO Affreter (comp, immat, dateAff) VALUES ('LUFT', 'D-FEDG', SYSDATE());
```



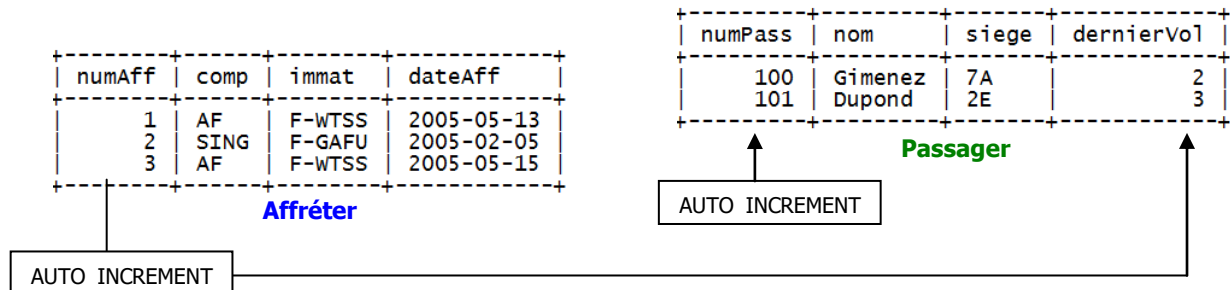
numAff	comp	immat	dateAff
1	AF	F-WTSS	2005-05-13
2	SING	F-GAFU	2005-02-05
3	AF	F-WTSS	2005-09-11
4	AF	F-GLFS	2005-09-11
100	SING	F-NEW	2011-02-14
101	LUFT	D-FEDG	2011-02-14

last_insert_id()
101



### Utilisation en tant que clé étrangère

Créons **deux séquences** qui vont permettre de donner leur valeur aux **clés primaires** des deux tables « Affréter » et « Passager » (les affrètements commencent à 1, les passagers à 100).



-- CLE PRIMAIRE

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



### Table Passager :

- ☐ La **valeur de départ** d'une séquence peut être définie à la fin de l'ordre CREATE TABLE.
- ☐ Une contrainte de **CLE ETRANGERE** spécifie que la numéro du dernier vol doit appartenir à l'ensemble des numéros de la table Affréter, déclarés en tant que clé primaire.

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

```
INSERT INTO Affreter(comp,immat,dateAff) VALUES ('AF', 'F-WTSS', '2005-05-13');
INSERT INTO Affreter(comp,immat,dateAff) VALUES ('SING', 'F-GAFU', '2005-02-05');
```

```
INSERT INTO Passager VALUES (NULL, 'Gimenez', '7A', LAST_INSERT_ID());
INSERT INTO Affreter VALUES (NULL, 'AF', 'F-WTSS', '2005-05-15');
INSERT INTO Passager VALUES (NULL, 'Dupond', '2E', LAST_INSERT_ID());
```



Utilisation de la fonction **LAST\_INSERT\_ID** dans les insertions pour récupérer la valeur de la clé primaire.

## b) MODIFICATIONS de colonnes (UPDATE)



L'instruction UPDATE permet la **mise à jour des colonnes** d'une table.

```
UPDATE [LOW_PRIORITY] [IGNORE] [nomBase.] nomTable
SET col_name1=expr1 [, col_name2=expr2 ...]
SET colonne1 = expression1 | (requête_SELECT) | DEFAULT
[,colonne2 = expression2...]
[WHERE (condition)]
[ORDER BY listeColonnes]
[LIMIT nbreLimite]
```

- ☐ LOW\_PRIORITY et IGNORE correspond à ce qui a été décrit pour la clause INSERT.
- ☐ SET actualise une colonne en lui affectant une expression (valeur, valeur par défaut, calcul ou résultat d'une requête).
- ☐ WHERE filtre les lignes à mettre à jour dans la table :
  - ☐ Si aucune condition n'est précisée, tous les enregistrements seront actualisés.
  - ☐ Si la condition ne filtre aucune ligne, aucune mise à jour ne sera réalisée.
- ☐ ORDER BY indique l'ordre de modification des colonnes.
- ☐ LIMIT spécifie le nombre maximum d'enregistrements à changer (par ordre de clé primaire croissante).



### Modification d'UNE colonne

Modifions la compagnie de code 'LUFT'  
en affectant la valeur 50 à la colonne nrue.

```
-- MAJ d'une colonne
UPDATE Compagnie
SET nrue = 50
WHERE comp = 'LUFT';
```

comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
LUFT	9	Salas	Munich	Luftansa



comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
LUFT	50	Salas	Munich	Luftansa

### Modification de PLUSIEURS colonnes

```
UPDATE Compagnie
SET nrue = 14, ville = DEFAULT
WHERE comp = 'SNCM';
```

SNCM	3	P. Paoli	Bastia	Corse Air
------	---	----------	--------	-----------



SNCM	14	P. Paoli	Paris	Corse Air
------	----	----------	-------	-----------

### Modification de plusieurs ENREGISTREMENTS

Au depart:

comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
LUFT	50	Salas	Munich	Luftansa
QUAN	1	Kangourou	Sydney	Quantas
SING	7	Camparols	Singapour	Singapore AL
SNCM	14	P. Paoli	Paris	Corse Air

```
UPDATE Compagnie SET nrue = 5 WHERE ville = 'Paris';
```

comp	nrue	rue	ville	nomComp
AF	5	Gambetta	Paris	Air France
LUFT	50	Salas	Munich	Luftansa
QUAN	1	Kangourou	Sydney	Quantas
SING	7	Camparols	Singapour	Singapore AL
SNCM	5	P. Paoli	Paris	Corse Air

Modification du numéro  
de la rue pour toutes les  
compagnies dont la ville  
est 'Paris'.

```
UPDATE Compagnie SET ville = 'Vichy' LIMIT 2;
```

comp	nrue	rue	ville	nomComp
AF	5	Gambetta	Vichy	Air France
LUFT	50	Salas	Vichy	Luftansa
QUAN	1	Kangourou	Sydney	Quantas
SING	7	Camparols	Singapour	Singapore AL
SNCM	5	P. Paoli	Paris	Corse Air

Modification des deux  
premières compagnies  
(par ordre de clé  
primaire, la valeur 'Vichy'  
à la colonne ville

### Ne pas respecter des CONTRAINTES

Modifications ne vérifiant pas les contraintes	
<pre>UPDATE Pilote SET brevet = 'PL-2' WHERE brevet = 'PL-1';</pre>	
<b>Erreur liée à la CLE PRIMAIRE</b> ERROR 1062 (23000): Duplicata du champ 'PL-2' pour la clef 'PRIMARY'	
<pre>UPDATE Pilote SET nom = NULL WHERE brevet = 'PL-1';</pre>	
<b>WARNING lié à la contrainte en ligne NOT NULL</b> warning   1048   Le champ 'nom' ne peut être vide (null)	
Contrairement à l'INSERT, un WARNING est généré (cf. SHOW WARNINGS) et le nom est modifié avec la CHAÎNE VIDE...	



Modifications ne vérifiant pas les contraintes	
<pre>UPDATE Pilote SET nom= 'Paul Durand' WHERE brevet = 'PL-1';</pre>	<p><b>ERREUR liée à l'UNICITE du nom</b></p> <p>ERROR 1062 (23000): Duplicata du champ 'Paul Durand' pour la clef 'un_nom'</p>
<pre>UPDATE Pilote SET compa = 'TOTO' WHERE brevet = 'PL-1';</pre>	<p><b>Erreur liée à la CLE ETRANGERE</b></p> <p>ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails  `pilote`, CONSTRAINT `fk_Pil_compa_Comp` FOREIGN KEY (`compa`) REFERENCES `compagni`</p>
<pre>UPDATE Compagnie SET comp = 'SIN2' WHERE comp = 'SING';</pre>	<p><b>Erreur liée à la CLE ETRANGERE : tentative de MAJ de la CLE PRIMAIRE dans la table PERE...</b></p> <p>ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint  `sio`.`pilote`, CONSTRAINT `fk_Pil_compa_Comp` FOREIGN KEY (`compa`) REFERENCES `</p>

### c) REMPLACEMENT d'un enregistrement (REPLACE)



L'instruction REPLACE consiste, comme son nom l'indique, à remplacer un enregistrement dans sa totalité (toutes ses colonnes). Il faut avoir les privilèges INSERT et DELETE sur la table.

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

```
REPLACE INTO Compagnie
VALUES ('QUAN', 33, 'Remplacer', 'Remplacer', 'Remplacer');
```



comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
LUFT	9	Salas	Munich	Luftansa
QUAN	1	Kangourou	Sydney	Quantas

Avant

comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
LUFT	9	Salas	Munich	Luftansa
QUAN	33	Remplacer	Remplacer	Remplacer

Après



L'enregistrement de clé primaire 'QUAN' a pu être remplacé car il n'existait aucune référence dans la table fils 'Pilote'

### d) SUPPRESSIONS D'ENREGISTREMENTS (DELETE)



L'instruction DELETE permet de supprimer un ou plusieurs enregistrements d'une table.

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM [nomBase.] nomTable
[WHERE (condition)]
[ORDER BY listeColonnes]
[LIMIT nbreLimite]
```

```
-- Effacer d'abord le FILS, puis le PERE
DELETE FROM Pilote WHERE compa = 'AF';
DELETE FROM Compagnie WHERE comp = 'AF';

-- Erreur fk
DELETE FROM Compagnie WHERE comp = 'SING';

--LIMIT
DELETE FROM Compagnie LIMIT 2;
```

- ☐ QUICK (pour les tables de type MyISAM) ne met pas à jour les index associés pour accélérer le traitement.
- ☐ WHERE sélectionne les lignes à supprimer dans la table.
  - Si aucune condition n'est précisée, toutes les lignes seront détruites.
  - Si la condition ne sélectionne aucune ligne, aucun enregistrement ne sera supprimé.
- ☐ ORDER BY réalise un tri des enregistrements qui seront effacés dans cet ordre.