

Introduction aux bases de données

Le Langage de Définition de Données

Hamida SEBA LAGRAA

Hamida.lagraa@univ-lyon1.fr

Organisation

- ❑ 4 semaines comportant en tout:
 - 1 séance de cours
 - 3 séances de TP(sur 4h chacune)
 - 1 Séances de TP noté (2h Evaluation)
- ❑ Evaluation
 - 1 TP noté et éventuellement une interro

Objectifs du cours

- ❑ Implémenter une base de données relationnelle

- ❑ Modifier une base de données

 - SQL LDD

- ❑ Manipuler des données

 - SQL LMD

Le langage SQL

☐ SELECT

Interrogation de données (LID)

☐ INSERT : insertion d'un ou plusieurs tuples

☐ UPDATE : modification

☐ DELETE : suppression

Langage de Manipulation de Données
(LMD)

☐ CREATE : création des tables, index, vues

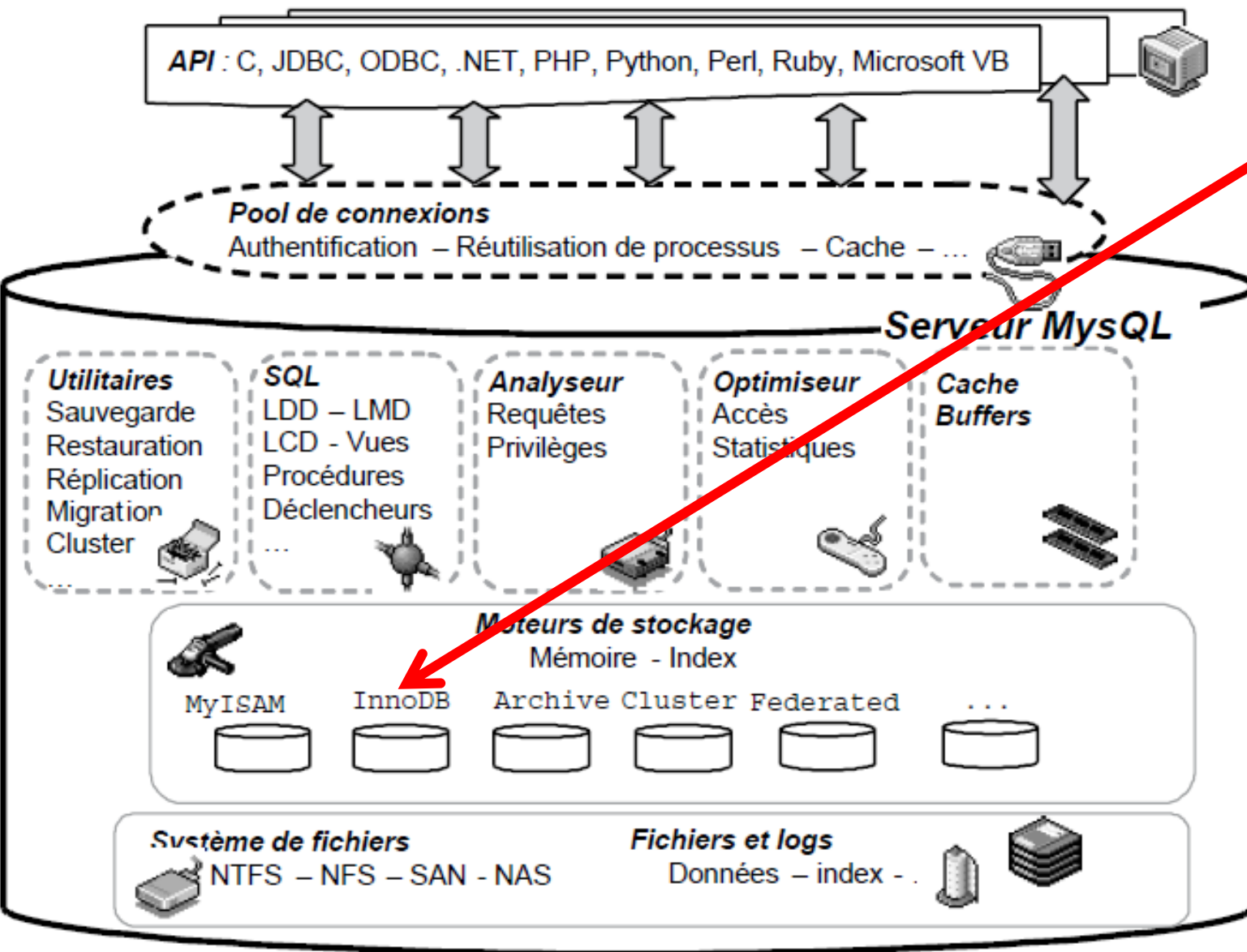
☐ ALTER : modification

☐ DROP : suppression

Langage de Définition de Données
(LDD)

Le SGBD MYSQL (ou MariaDB)

Moteur qui implémente le modèle relationnel
C'est le moteur par défaut depuis la version 5.7



A partir du schéma relationnel d'une BD

1

Personne		
<u>Num_personne</u>	int	<pk>
Nom	varchar(30)	
Prenom	varchar(30)	
Adresse	varchar(200)	

2

Vehicule		
<u>Immat</u>	varchar(50)	<pk>
Num_personne	int	<fk>
marque	varchar(30)	
couleur	varchar(20)	
Puissance	int	

Contrainte
d'intégrité
référentielle

```
SET default_storage_engine= InnoDB;  
drop table if exists Vehicule;  
drop table if exists Personne;
```

1

```
create table Personne  
(  
    numPersonne    int,  
    nom            varchar(30),  
    prenom         varchar(30),  
    adresse        varchar(200),  
    Constraint PK_Personne  
primary key (numPersonne)  
) ENGINE= InnoDB ;
```

2

```
create table Vehicule  
(  
    immat          varchar(50),  
    numPersonne    int,  
    marque         varchar(30),  
    couleur        varchar(20),  
    puissance      int,  
    constraint PK_Vehicule primary key  
        (immat),  
    constraint FK_Vehicule_Personne  
        foreign key (numPersonne)  
        references Personne (numPersonne)  
on delete restrict on update restrict  
) ENGINE= InnoDB ;
```

3

LDD: Create table

```
CREATE TABLE [IF NOT EXISTS] [nomBase.]nomTable  
( colonne1 type1 [NOT NULL | NULL] [DEFAULT valeur1] [COMMENT 'chaine1']  
[, colonne2 type2 [NOT NULL | NULL] [DEFAULT valeur2] [COMMENT 'chaine2']]  
[CONSTRAINT nomContrainte1 typeContrainte1] ...)  
[ENGINE= InnoDB | MyISAM | ...];
```

- ❑ IF NOT EXISTS : éviter une erreur « table existe déjà »
- ❑ COMMENT : (60 caractères) permet de commenter une colonne.
- ❑ ENGINE : définit le type de table (par défaut InnoDB, qui implemente le modèle relationnel).
- ❑ Choisir le moteur lors de la définition de la table ou dans le fichier de configuration ou à défaut dans la session avec **SET default_storage_engine= InnoDB**

CREATE TABLE: Example

```
CREATE TABLE dept
(deptno int,
  dname VARCHAR(14),
  loc VARCHAR(13));
```

```
DESCRIBE dept
```

Name	Null?	Type
-----	-----	-----
DEPTNO	NOT NULL	NUMBER (2)
DNAME		VARCHAR2 (14)
LOC		VARCHAR2 (13)

CREATE TABLE: Exemple

```
CREATE TABLE vehicule (  
  n_vehicule      VARCHAR(10) ,  
  type_vehicule   VARCHAR(1) ,  
  kilometrage     INTEGER ,  
  capacite        INTEGER ,  
  CONSTRAINT pk_vehicule PRIMARY KEY(n_vehicule)  
) engine INNODB;
```

Creation d'une table à partir d'une autre table

CREATE TABLE table_a_creer

AS SELECT * | colonnes FROM table_source | jointure
[WHERE condition];

```
CREATE TABLE deptRA AS SELECT deptno,dname FROM dept
WHERE upper(loc)='RHONES-ALPES' ;
```

```
CREATE TABLE deptRA_col_renome AS SELECT deptno NUM_D,
dname Nom_D
FROM dept
WHERE upper(loc)='RHONES-ALPES' ;
```

Types de données

- ❑ caractères (CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT) ;
- ❑ valeurs numériques (TINYINT, SMALLINT, MEDIUMINT, INT, INTEGER, BIGINT, FLOAT, DOUBLE, REAL, DECIMAL, NUMERIC, et BIT) ;
- ❑ date/heure (DATE, DATETIME, TIME, YEAR, TIMESTAMP) ;
- ❑ données binaires (BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB) ;
- ❑ énumérations (ENUM, SET).

Types de données caractères

Type	Description	Commentaire pour une colonne
<code>CHAR (n)</code> [<code>BINARY</code> <code>ASCII</code> <code>UNICODE</code>]	Chaîne fixe de n octets ou caractères.	Taille fixe (maximum de 255 caractères).
<code>VARCHAR (n)</code> [<code>BINARY</code>]	Chaîne variable de n caractères ou octets.	Taille variable (maximum de 65 535 caractères).
<code>BINARY (n)</code>	Chaîne fixe de n octets.	Taille fixe (maximum de 255 octets).
<code>VARBINARY (n)</code>	Chaîne variable de n octets.	Taille variable (maximum de 255 octets).
<code>TINYTEXT (n)</code>	Flot de n octets.	Taille fixe (maximum de 255 octets).
<code>TEXT (n)</code>	Flot de n octets.	Taille fixe (maximum de 65 535 octets).
<code>MEDIUMTEXT (n)</code>	Flot de n octets.	Taille fixe (maximum de 16 mégaoctets).
<code>LONGTEXT (n)</code>	Flot de n octets.	Taille fixe (maximum de 4,29 gigaoctets).

Types de données numériques

Type	Description
BIT [(n)]	Ensemble de n bits. Taille de 1 à 64 (par défaut 1).
TINYINT [(n)] [UNSIGNED] [ZEROFILL]	Entier (sur un octet) de -128 à 127 signé, 0 à 255 non signé.
BOOL et BOOLEAN	Synonymes de TINYINT(1), la valeur zéro est considérée comme fausse. Le non-zéro est considéré comme vrai. Dans les prochaines versions, le type <i>boolean</i> , comme le préconise la norme SQL, sera réellement pris en charge.
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 $-3.4 \cdot 10^{+38}$ à $-1.1 \cdot 10^{-38}$, 0, signé, et de $1.1 \cdot 10^{-38}$ à $3.4 \cdot 10^{+38}$ non signé.
DOUBLE [(n[,p])] [UNSIGNED] [ZEROFILL]	Flottant (sur 8 octets) p désigne la précision double (jusqu'à 15 décimales) de $-1.7 \cdot 10^{+308}$ à $-2.2 \cdot 10^{-308}$, 0, signé, et de $2.2 \cdot 10^{-308}$ à $1.7 \cdot 10^{+308}$ 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.

Type de données Date

Type	Description	Commentaire pour une colonne
DATE	Dates du 1 ^{er} 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.
TIMESTAMP	Instants du 1 ^{er} Janvier 1970 0 h 0 minute 0 seconde à l'année 2037.	Estampille sur 4 octets (au format 'YYYY-MM-DD HH:MM:SS') ; mise à jour à chaque modification sur la table.

- ❑ NOW() et SYSDATE() retournent la date et l'heure courantes.
- ❑ Les fonctions CURRENT_TIMESTAMP(), CURRENT_DATE() et CURRENT_TIME(), UTC_TIME(), renseignent sur l'instant, la date, l'heure et l'heure GMT de la session en cours.

Synonymes et alias

- ❑ INT est synonyme de INTEGER.
- ❑ DOUBLE PRECISION et REAL sont synonymes de DOUBLE.
- ❑ DEC NUMERIC et FIXED sont synonymes de DECIMAL.
- ❑ SERIAL est un alias pour BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE.

Destruction d'une table

DROP TABLE [IF EXISTS]

[nomBase.] nomTable1 [, [nomBase2.] nomTable2, ...]

[RESTRICT | CASCADE]

- ❑ RESTRICT et CASCADE : pas encore opérationnels. Le premier permet de vérifier qu'aucun autre élément n'utilise la table. Le second répercute la destruction à tous les éléments référencés.

Insertion d'enregistrement

```
INSERT INTO [nomBase.] nomTable [(nomColonne,...)]  
VALUES ({expression / DEFAULT},...),(...),...
```

- ❑ *Insérer une ligne avec des valeurs pour toutes les colonnes*

```
INSERT INTO Dept  
VALUES (74,'Haute-Savoie','rhones-Alpes');
```

- ❑ *Insérer une ligne avec des valeurs uniquement pour certaines colonnes*

```
INSERT INTO Dept (deptno,dname)  
VALUES (74,'Haute-Savoie');  
Ou  
INSERT INTO Dept VALUES (74,'Haute-Savoie',NULL);
```

Mises à jour : modification d'un champ

```
UPDATE [nomBase.] nomTable  
SET colonne1 = expression1 | (requête_SELECT) | DEFAULT  
[,colonne2 = expression2...]  
[WHERE (condition)]
```

- ❑ La clause SET actualise une colonne en lui affectant une expression (valeur, valeur par défaut, calcul ou résultat d'une requête).

```
UPDATE Dept  
SET dname=upper(dname) WHERE deptno=74;
```

Suppression d'enregistrements

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

supprime un ou plusieurs enregistrements d'une table

```
DELETE FROM Dept LIMIT 2;  
DELETE FROM Dept WHERE deptno=74;
```

```
TRUNCATE [TABLE] [nomBase.] nomTable;
```

supprime tous les enregistrements d'une table et libère éventuellement l'espace de stockage utilisé par la table

```
TRUNCATE TABLE Dept;
```

Séquences (ou génération automatique de clés primaires)

- ❑ Offre la possibilité de générer automatiquement des valeurs numériques qui serviront à composer des clés primaires.

```
DROP TABLE if exists EMP ;
CREATE TABLE EMP(
  NumEmp INT AUTO_INCREMENT,
  Nom VARCHAR(12),
  CONSTRAINT pk_emp
  PRIMARY KEY (NumEmp));
INSERT INTO EMP(Nom) VALUES ('Dupond');
INSERT INTO EMP VALUES (NULL,'Durand');
INSERT INTO EMP VALUES (0,'Pelltier');
SELECT * FROM EMP;
```

🔑 NumEmp	Nom
1	Dupond
2	Durand
3	Pelltier

Séquences: modification

```
ALTER TABLE EMP AUTO_INCREMENT = 100;  
INSERT INTO EMP VALUES (0, 'Bronchier');  
SELECT * FROM EMP;  
SELECT LAST_INSERT_ID();|
```

NumEmp	Nom
1	Dupond
2	Durand
3	Peltier
100	Bronchier

LAST_INSERT_ID()
100

Contraintes

CONSTRAINT *nomContrainte*

UNIQUE (*colonne1* [,*colonne2*]...)

PRIMARY KEY (*colonne1* [,*colonne2*]...)

FOREIGN KEY (*colonne1* [,*colonne2*]...)

REFERENCES *nomTablePere* [(*colonne1* [,*colonne2*]...)]

[ON DELETE {CASCADE | SET NULL|RESTRICT|NO ACTION}]

[ON UPDATE {CASCADE | SET NULL|RESTRICT|NO ACTION}]

CHECK (*condition*)

Contraintes: Exemple (NOT NULL et primary key)

```
CREATE TABLE emp
(empno    INT,
ename     VARCHAR(10) NOT NULL,
job       VARCHAR(9) ,
mgr       int,
hiredate  DATE,
sal       FLOAT,
comm      int,
deptno    int NOT NULL,
CONSTRAINT PK_Emp PRIMARY KEY (EMPNO) ) ;
```

Contrainte PRIMARY KEY

Clé primaire composée de plusieurs colonnes doit être définie au niveau de la table

```
CREATE TABLE dept(  
  deptno    INT,  
  dname     VARCHAR(12) ,  
  loc       VARCHAR(13) ,  
  CONSTRAINT dept_dname_uk UNIQUE (dname) ,  
  CONSTRAINT pk_dept PRIMARY KEY (deptno,dname)) ;
```


Contrainte UNIQUE

DEPT  **Contrainte UNIQUE**

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

 **Insert into**

50	SALES	DETROIT
60		BOSTON

 impossible
(DNAME:SALES
existe déjà)

 permis

Contrainte UNIQUE

```
CREATE TABLE dept(  
    deptno    INT,  
    dname     VARCHAR(14) UNIQUE,  
    loc       VARCHAR(13) ,  
    );
```

```
CREATE TABLE dept(  
    deptno    INT,  
    dname     VARCHAR(14) ,  
    loc       VARCHAR(13) ,  
    CONSTRAINT UK_dname UNIQUE (dname)) ;
```

Contrainte FOREIGN KEY

**[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}]**

- ❑ FOREIGN KEY: Définit la colonne dans la table pour une contrainte de table
- ❑ REFERENCES: Identifie la table liée et sa colonne
- ❑ ON DELETE CASCADE: Permet la suppression des lignes de la table liée (clé primaire) et supprime en même temps les lignes qui référencent les valeurs de la table liée (clé étrangère)
- ❑ ON DELETE SET NULL: Permet la suppression des lignes de la table liée (clé primaire) et remplace en même temps les clés étrangères associées par des NULL.
- ❑ ON DELETE RESTRICT : interdit la suppression du parent si au moins un enfant existe
- ❑ ON DELETE No ACTION : même effet que RESTRICT
- ❑ ON UPDATE CASCADE/SET NULL: Propager ou pas une mise à jour du père

Contrainte FOREIGN KEY

EMP		
<u>EMPNO</u>	int(11)	<pk>
ENAME	varchar(10)	
JOB	varchar(9)	
HIREDATE	date	
SAL	decimal(7,2)	
COMM	decimal(7,2)	
DEPTNO	int(11)	<ak2, fk>

FK_EMP_DEPT

DEPT		
<u>DEPTNO</u>	int(11)	<pk>
DNAME	varchar(50)	
LOC	varchar(50)	

```
CREATE TABLE dept(  
    deptno          INT,  
    dname   VARCHAR(14),  
    loc     VARCHAR(13),  
    );
```

```
CREATE TABLE emp  
(empno      INT,  
  ename      VARCHAR(10) NOT NULL,  
  job        VARCHAR(9),  
  mgr        int,  
  hiredate   DATE,  
  sal        FLOAT,  
  comm       int,  
  deptno     int,  
  CONSTRAINT fk_emp_dept FOREIGN KEY (deptno)  
  REFERENCES dept(deptno));
```

Contrainte FOREIGN KEY

EMP		
<u>EMPNO</u>	int(11)	<pk>
ENAME	varchar(10)	
JOB	varchar(9)	
HIREDATE	date	
SAL	decimal(7,2)	
COMM	decimal(7,2)	
DEPTNO	int(11)	<ak2, fk>

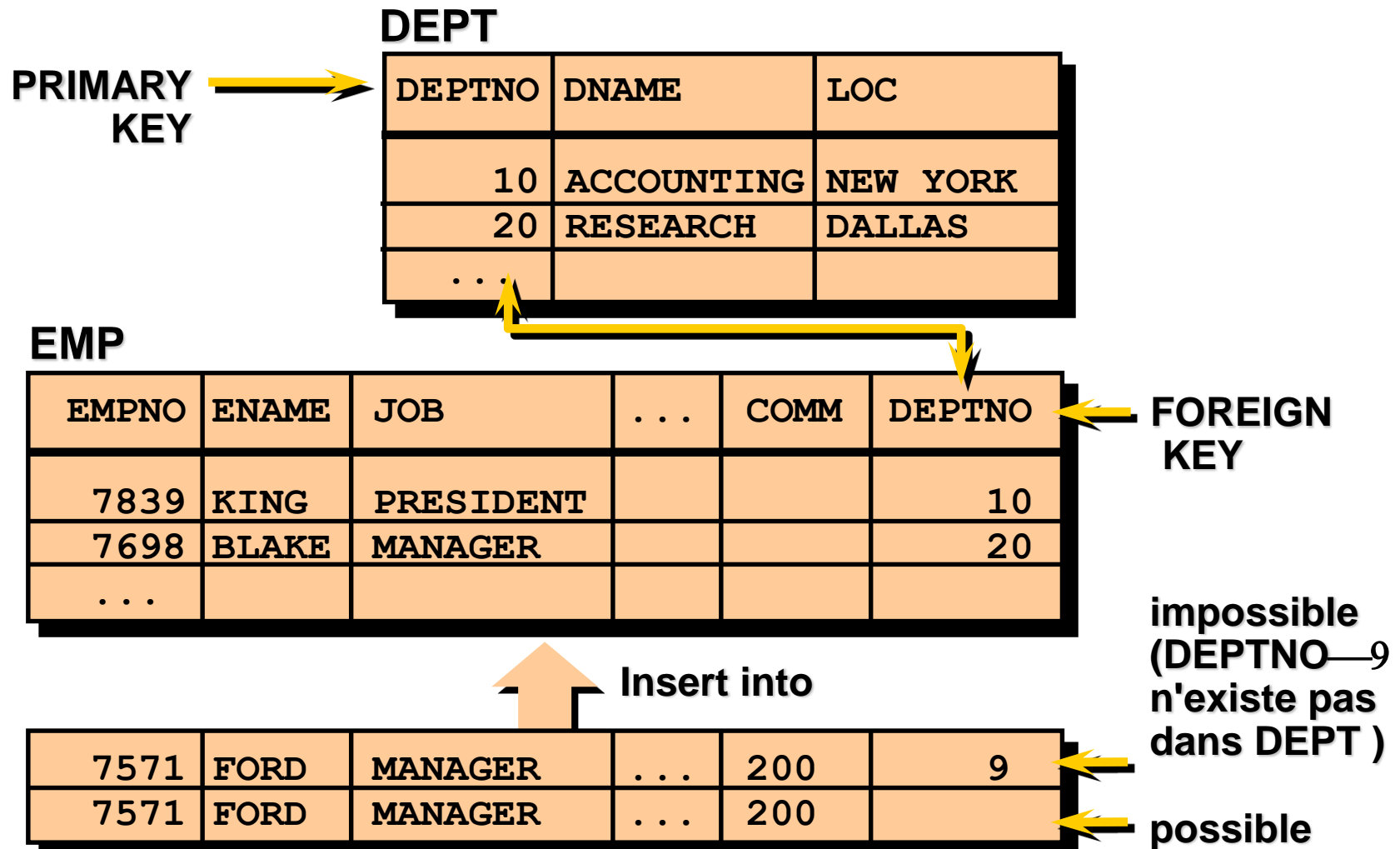
FK_EMP_DEPT

DEPT		
<u>DEPTNO</u>	int(11)	<pk>
DNAME	varchar(50)	
LOC	varchar(50)	

```
CREATE TABLE dept(  
    deptno          INT,  
    dname   VARCHAR(14),  
    loc     VARCHAR(13),  
    );
```

```
CREATE TABLE emp  
(empno      INT,  
  ename      VARCHAR(10) NOT NULL,  
  job        VARCHAR(9),  
  mgr        int default ,  
  hiredate   DATE,  
  sal        FLOAT,  
  comm       int,  
  deptno     int REFERENCES dept(deptno));
```

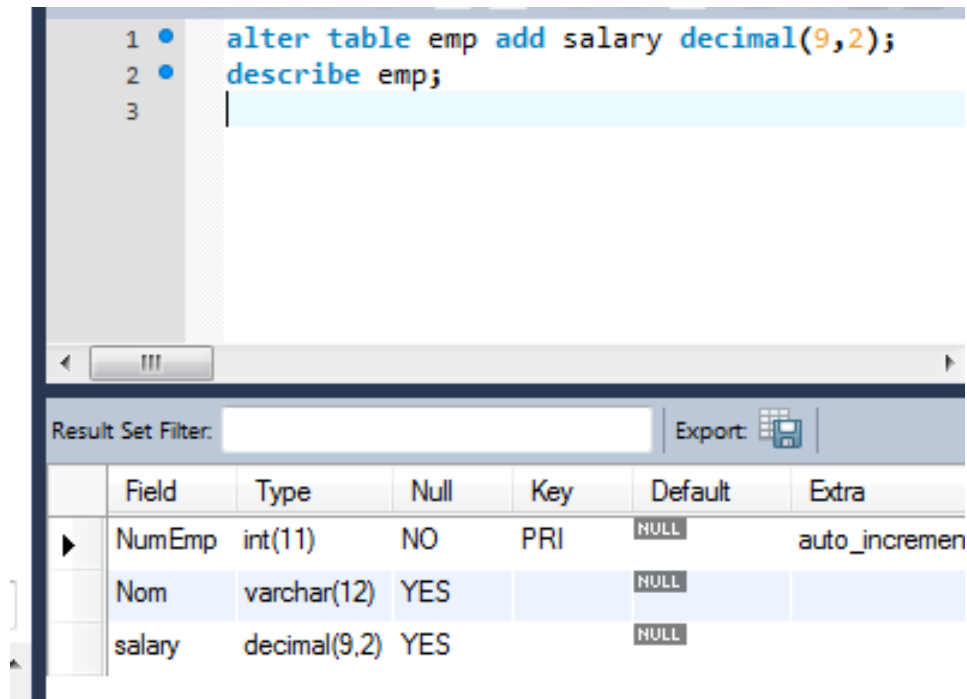
Contrainte FOREIGN KEY



Evolution de schémas

Ajout de colonnes, de contraintes, etc.

```
ALTER TABLE [nomBase].nomTable ADD  
{nomColonne typeMySQL [NOT NULL | NULL] [DEFAULT valeur]  
| INDEX [nomIndex] [typeIndex] (nomColonne1,...)  
| CONSTRAINT nomContrainte typeContrainte }
```



The screenshot shows a database management interface. At the top, a SQL editor contains the following commands:

```
1 • alter table emp add salary decimal(9,2);  
2 • describe emp;  
3 |
```

Below the editor, a "Result Set Filter:" field and an "Export:" button are visible. The main area displays the structure of the 'emp' table as a table with the following data:

	Field	Type	Null	Key	Default	Extra
▶	NumEmp	int(11)	NO	PRI	NULL	auto_incremen
	Nom	varchar(12)	YES		NULL	
	salary	decimal(9,2)	YES		NULL	

Evolution de schémas

Suppression de colonnes, de contraintes, etc.

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

```
ALTER TABLE emp DROP COLUMN job;
```


Evolution de schémas

modification de colonnes

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

FIRST|AFTER nomColonne: permettent de repositionner la colonne

```
ALTER TABLE dept MODIFY COLUMN loc varchar(30);
```

Evolution de schémas

Renommer des colonnes/ Renommer une table

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

```
ALTER TABLE dept CHANGE loc region VARCHAR(30);
```

```
RENAME [nomBase.]ancienNomTable TO [nomBase.] nouveauNomTable  
[, [nomBase.] ancienNom2 TO [nomBase.]nouveauNom2];
```

```
RENAME Dept TO Departements;
```