

Chapitre#5

SQL part.1

SQL: introduction

- Langage développé par IBM dans les années 1970.
- Norme ISO en 1987.
- Devenu depuis le langage « standard » pour interagir avec un SGBD relationnel.
- Plus d'une centaine de SGBDR utilisent SQL.

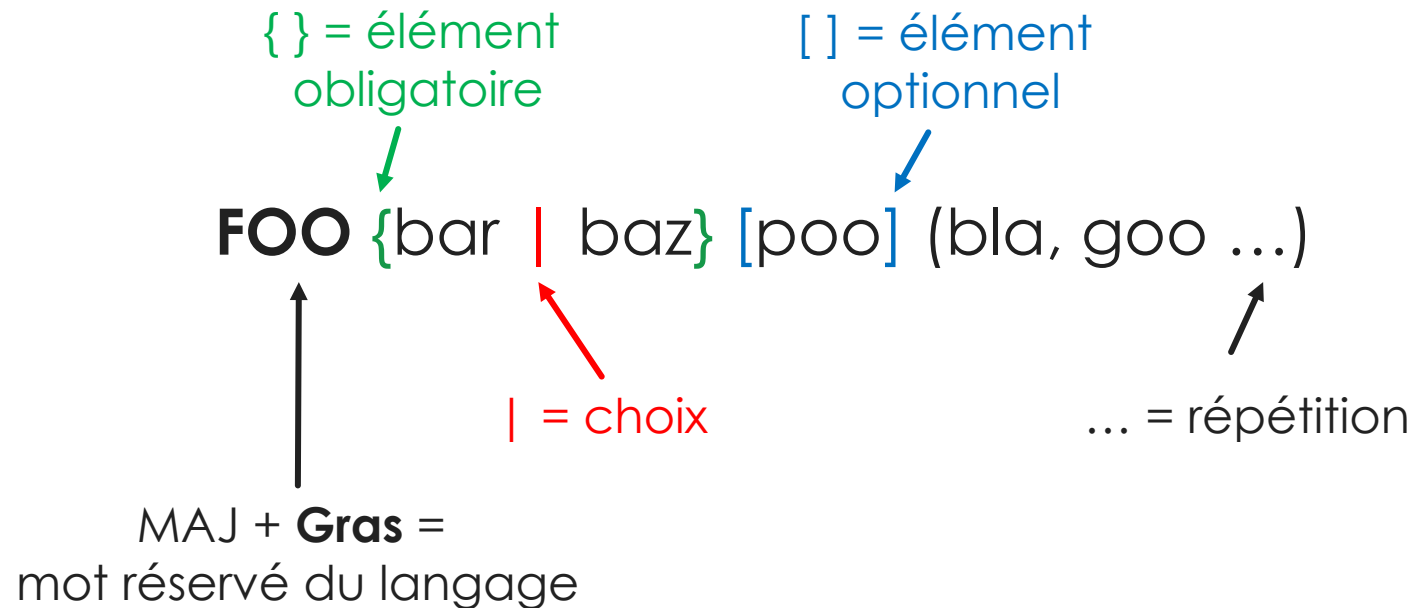
SQL: terminologie

La terminologie utilisée par les notions mathématiques derrière le langage et le langage lui-même changent (un peu).

Modèle relationnel	SQL
Relation	Table
Attribut	Colonne
Domaine	Type
Tuple	Ligne

SQL: syntaxe

Pour décrire la syntaxe des commandes, la notation Backus Naur Form (BNF) sera utilisée.



SQL: syntaxe

Un SGBD relationnel possède deux langages :

- Le **Langage de Définition de Données (LDD)** qui permet de déclarer les structures de la base de données (tables, colonnes, *etc.*)
- Le **Langage de Manipulation de Données (LMD)** qui permet d'interagir avec la base de données (interrogation, modification, *etc.*)

→ SQL permet de faire les deux.

SQL: syntaxe

Le cours d'aujourd'hui !

Un SGBD relationnel possède deux langages :

- Le **Langage de Définition de Données (LDD)** qui permet de déclarer les structures de la base de données (tables, colonnes, etc.)
- Le **Langage de Manipulation de Données (LMD)** qui permet d'interagir avec la base de données (interrogation, modification, etc.)

→ SQL permet de faire les deux.

SQL: le LDD, créer une table

Pour créer une table :

```
CREATE TABLE nomTable (  
    {nomColonne type [NOT NULL][UNIQUE]  
      [DEFAULT valeur][CHECK condition][, ...]}  
  
    {[CONSTRAINT nomPK] PRIMARY KEY (listeColonnes),}  
  
    {[CONSTRAINT nomUN] UNIQUE (listeColonnes),}[, ...]}  
  
    {[CONSTRAINT nomFK] FOREIGN KEY (listeColonnes)  
      REFERENCES tabbleParent[(listeColonnes)][, ...]}  
  
    {[CHECK (condition)][, ...]}  
);
```

SQL: le LDD, créer une table

```
CREATE TABLE nomTable (  
    {nomColonne type [NOT NULL][UNIQUE]  
    [DEFAULT valeur][CHECK condition][,...]}  
);
```

- **NOT NULL** stipule que l'attribut doit toujours avoir une valeur.
- **DEFAULT** peut donner une valeur par défaut.
- **UNIQUE** identifie une colonne ayant des valeurs uniques.
- **CHECK** permet de vérifier une contrainte sur le domaine d'un attribut.

SQL: le LDD, créer une table

```
[[CONSTRAINT nomPK] PRIMARY KEY (listeColonnes),]
```

→ Définit les colonnes faisant partie de la clé primaire de la table. Les attributs de la clé primaire doivent être **NOT NULL**.

SQL: le LDD, créer une table

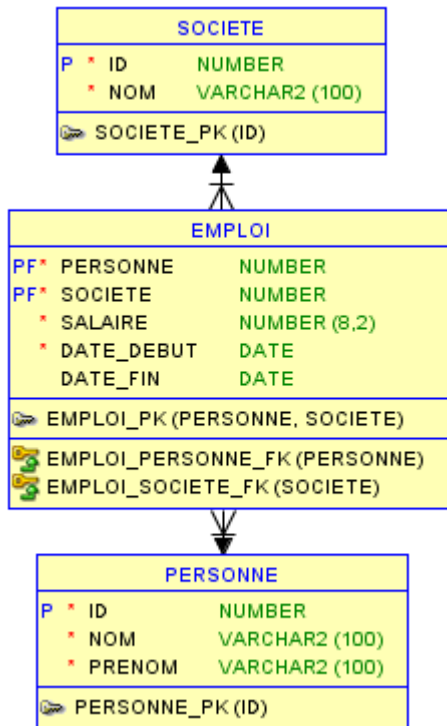
```
[[CONSTRAINT nomFK] FOREIGN KEY (listeColonnes)  
  REFERENCES tabbleParent[(listeColonnes)]][, ...]
```

→ Définit les colonnes qui sont des clés étrangères: de quelle autre table elles sont la clé primaire.

SQL: le LDD, créer une table

Exemple

(créer la table *EMPLOI* du modèle relationnel donné)



```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE,  
  
    CONSTRAINT EMPLOI_PK PRIMARY KEY  
        (PERSONNE, SOCIETE) ,  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
        (PERSONNE) REFERENCES PERSONNE (ID) ,  
  
    CONSTRAINT EMP_SOC_FK FOREIGN KEY  
        (SOCIETE) REFERENCES SOCIETE (ID)  
  
);
```

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE,  
  
    CONSTRAINT EMPLOI_PK PRIMARY KEY  
        (PERSONNE, SOCIETE),  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
        (PERSONNE) REFERENCES PERSONNE (ID),  
  
    CONSTRAINT EMP_SOC_FK FOREIGN KEY  
        (SOCIETE) REFERENCES SOCIETE (ID)  
);
```

Contrainte
d'intégrité:
donnée requise.

SQL: le LDD, les contraintes


```
CREATE TABLE EMPLOI (  
  PERSONNE NUMBER NOT NULL,  
  SOCIETE NUMBER NOT NULL,  
  SALAIRE NUMBER(8,2) NOT NULL,  
  DATE_DEBUT DATE NOT NULL,  
  DATE_FIN DATE,  
  
  CONSTRAINT EMPLOI_PK PRIMARY KEY  
    (PERSONNE, SOCIETE),  
  
  CONSTRAINT EMP_PERS_FK FOREIGN KEY  
    (PERSONNE) REFERENCES PERSONNE (ID),  
  
  CONSTRAINT EMP_SOC_FK FOREIGN KEY  
    (SOCIETE) REFERENCES SOCIETE (ID)  
);
```

Contrainte
d'intégrité:
donnée requise.

Possible de
créer une
contrainte de
domaine sur
l'attribut salaire.

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE,  
  
    CONSTRAINT EMPLOI_PK PRIMARY KEY  
        (PERSONNE, SOCIETE),  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
        (PERSONNE) REFERENCES PERSONNE (ID),  
  
    CONSTRAINT EMP_SOC_FK FOREIGN KEY  
        (SOCIETE) REFERENCES SOCIETE (ID)  
);
```



```
CHECK (  
    SALAIRE BETWEEN 0  
    AND 150000.00  
)
```

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE,  
  
    CONSTRAINT EMPLOI_PK PRIMARY KEY  
    (PERSONNE, SOCIETE),  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
    (PERSONNE) REFERENCES PERSONNE (ID),  
  
    CONSTRAINT EMP_SOC_FK FOREIGN KEY  
    (SOCIETE) REFERENCES SOCIETE (ID)  
  
);
```

Sur quel autre
attribut est-il
pertinent de
créer une
contrainte de
domaine ?

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE,  
  
    CONSTRAINT EMPLOI_PK PRIMARY KEY  
        (PERSONNE, SOCIETE),  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
        (PERSONNE) REFERENCES PERSONNE (ID),  
  
    CONSTRAINT EMP_SOC_FK FOREIGN KEY  
        (SOCIETE) REFERENCES SOCIETE (ID)  
);
```

```
CHECK (  
    DATE_DEBUT <  
        DATE_FIN  
)
```



SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE,  
    CONSTRAINT EMPLOI_PK PRIMARY KEY  
        (PERSONNE, SOCIETE),  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
        (PERSONNE) REFERENCES PERSONNE (ID),  
    CONSTRAINT EMP_SOC_FK FOREIGN KEY  
        (SOCIETE) REFERENCES SOCIETE (ID)  
);
```

Intégrité
référentielle:
PRIMARY KEY

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE,  
    CONSTRAINT EMPLOI_PK PRIMARY KEY  
        (PERSONNE, SOCIETE),  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
        (PERSONNE) REFERENCES PERSONNE (ID),  
    CONSTRAINT EMP_SOC_FK FOREIGN KEY  
        (SOCIETE) REFERENCES SOCIETE (ID)  
);
```

- L'instruction **PRIMARY KEY** s'assure que la clé primaire contienne des valeurs uniques et non nulles.
- Une seule clé primaire possible par table. Les autres colonnes qui doivent contenir des valeurs uniques sont identifiées par la contrainte **UNIQUE**.

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE,  
  
    CONSTRAINT EMPLOI_PK PRIMARY KEY  
    (PERSONNE, SOCIETE),  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
    (PERSONNE) REFERENCES PERSONNE (ID),  
  
    CONSTRAINT EMP_SOC_FK FOREIGN KEY  
    (SOCIETE) REFERENCES SOCIETE (ID)  
);
```

Intégrité
référentielle:
FOREIGN KEY

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE,  
  
    CONSTRAINT EMPLOI_PK PRIMARY KEY  
        (PERSONNE, SOCIETE),  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
        (PERSONNE) REFERENCES PERSONNE (ID),  
  
    CONSTRAINT EMP_SOC_FK FOREIGN KEY  
        (SOCIETE) REFERENCES SOCIETE (ID)  
);
```

L'instruction ***FOREIGN KEY*** permet de s'assurer que la valeur de ses attributs correspond bien à une clé qui existe dans la table parent (= table référencée).

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE,  
  
    CONSTRAINT EMPLOI_PK PRIMARY KEY  
    (PERSONNE, SOCIETE),  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
    (PERSONNE) REFERENCES PERSONNE (ID),  
  
    CONSTRAINT EMP_SOC_FK FOREIGN KEY  
    (SOCIETE) REFERENCES SOCIETE (ID)  
);
```

- Toute opération de création ou de modification qui tente de donner une valeur de clé étrangère n'existant pas dans la table d'origine est rejetée.
- Qu'arrive-t-il si la clé primaire d'une table d'origine référencée par une autre table est modifiée ou supprimée ?

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    ...  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
    (PERSONNE) REFERENCES PERSONNE (ID) ON  
    {DELETE|UPDATE} « CLAUSE »,  
  
    ...  
);
```

Les clauses
ON DELETE et
ON UPDATE
permettent de spécifier
ce qui doit être fait
lorsqu'une valeur de
clé primaire de la table
d'origine est supprimée
ou modifiée.

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    ...  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
    (PERSONNE) REFERENCES PERSONNE (ID) ON  
    {DELETE | UPDATE} CASCADE,  
  
    ...  
);
```

Il existe plusieurs possibilités pour ces deux clauses :

- **CASCADE** :
supprime toutes les lignes de la table référencée par la valeur de la clé primaire en question.

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    ...  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
    (PERSONNE) REFERENCES PERSONNE (ID) ON  
    {DELETE|UPDATE} SET NULL,  
  
    ...  
);
```

Il existe plusieurs possibilités pour ces deux clauses :

- ***SET NULL*** :
remplace la valeur de la clé étrangère par ***NULL*** pour toutes les lignes de la table référencée.

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    ...  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
    (PERSONNE) REFERENCES PERSONNE (ID) ON  
    {DELETE|UPDATE} SET DEFAULT,  
  
    ...  
);
```

Il existe plusieurs possibilités pour ces deux clauses :

- ***SET DEFAULT*** : remplace la valeur de la clé étrangère par la valeur par défaut pour toutes les lignes de la table référencée.

SQL: le LDD, les contraintes

```
CREATE TABLE EMPLOI (  
    ...  
  
    CONSTRAINT EMP_PERS_FK FOREIGN KEY  
    (PERSONNE) REFERENCES PERSONNE (ID) ,  
  
    ...  
);
```

L'action par défaut si aucune clause n'est spécifiée est **le rejet de l'opération sur la table parent**.

SQL: le LDD, modifier une table

Qu'arrive-t-il si pour une raison ou pour une autre il faut :

- ajouter / supprimer une colonne d'une table ;
- ajouter / supprimer une contrainte ;
- ajouter / supprimer une valeur par défaut à une colonne ?

Remarque : la plupart du temps, pour toute modification, il faudra, au préalable supprimer puis ajouter!

SQL: le LDD, modifier une table

Ajouter une colonne à une table

```
ALTER TABLE nomTable ADD (  
    nomColonne typeColonne [contrainteColonne] [,...]  
);
```

Supprimer une colonne d'une table

```
ALTER TABLE nomTable DROP (nomColonne[,...]);
```

Que se passe-t-il si je souhaite ajouter une colonne qui doit avoir une contrainte *NOT NULL* à une table qui contient déjà des données ?

SQL: le LDD, modifier une table

Ajouter une colonne à une table

```
ALTER TABLE nomTable ADD (  
    nomColonne typeColonne [contrainteColonne] [,...]  
);
```

Supprimer une colonne d'une table

```
ALTER TABLE nomTable DROP (nomColonne[,...]);
```

Que se passe-t-il si je souhaite ajouter une colonne qui doit avoir une contrainte *NOT NULL* à une table qui contient déjà des données ? **L'opération est rejetée!!**

SQL: le LDD, modifier une table

(1) ajouter la colonne sans la contrainte ***NOT NULL***.

```
ALTER TABLE maTable ADD (maColonne VARCHAR2 (10)) ;
```

(2) modifier la table pour insérer des données à la nouvelle colonne pour chaque ligne déjà existante. (on verra ces ordres SQL au prochain cours ;-)

(3) modifier la colonne pour y ajouter la contrainte ***NOT NULL***.

```
ALTER TABLE maTable MODIFY (maColonne NOT NULL) ;
```

SQL: le LDD, modifier une table

Modifier le type d'une colonne

```
ALTER TABLE nomTable MODIFY nomColonne nouveauType;
```

L'opération sera rejetée si la colonne contient des données!

→ nouvelle colonne, mise à jour de la nouvelle colonne avec les nouvelles données, suppression de l'ancienne colonne, renommage de la nouvelle colonne.

Modifier le nom d'une colonne

```
ALTER TABLE nomTable RENAME COLUMN nomColonne TO  
nouveauNomColonne;
```

SQL: le LDD, modifier une table

Modifier une valeur par défaut

```
ALTER TABLE nomTable MODIFY nomColonne DEFAULT  
nouvelleValeur;
```

Supprimer une valeur par défaut

```
ALTER TABLE nomTable MODIFY nomColonne DEFAULT NULL;
```


SQL: le LDD, modifier une table

Ajouter une contrainte

```
ALTER TABLE nomTable ADD CONSTRAINT nomContrainte  
PRIMARY KEY ([listeColonnes]);
```

```
ALTER TABLE nomTable ADD CONSTRAINT nomContrainte  
FOREIGN KEY (listeColonnes) REFERENCES  
tabbleParent([listeColonnes]);
```

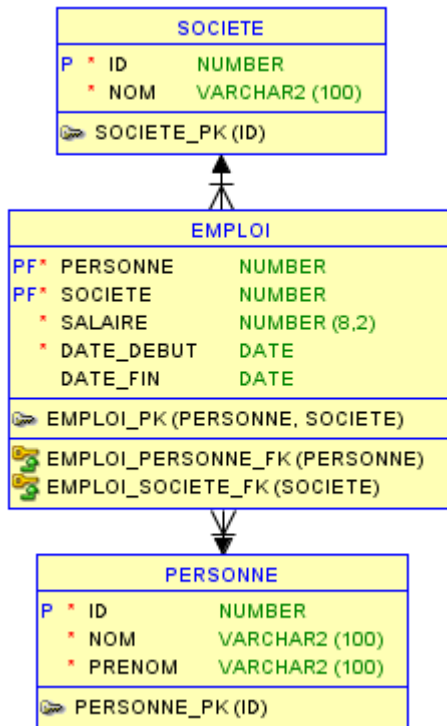
```
ALTER TABLE nomTable ADD CONSTRAINT nomContrainte  
UNIQUE (listeColonnes);
```

```
ALTER TABLE nomTable ADD CONSTRAINT nomContrainte  
CHECK (condition);
```

SQL: le LDD, modifier une table

Exemple

(créer la table *EMPLOI* du modèle relationnel donné)



```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE NUMBER NOT NULL,  
    SALAIRE NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE  
);
```

```
ALTER TABLE EMPLOI ADD CONSTRAINT EMPLOI_PK PRIMARY  
KEY (PERSONNE, SOCIETE);
```

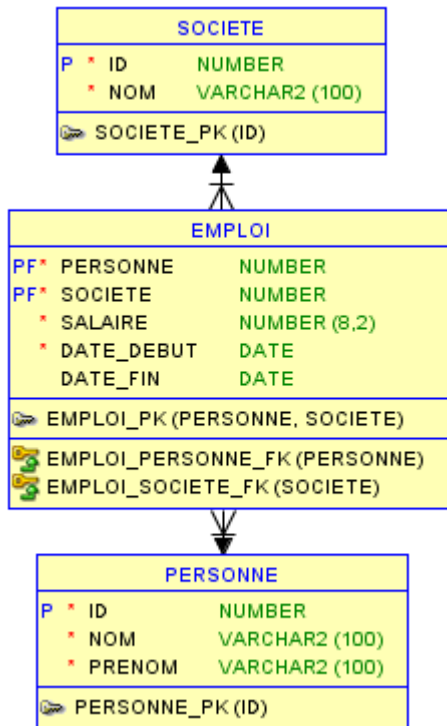
```
ALTER TABLE ADD CONSTRAINT EMP_PERS_FK FOREIGN KEY  
(PERSONNE) REFERENCES PERSONNE (ID);
```

```
ALTER TABLE ADD CONSTRAINT EMP_SOC_FK FOREIGN KEY  
(SOCIETE) REFERENCES SOCIETE (ID);
```

SQL: le LDD, modifier une table

Exemple

(créer la table *EMPLOI* du modèle relationnel donné)



```
CREATE TABLE EMPLOI (  
    PERSONNE NUMBER NOT NULL,  
    SOCIETE  NUMBER NOT NULL,  
    SALAIRE  NUMBER(8,2) NOT NULL,  
    DATE_DEBUT DATE NOT NULL,  
    DATE_FIN DATE  
);
```

```
ALTER TABLE EMPLOI ADD CONSTRAINT EMPLOI_SALAIRE_CK  
CHECK(SALAIRE BETWEEN 0 AND 150000.00);
```

SQL: le LDD, modifier une table

Supprimer une contrainte

```
ALTER TABLE nomTable DROP CONSTRAINT nomContrainte  
[CASCADE] ;
```

Par défaut si l'on souhaite supprimer la contrainte ***PRIMARY KEY*** d'une table parent (référéncée par une autre table) l'opération sera rejetée. En ajoutant la clause ***CASCADE*** à l'opération de suppression, le lien entre les deux tables est « rompu ».

SQL: le LDD, supprimer une table

```
DROP TABLE nomTable [CASCADE] ;
```

Permet de supprimer la table avec toutes ces lignes.

Par défaut, si la table à supprimer est référencée par une autre table par une contrainte de clé étrangère, l'opération sera rejetée.

Avec la clause ***CASCADE***, tous les objets dépendants sont alors supprimés le lien entre la table parent et celle qui la référence est « rompu ».

SQL: le LDD, « vider » une table

Pour « vider » une table sans la supprimer

```
TRUNCATE TABLE nomTable;
```

Commande non valide si la table est une table référencée par une contrainte de clé étrangère.

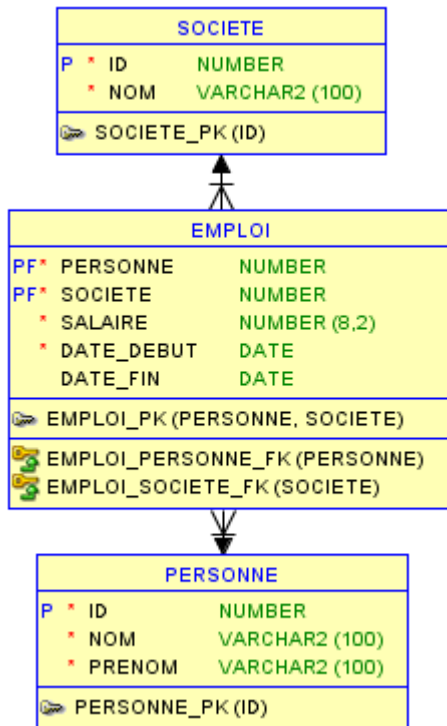
```
DELETE FROM TABLE nomTable [CASCADE];
```

→ Résultat identique. Cependant la close **CASCADE** optionnelle permet de « vider » une table référencée.

SQL: le LDD, auto incrémentation

Exemple

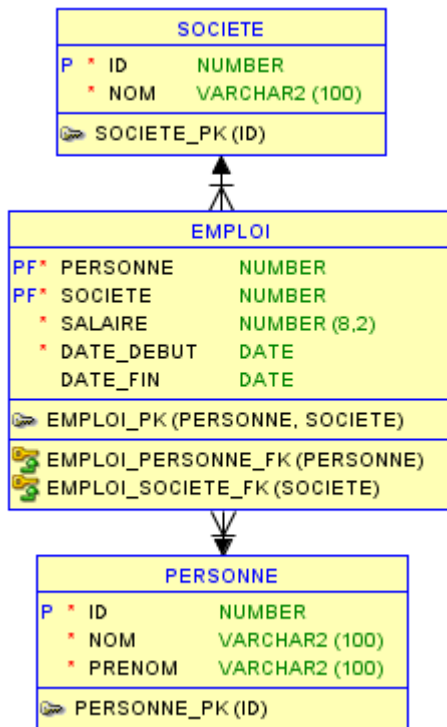
(créer la table *SOCIETE* du modèle relationnel donné)



```
CREATE TABLE SOCIETE (
    ID NUMBER NOT NULL,
    NOM VARCHAR2(100) NOT NULL,

    CONSTRAINT SOCIETE_PK PRIMARY KEY (ID),
);
```

SQL: le LDD, auto incrémentation

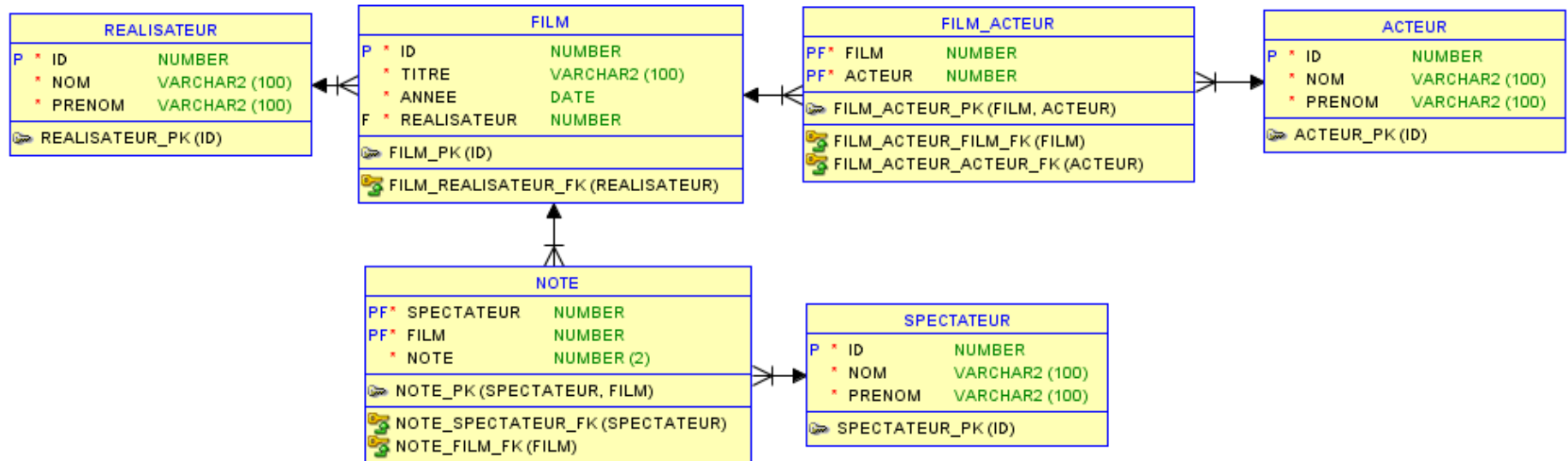


Comment gérer l'auto incrémentation sous Oracle 12c (on verra comment pour les versions précédentes dans un autre cours, car c'est un peu plus tricky).

```
CREATE TABLE SOCIETE (  
    ID NUMBER GENERATED BY DEFAULT ON NULL AS  
        IDENTITY,  
    ...  
);
```


SQL: le LDD, exercice

Donner le LDD du modèle relationnel suivant :



identifier les contraintes qui n'apparaissent pas sur le schéma.

SQL: le LDD, solution

```
CREATE TABLE acteur (  
    id          NUMBER NOT NULL,  
    nom         VARCHAR2(100) NOT NULL,  
    prenom      VARCHAR2(100) NOT NULL,  
  
    CONSTRAINT acteur_pk PRIMARY KEY (id)  
);
```

```
CREATE TABLE realisateur (  
    id          NUMBER NOT NULL,  
    nom         VARCHAR2(100) NOT NULL,  
    prenom      VARCHAR2(100) NOT NULL,  
  
    CONSTRAINT realisateur_pk PRIMARY KEY (id)  
);
```

SQL: le LDD, solution

```
CREATE TABLE film (  
    id                NUMBER NOT NULL,  
    titre             VARCHAR2(100) NOT NULL,  
    annee             DATE NOT NULL,  
    realisateur       NUMBER NOT NULL,  
  
    CONSTRAINT film_pk PRIMARY KEY (id)  
);
```

```
ALTER TABLE film ADD CONSTRAINT film_realisateur_fk  
FOREIGN KEY (realisateur) REFERENCES realisateur (id);
```

SQL: le LDD, solution

```
CREATE TABLE film_acteur (  
    film      NUMBER NOT NULL,  
    acteur    NUMBER NOT NULL,  
  
    CONSTRAINT film_acteur_pk PRIMARY KEY (film,acteur)  
);
```

```
ALTER TABLE film_acteur ADD CONSTRAINT film_acteur_acteur_fk  
FOREIGN KEY (acteur) REFERENCES acteur (id);
```

```
ALTER TABLE film_acteur ADD CONSTRAINT film_acteur_film_fk  
FOREIGN KEY (film) REFERENCES film (id);
```

SQL: le LDD, solution

```
CREATE TABLE spectateur (  
    id          NUMBER NOT NULL,  
    nom         VARCHAR2(100) NOT NULL,  
    prenom      VARCHAR2(100) NOT NULL,  
  
    CONSTRAINT spectateur_pk PRIMARY KEY (id)  
);
```

SQL: le LDD, solution

```
CREATE TABLE note (  
    spectateur    NUMBER NOT NULL,  
    film          NUMBER NOT NULL,  
    note          NUMBER(2) NOT NULL,  
  
    CONSTRAINT note_pk PRIMARY KEY (spectateur, film),  
  
    CONSTRAINT note_ck CHECK (note BETWEEN 0 AND 10)  
);
```

```
ALTER TABLE note ADD CONSTRAINT note_film_fk FOREIGN KEY  
(film) REFERENCES film (id);
```

```
ALTER TABLE note ADD CONSTRAINT note_spectateur_fk  
FOREIGN KEY (spectateur) REFERENCES spectateur (id);
```