Département Microélectronique
& Télécommunications

POLYTECH MARSEILLE
Ecole Universitaire d'Ingénieurs

Réseaux IP & Bases de données

Session 2 : SQL & Programmation Java (JDBC)

Erwan TRANVOUEZ
erwan.tranvouez@univ-amu.fr

1. Modélisation des données :
traduction MCD -> MLD

MCD et MLD

- Le MCD a permis d'identifier les informations importantes et utiles. Il les a organisées.
- Le MLD traduit ces considérations en vue de les intégrer dans le SGBD. Nous considérons dans ce cours un langage relationnel (la plupart des SGBD sont relationnels)
- Il s'agit :
  - D'affiner pour chaque élément d'information son type de données (numérique, chaine, …)
    => équivalent du type de données d'une variable en programmation
  - D'ajouter des informations d'identification des informations et des liens entre elles (clefs primaires, clefs étrangères)
  - Pour optimiser leur stockage et traitement qui sera poursuivi au niveau physique par le SGBD.

Modèle relationnel

- Langage de modélisation « textuel » des informations en vue de leur traduction en tables.
- Principe:
  - Paradigme relationnel: Tout est Relation.
  - Les relations sont caractérisées par des attributs
- Notations
  - Nom_relation ($\underline{\text{attribut}}_1$, attribut_2... $\textit{attribut}_n$)
  - $\underline{\text{Attribut}}_1$ est une clef primaire (contient un identifiant)
  - Attribut_2 est un attribut simple
  - $\textit{attribut}_n$ est un attribut qui doit contenir un valeur correspondant à celle d'une clef primaire d'une autre table.

Traduction d'un MCD en MLD

- Rappel l'objectif est d'arriver à produire quelque chose comme ça :

Entité Livre

- Forme relationnelle: **Livre** (**ISBN**, NomAuteur, PrenomAuteur, Titre)

- Qui permettra de mémoriser des informations comme suivant

Table Livre

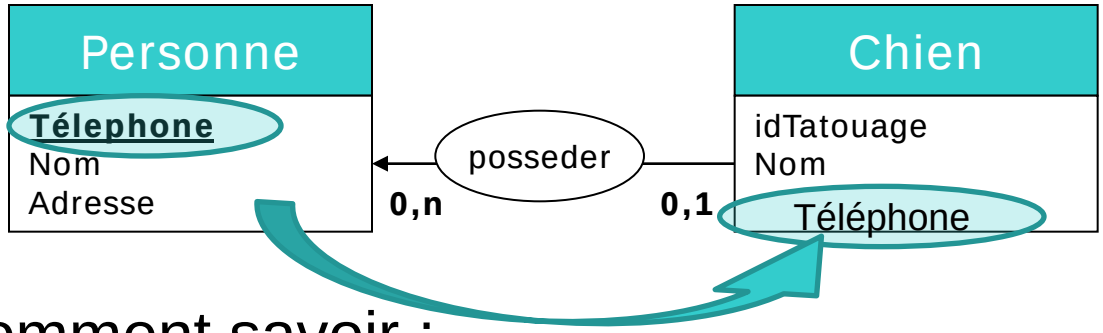
ISBN	*NomAuteur*	*PrénomAuteur*	*Titre*
209178527X	NEY	Henry	Automatique et informatique industrielle
2851102869	FAURE	Jacques	Almanach Vermot 2010
2070628035	ABOUET…	Marguerite	Aya de Yopougon, Tome 5

- Processus traduction relativement simple :
  - Entités => Tables
  - Relations => Importation clefs étrangères & propriétés OU Table
  - Propriétés => Propriétés avec type de données

  => règles de traduction différentes selon les multiplicités

Relation binaire 1 .. N

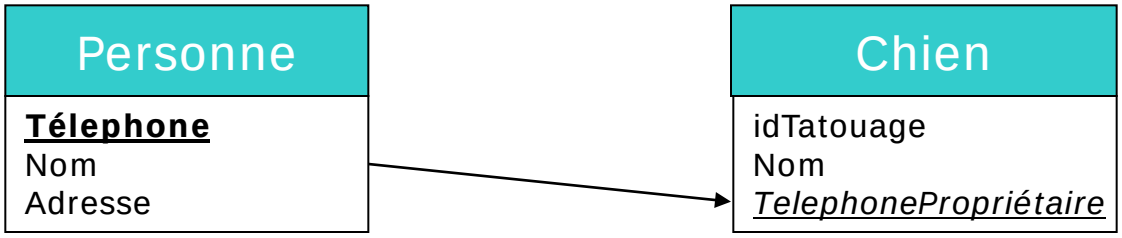
Illustration



MCD

- Comment savoir :
  - Si un chien à un propriétaire ?
  - Qui est son propriétaire ?

- Le numéro de téléphone rend possible l'identification de son propriétaire



MLD

Relation binaire 1 .. N

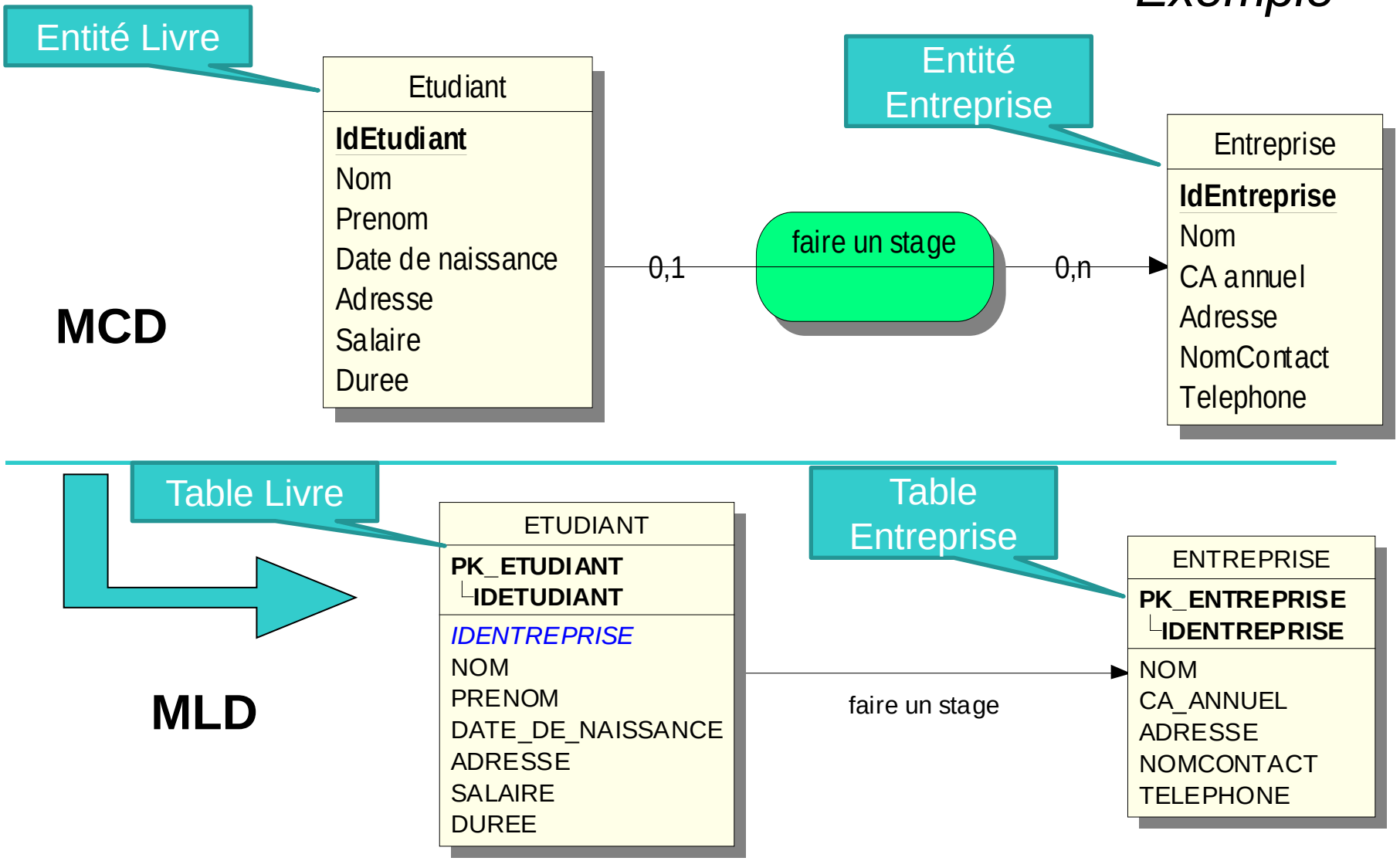
Principe de traduction

- C’est-à-dire une relation (x .. 1) -> (x .. n)
- Ce type de relation traduit une dépendance fonctionnelle.

- Se traduit au passage du MLD
  - par la création d’une **clé étrangère** sur la Table **dépendante**.
  - Si x vaut 0 (donc participation pas obligatoire) cela signifie que l’attribut portant cette clé étrangère peut ne pas être renseigné (***null*** autorisé au niveau implémentation).

Relation binaire 1 .. N

Exemple



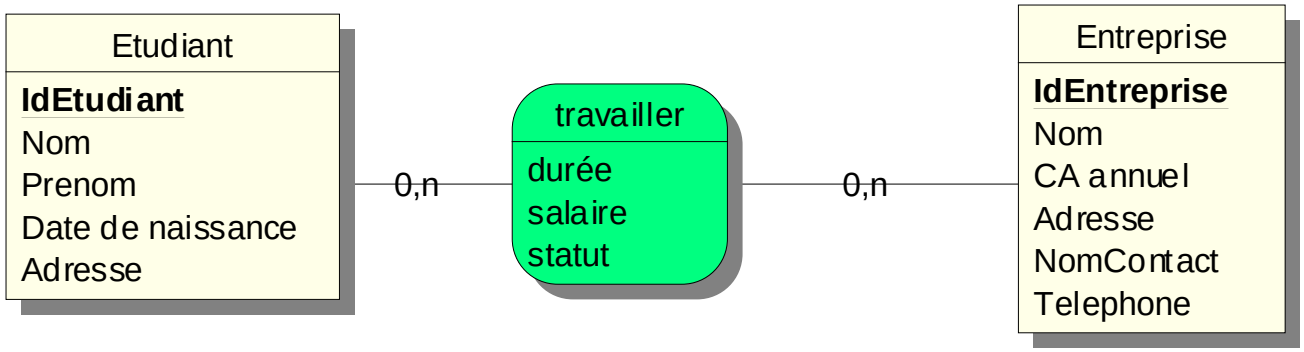
Relation binaire N .. N

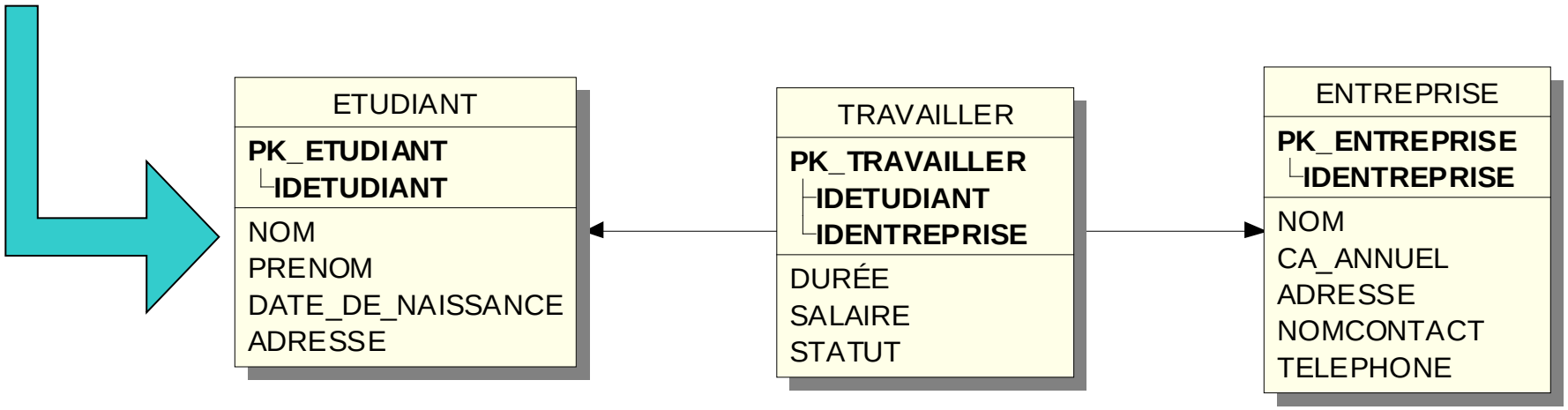
Principe de traduction

- C’est-à-dire une relation (x .. n) -> (x .. n)
- Cas précédent inapplicable
  - car sinon plusieurs valeurs dans la clef etrangères
  - => *Collier un peu gros*
- Solution :
  - Utilise une 3ème table qui associera les occurrences participant à la relation (MCD)

Relation binaire N .. N

Exemple



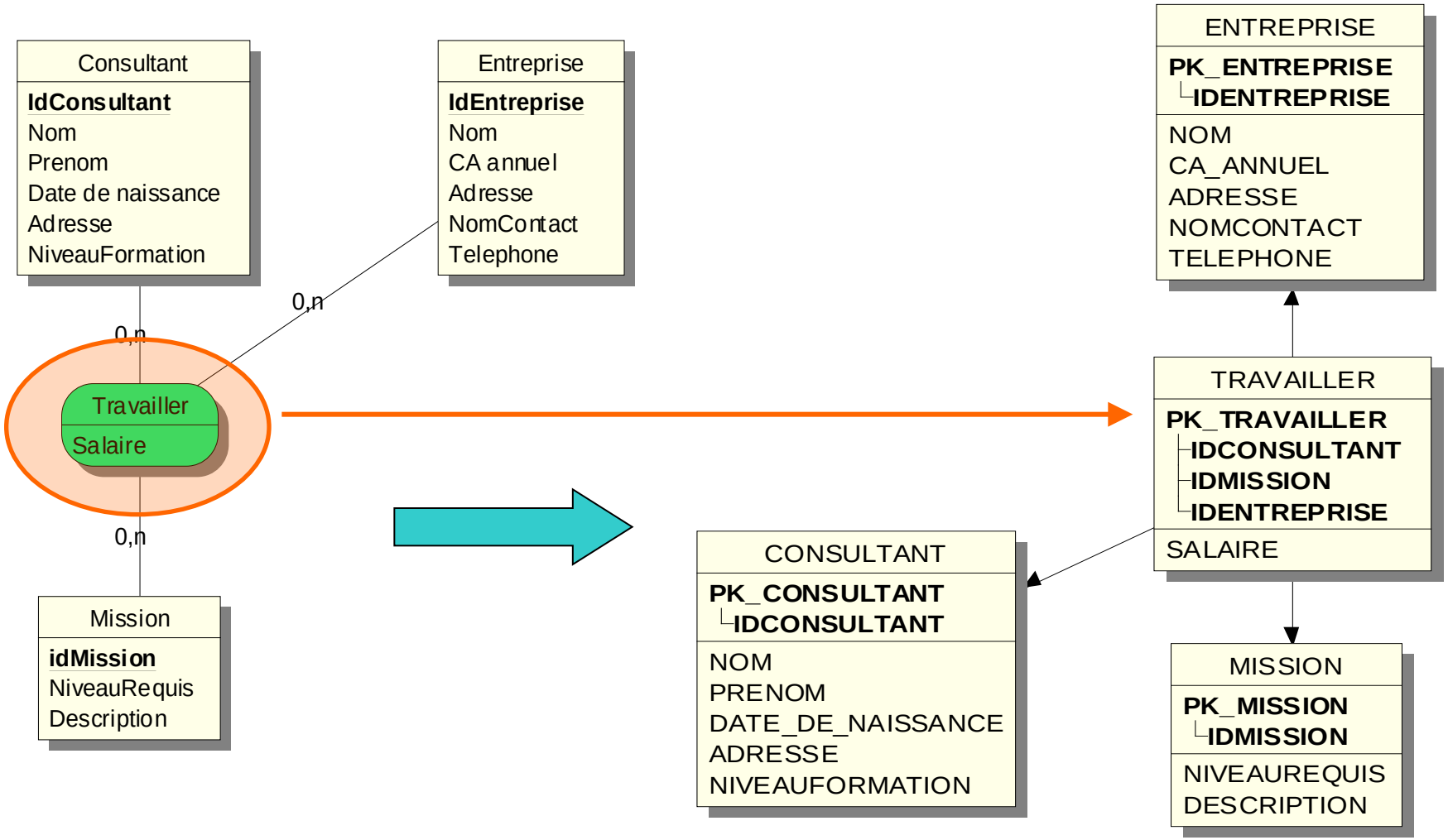


Traduction réalisée avec WinDesign

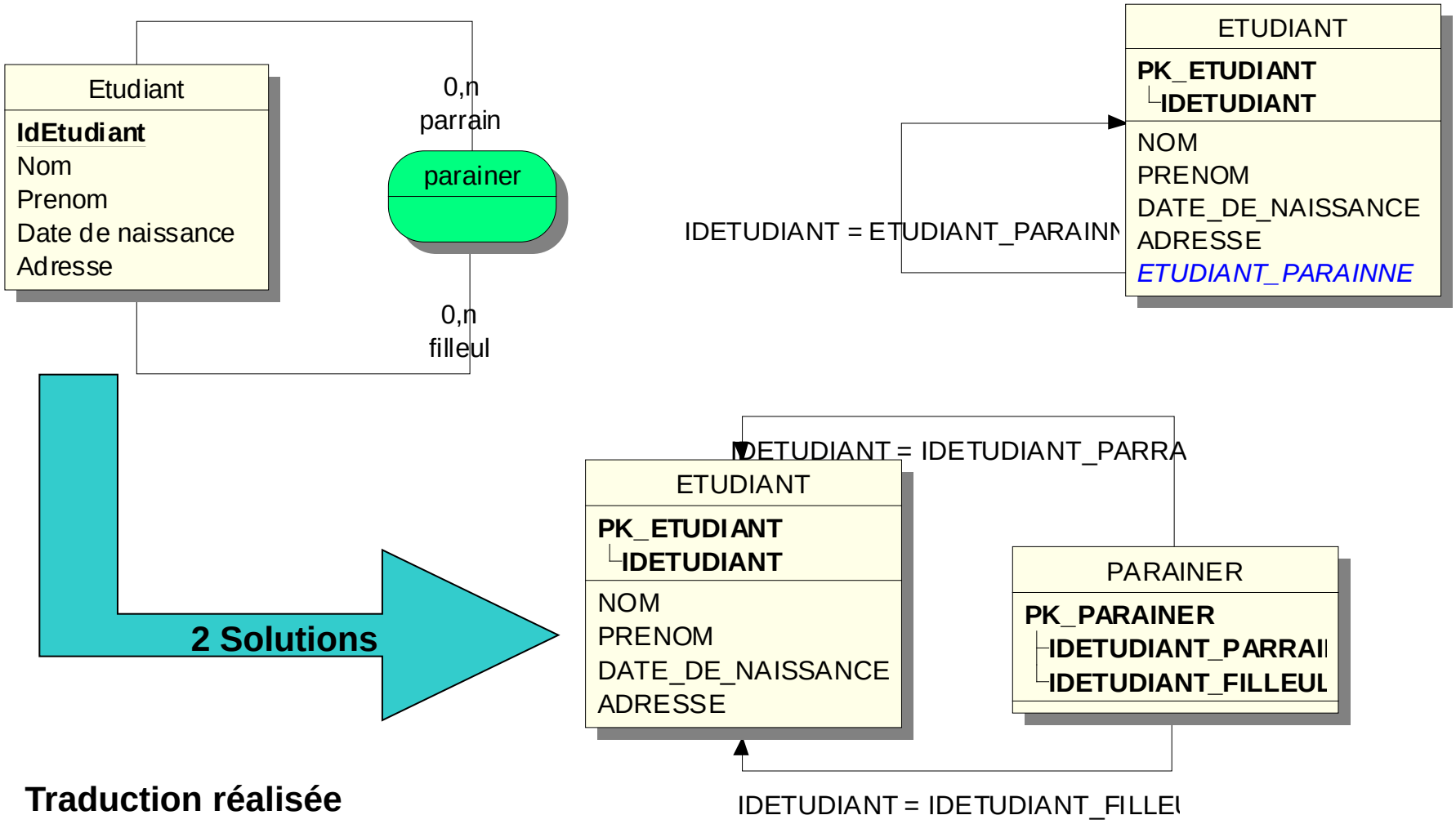
Relation ternaire

Principe de traduction

- Même principe que binaire n..n.



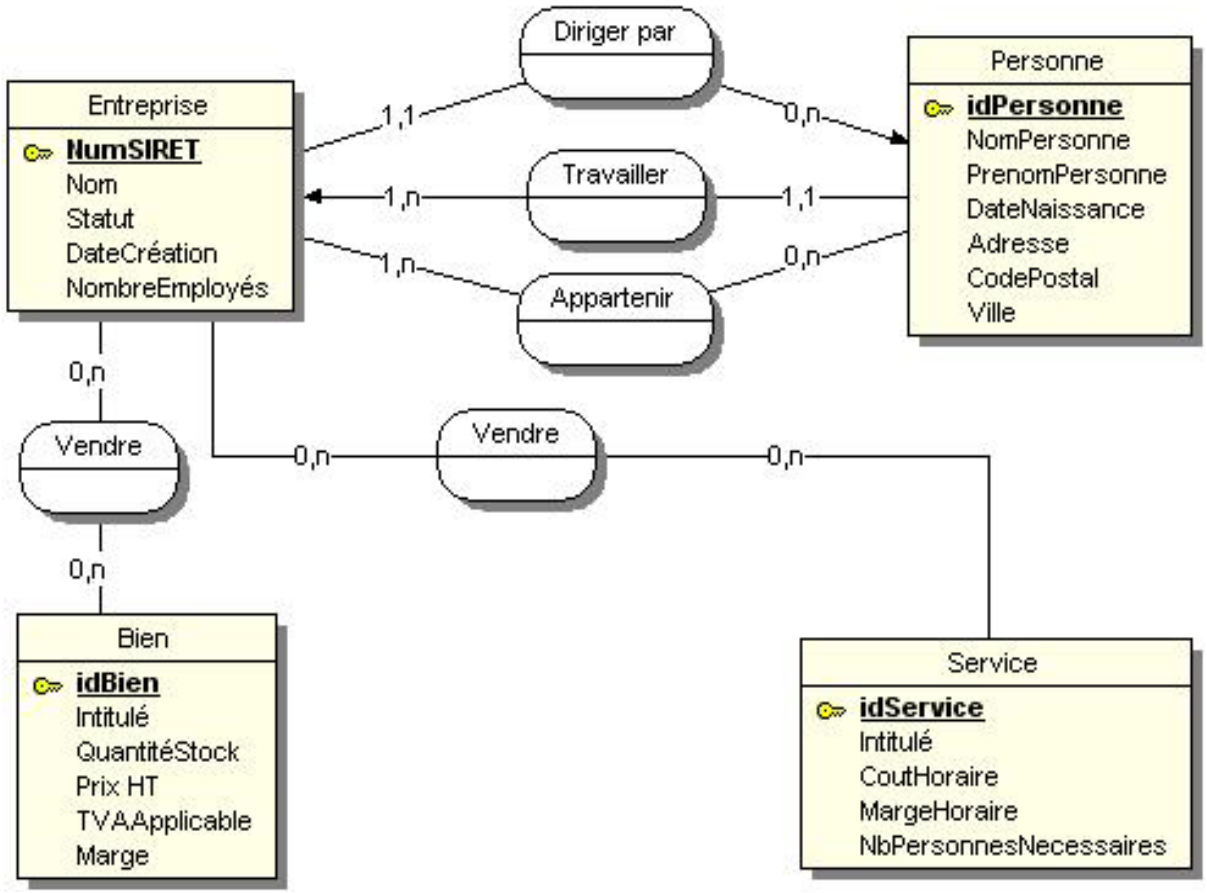
Relation réflexive



Traduction réalisée avec WinDesign

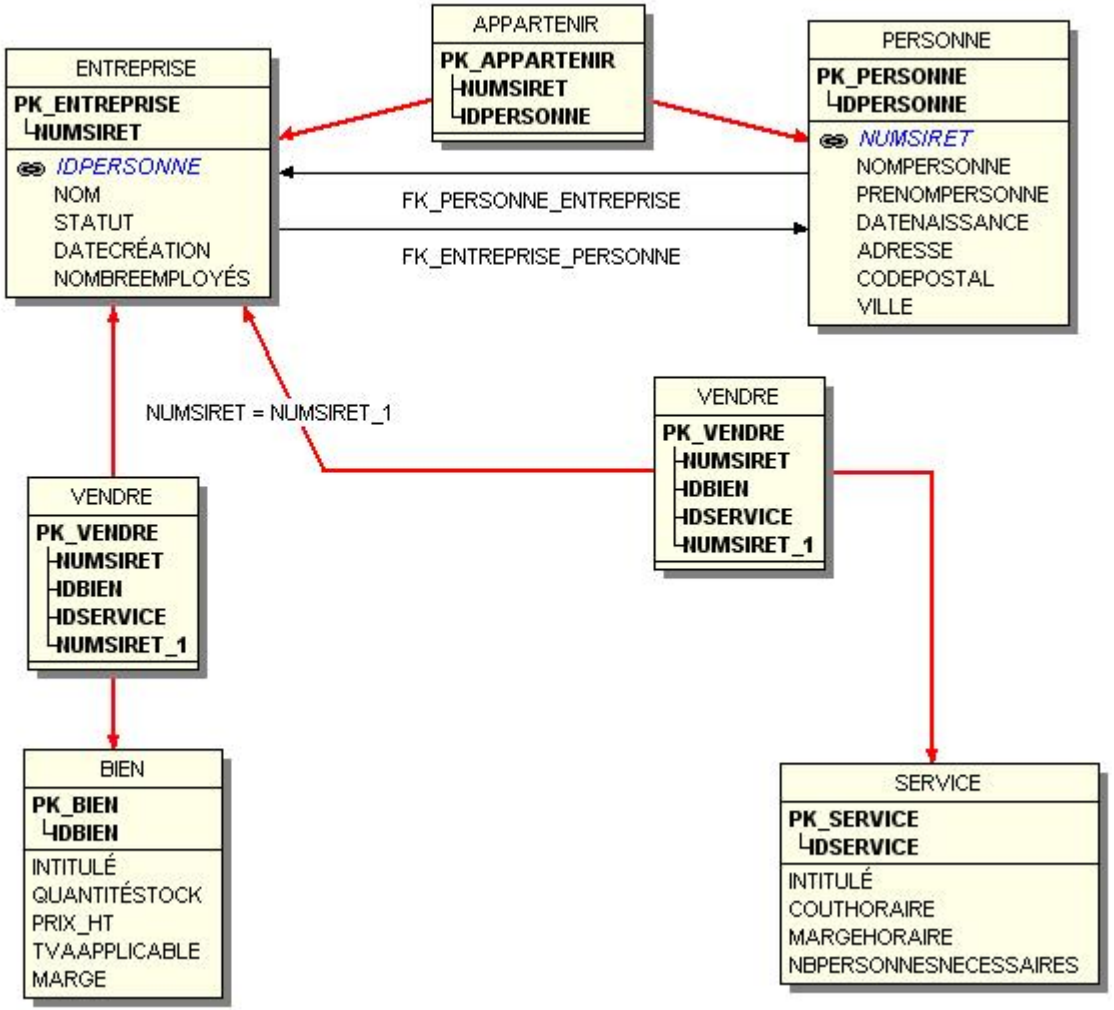
Autre exemple de traduction

MCD -> MLD



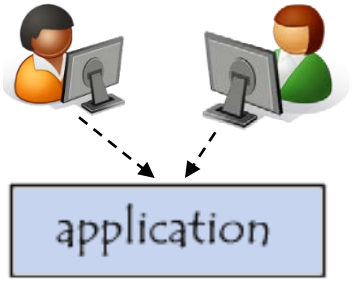
Autre exemple de traduction

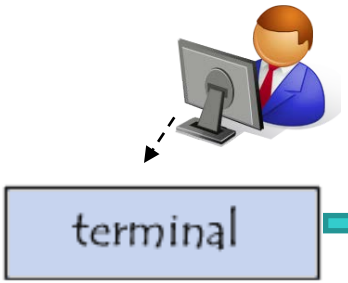
MCD -> MLD

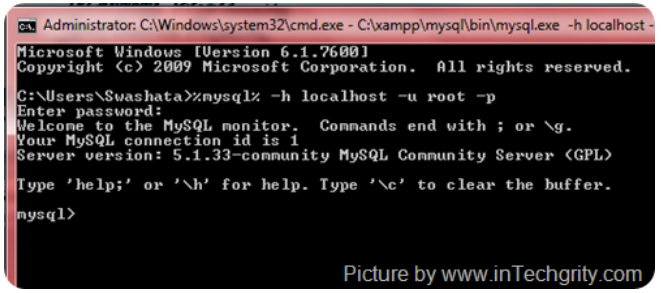


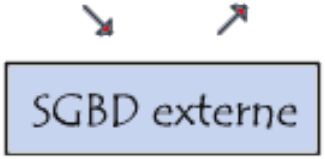
2. SGBD & SQL

Architecture d'un SGBD

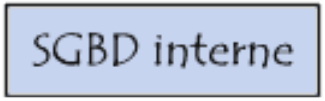






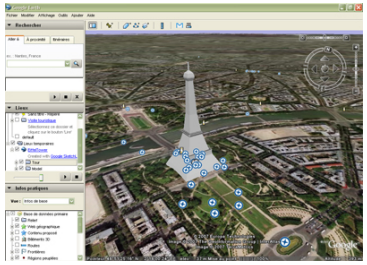


Gere l'interface utilisateur
(SQL) Vues

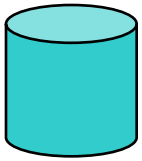


Gere les tables réelles
Optimise les requêtes

Gestion physique des fichiers :
création, suppression, Indexes
et hachages



Google Earth

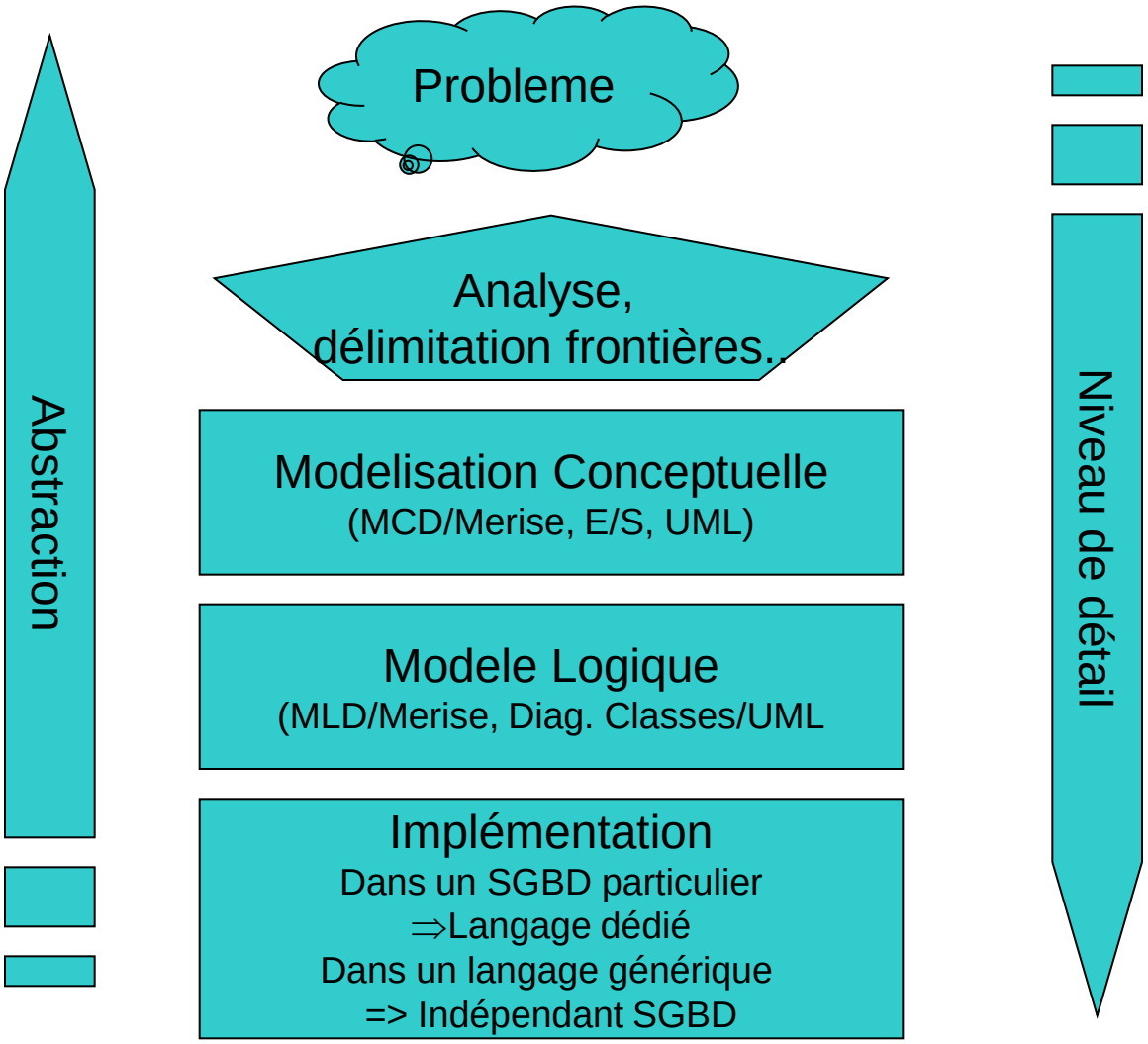


Données (Fichiers)

Disques

Adapté de http://georges.gardarin.free.fr/Livre%20DM_fichiers/1-RappelsBD.ppt

Mise en perspective du processus de développement d'un SI : point de vue des données



Retour sur les fonctions d'un SGBD

- Dans sa mission de mémorisation et d'exploitation des données, le SGBD doit permettre d'/e:
  - *Ajouter*
  - *Modifier (mise à jour)*
  - *Supprimer*
  - *Rechercher*

des données

Langage De Manipulation des Données

- Ce qui suppose d'avoir auparavant :
  - *Défini les données*

Langage De Description des Données

- => *C'est ce que propose SQL de manière générique*

Structured Query Language : SQL

- Standard de langage de manipulation de donnée : tous les SGBD respectent cette norme (ou du moins la majorité des spécifications du langage : par ex. MySQL Lite utilise un sous ensemble de SQL)
- Pourquoi un standard ?

- *A noter : dans le cadre de l'objectif de ce cours, nous n'exploreront pas l'ensemble des possibilités du langage.*

SQL : Illustration

- Retour sur l'exemple du 1ere session

- Cette entité décrite au niveau conceptuelle voire logique doit être traduite au niveau du SGBD pour pouvoir contenir des données comme les suivantes :

ISBN	*NomAuteur*	*PrénomAuteur*	*Titre*
209178527X	NEY	Henry	Automatique et informatique industrielle
2851102869	FAURE	Jacques	Almanach Vermot 2010
2070628035	ABOUET…	Marguerite	Aya de Yopougon, Tome 5

SQL : Illustration

- Code SQL de création de la table

```
CREATE TABLE Livre (
   ISBN              CHAR(10)NOT NULL PRIMARY KEY,
   Nom_Auteur        CHAR(10)NULL,
   Prenom_Auteur     CHAR(10)NULL,
   Titre             CHAR(20)NOT NULL
);
```

- Insertion de données :

```
INSERT INTO     Livre
VALUES
   (‘209178527X’, ‘NEY’,NULL,’Automatique … ’)
);
```

SQL : Illustration

- "Affichage" des données contenu dans une table :

```
SELECT ISBN, Nom_Auteur, Prenom_Auteur, Titre
FROM Livre
;
```

- Modification de données :

```
UPDATE Livre (
   SET PrenomAuteur='Henry'
WHERE
   Nom_Auteur='NAY'
);
```

BD & SQL : vocabulaire

- *Colonne*: correspond à l'attribut déclaré dans le MCD
- *Enregistrement* ou *ligne* : une occurrence d'une entité enregistrable dans la table correspondante dans la base de donnée.
- Requête : demande d'interrogation de la base de donnée faite auprès du SGBD par l'intermédiaire d'un SELECT.

3. Syntaxe SQL
Les bases

Types de donnée SQL

Type de donnée	Syntaxe	Definition
Integer	Integer	entier
Small Int	Smallint	
Numeric	numeric(p,s)	***p*** : precision globale ie nb chiffres, ***s*** précision après la virgule. Par ex. numeric(4,2) peut contenir 1234 123,4 12,34 1,234 -> converti en 1,23 12,349 -> converti en 12,35.
Decimal	decimal(p,s)	idem
Real	real	Nombre décimal à simple précision
double precision	double precision	Nombre décimal de double précision
Real (Float)	float(p)	***p*** précision
Chaîne de caractères	char(n)	***n*** nombre de caractères à stocker. S'il y a moins de n caractère, la chaîne est remplie d'espaces.
Chaine de caractères variables	varchar2(x)	Where ***x*** is the number of characters to store. This data type does NOT space pad.
bit	bit(n)	***n*** nb de bits à stocker.
bit variable	bit varying(n)	***n*** nombre de bits à stocker. La longueur peut aller jusqu'.

Types de donnée SQL

Type de donnée	Syntaxe	Definition
date	date	Stocke des dates (année, mois et jour)
time	time	Stores the hour, minute, and second values.
Timestamp	timestamp	Stocke année, mois, jour, heure, minute et secondes
time with time zone	time with time zone	Pareil que time avec en plus l'information de décalage par rapport UTC de l'heure stockee.
timestamp with time zone	timestamp with time zone	Idem que timestamp …
	…	…

Remarque: bien que censé être générique, chaque SGBD tend a vouloir surdéfinir ses propres données adaptés à des usages spécifiques ou répondant à des objectifs d'optimisation.

Création d'une base de données

- C'est là que seront créées, gérées les tables et les données qu'elles contiennent.
- Syntaxe SQL :

```
CREATE DATABASE Nom_Base_de_donnee ;
```

- Cette instruction « ordonne » le SGBD de créer cette table.
- Charge à lui de se « débrouiller » pour gérer ça au niveau du disque dur/système de fichiers.

CREATE TABLE : Création d'une table

- Syntaxe générale

```
CREATE TABLE nom_table [Base de donnée . ] (
                nom_colonne1 type [contraintes] ,
                nom_colonne2 type [contraintes] ,
                 …,
   )
```

- Contraintes usuelles :
  - Clef primaire :
    - PRIMARY KEY [AutoIncrement]
    - Autoincrement laisse le SGBD gérer les clefs numériques (incrémente à chaque ajout d'enregistrement/ligne => ne pas remplir soit même la colonne)
  - Clef étrangère :
    - REFERENCES table_etrangere (nom_colonne)
  - « remplissage » de la colonne obligatoire (pas de valeur vide) :
    - NOT NULL
  - Contraintes sur les valeurs autorisées :
    - CHECK (condition sur la valeur) (cf. section 3.)

Création d'une table (suite)

- Contraintes usuelles :
  - Unique (pas de doublon) :
    - UNIQUE
    - => ex. dans le cas d'une clef étrangère indique la relation 0.1 (la clef ne peut apparaître/etre utilisée qu'1 fois. Si en plus on ajoute NOT NULL correspond à 1.1)

- Il est possible de rajouter la contrainte en fin de déclaration. Cela est indispensable en cas de clefs composées.

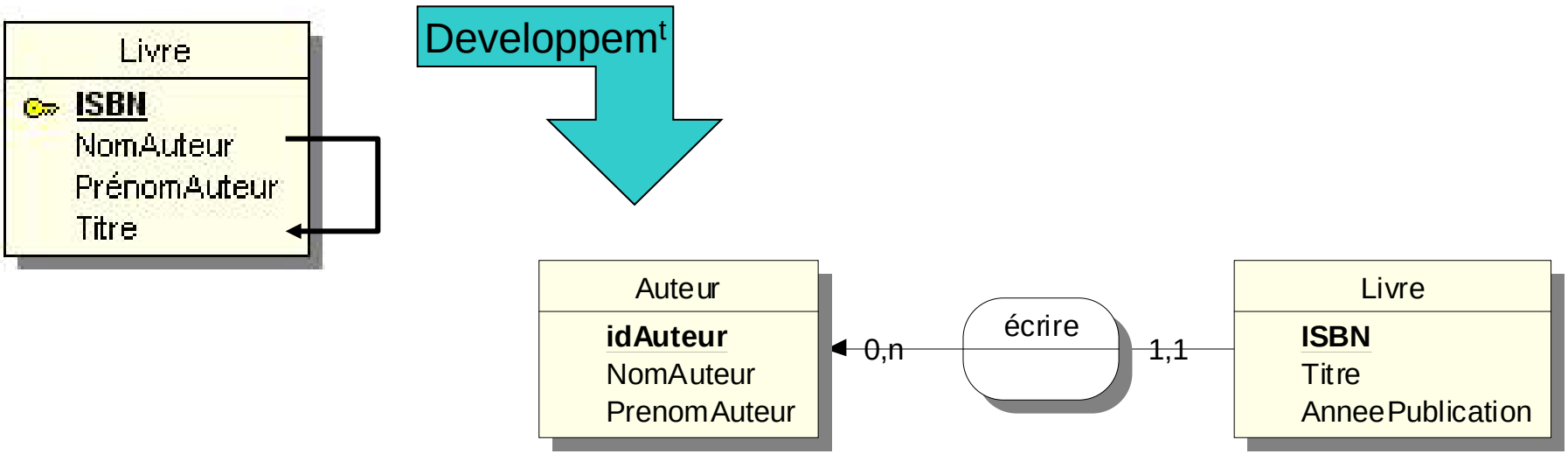
```
CREATE TABLE nom_table [Base de donnée . ] (
  col1 type [contraintes] ,
  col2 type [contraintes] ,
   …,
  PRIMARY KEY (col1, col2)
);
```

Reprendre l'ex. chien proprietaire avec entité plus politiquement correct avec traduction des contraintes entre entité au niveau des contraintes : 01-> Unique e

Création d'une table : exemples

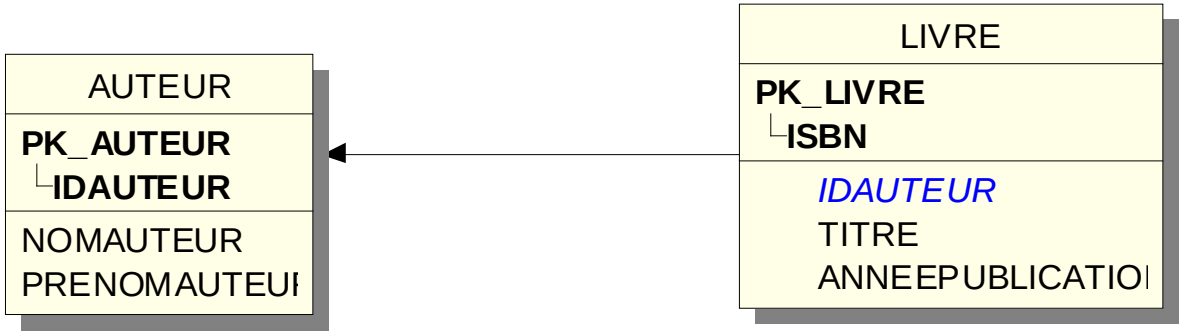
- Exemple initial

```
CREATE TABLE Auteur (
   PK_ISBN      CHAR(10)   NOT NULL PRIMARY KEY,
   NomAuteur    CHAR(10),
   PrenomAuteur CHAR(10),
   Titre        CHAR(20)   NOT NULL
);
```



Création d'une table : exemples

- Exemple revu & corrigé :



- Code SQL correspondant :

```
CREATE TABLE Auteur (
    PK_IdAuteur INTEGER      PRIMARY KEY,
    Nom_Auteur     CHAR(10),
    Prenom_Auteur  CHAR(10)
    );
```

```
CREATE TABLE Livre (
    PK_ISBN        CHAR(10)  PRIMARY KEY,
    Titre          CHAR(10)  NOT NULL,
    AnneePubli     DATE,
    FK_IdAuteur    INTEGER
            REFERENCES Auteur (PK_IdAuteur)
    );
```

Rque: par simplicité le type chaine de caractères fixe a été utilisé ici alors qu'une chaîne variable aurait plus de sens pour stocker des noms (optimisation de l'espace).

Insérer des données dans une table

- Syntaxe SQL (version simple) :

```
INSERT INTO Nom_Table
   (‘colonne1’, ‘colonne2’, …,’colonnen’)
VALUES
   (‘val1’, ‘val2’,…, ‘valn’)
;
```

- La ligne colonne est optionnelle si pour chaque enregistrement la donnée de chaque colonne est fournie dans l’ordre de leur déclaration.
- Si l’enregistrement ne respecte pas les contraintes, il n’est pas ajouté dans la base de données.

Insertion de données : exemple

- Syntaxe SQL (version simple) :

```
INSERT INTO       Livre
(‘ISBN’, ‘Nom_Auteur’,’Prenom_Auteur’,’titre’)
VALUES
   (‘209178527X’, ‘NEY’,NULL,’Automatique … ’)
;
```

- Syntaxe SQL (version avec clef etrangere) :

```
INSERT INTO       Auteur
   (‘PK_IdAuteur ’, ‘Nom_Auteur’,’Prenom_Auteur’)
VALUES
   (‘1’, ‘FAURE’, ‘Jacques’)
;

INSERT INTO       Livre
   (‘PK_ISBN’,’titre’, ‘FK_IdAuteur’)
VALUES
   (‘209178527X’, ‘Almanach Vermot 2010’ , ‘1’)
;
```

Mettre à jour les données dans une table

- Modification de valeurs

```
UPDATE Nom_Table
SET
   nom_col1 = ‘val’ ,
   nom_col2 = ‘val’
WHERE
   <condition>
;
```

- Suppression d’enregistrements

```
DELETE FROM Nom_Table
WHERE
   <condition>
;
```

- Conditions : cf. SELECT

Afficher les données d'une table

- SELECT

```
SELECT
   (Nom_Col1 [As nom1], …, Nom_Coln [As nomn] )
FROM
   TABLE1[, TABLE2…]
WHERE
   condition
[ORDER BY COL1 [ASC|DESC] ];
```

- Condition est une expression booléenne :
  - Comparant 2 valeurs (‘=‘, ‘<>’, ‘<‘, ‘>’,’>=‘, ‘<=‘):
    - Ex. prix < 10,50 ou prix est la valeur d'une colonne pour un enregistrement
  - Combinant plusieurs expressions avec des opérateurs conjonctifs (AND, OR, NOT)
    - Prix > 10 AND Prix < 20

Afficher les données d'une table

- SELECT

- Condition est une expression booléenne :
  - Utilisant des opérateurs logiques ('IS NULL', 'BETWEEN', 'IN', 'LIKE'…)
    - Ex. NOM LIKE H_G% : nom commencant par H suivit de n'importe quel caractere puis d'un G… la fin n'etant pas importante => HUGO, HUGOT, HEGEL valide cette expression.
- ORDER BY permet de trier les résultats selon une colonne dans l'ordre croissant (ASC) ou décroissant (DESC)

SELECT Exemple

- Afficher toutes les informations sur toutes les colonnes

```
SELECT * FROM Livre;
```

- Afficher les noms des auteurs :

```
SELECT Nom_Auteur FROM Auteur;
```

- Afficher les livres publiés apres une date

```
SELECT Titre, annee_publi
FROM Livre
WHERE annee_publi > ‘2000:11:15’;
```

- Afficher les livres et le nom de leurs auteurs

```
SELECT Nom_Auteur, Prenom_Auteur, Titre
FROM Auteur, Livre
WHERE Pk_IdAuteur=FK_idAuteur;;
```

3. Syntaxe SQL

Un peu plus loin dans SQL

Jointures … théorie…

- Lié à l'approche relationnelle dans la conception des bases de données.
- Mise en relations de 2 (ou plusieurs) tables puis affichage du résultat :
  - Produit toute les combinaison possible (a, b) avec a provenant de la 1ere table et b de la 2eme…
  - => peu utile sans critère de jointure

```
Equivalent à
SELECT *
FROM Auteur JOIN Livre ;
```

```
sqlite:> SELECT * FROM Auteur, Livre;

PK_IdAuteur  Nom_Auteur  Prenom_Auteur  PK_ISBN     Titre                AnneePubli  FK_IdAuteur
-----------  ----------  -------------  ----------  -------------------  ----------  -----------
1            Faure       Jacques        209178527X  Almanach Vermot 2010             1
1            Faure       Jacques        2070628035  Aya de Yopougon      3           3
1            Faure       Jacques        209178527X  Automatique et infor 2           2
1            Faure       Jacques        XXXXSQSS    Aya de youpougon 2               3
2            NEY         Henry          209178527X  Almanach Vermot 2010             1
2            NEY         Henry          2070628035  Aya de Yopougon      3           3
2            NEY         Henry          209178527X  Automatique et infor 2           2
2            NEY         Henry          XXXXSQSS    Aya de youpougon 2               3
3            Abouet      Marguerite     209178527X  Almanach Vermot 2010             1
3            Abouet      Marguerite     2070628035  Aya de Yopougon      3           3
3            Abouet      Marguerite     209178527X  Automatique et infor 2           2
3            Abouet      Marguerite     XXXXSQSS    Aya de youpougon 2               3
```

Jointures Interne

- Permet de joindre des données de tables différentes :
  - Exemple : afficher les auteurs et les livres qu'ils ont écrits :
- => indiquer
  - les tables à joindre
  - Le critère de jointure : ex. l'id de l'auteur est le même que la valeur de la clef étrangère dans l'autre table

```
SELECT PK_ISBN as ISBN, Titre, Nom_Auteur as Nom,
                                Prenom_Auteur as "Prénom"
FROM Livre, Auteur
WHERE
   Auteur.Pk_IDAuteur = Livre.FK_IdAuteur
```

  - Appelée aussi jointure interne : la syntaxe normée est :
    SELECT … FROM Table1 INNER JOIN Table2
    ON Table1.col = Table2.col2

Jointures Externe

- Joint des tables dans lesquelles on n'a pas pour chaque enregistrement de lien avec la 2eme (cas des valeurs nulles)
  - Exemple : afficher les auteurs et les livres y compris quand les auteurs n'ont pas encore publié de livre (donc leur id n'apparaît nulle part dans la table)
- Syntaxe :
  - les tables à joindre
  - Le critère de jointure : ex. l'id de l'auteur est le même que la valeur de la clef étrangère dans l'autre table

```
SELECT *
FROM Auteur LEFT OUTER JOIN Livre
     ON PK_IdAuteur = FK_IdAuteur;
```

  - Cad : on fait la jointure sur la clef, on affiche les valeurs qui vont bien et on ajoute les valeurs de la table de gauche (LEFT)
  - RIGHT fait la meme chose avec la table de droite et FULL le fait sur les 2 tables
  - => On voit apparaître des enregistrement avec par endroit aucune valeur en face de la colonne.

Opérateurs d'agrégation

- Permet d'effectuer des calculs sur les tables. Dans le cadre d'un SELECT, on peut calculer
  - une moyenne AVG(x)
  - un nombre de valeur COUNT(x)
  - Max/min Max(x),
  - une somme : SUM(x).
  - Où x est le nom de la colonne sur laquelle porte l'opération
- Nécessite de grouper les enregistrements par valeur identique puisque c'est sur ces groupes de valeurs que se fera le calcul
- => indiquer
  - La colonne sur laquelle doit se faire le calcul
  - Répéter le nom de la colonne afin que l'on sache à quelle colonne correspond le calcul
- Pour restreindre le calcul on utilise la clause HAVING (restreindra l'affichage des résultats des calcul) de la même manière que WHERE (ne concerne que les données à "afficher" sur lesquelles se feront les calculs). Ex. n'afficher que les auteurs d'un seul livre.

Opérateurs d'agrégation

- Exemple
  - Calculer le nombre de livres dans la table Livre

```
SELECT Count(id) FROM Livre ;
```

  - Nombre de livre par auteur :

```
SELECT Fk_IdAuteur , COUNT(PK_ISBN)
FROM Livre GROUP BY Fk_IdAuteur;
```

  - Mais si ont rendre le résultat un peu plus clair => il faut ajouter le nom de l'auteur => faire une jointure

```
SELECT Nom_Auteur, COUNT(PK_ISBN)
FROM  Livre, Auteur
WHERE Pk_IdAuteur=Fk_IdAuteur
GROUP BY Fk_IdAuteur
;
```

UPDATE TABLE

- Modifier la valeur d'une ou plusieurs colonnes:

```
UPDATE Nom_Table1 SET
   col1 = val1
   [, col1=val2]
WHERE
   condition ;
```

- Exemple

```
UPDATE Auteur SET
   Nom_Auteur = 'Abouet',
   Prenom_Auteur = 'Marguerite'
WHERE
   Nom_auteur LIKE 'Abouet%'
AND Prenom_Auteur LIKE 'Marguerite%';
```

Requêtes imbriquées …

- Un sélect permet de filtrer l'affichage des données d'une table… mais son résultat peut intervenir dans une autre notamment.
- Peut être utilisée pour trouver une valeur qui servira de contrainte pour l'autre requête.
  - Pour simplifier une requête : par ex. au lieu d'avoir une condition longue, on peut la découper en 2 requêtes imbriquées => plus facile a tester (evite le big bang).
  - Pour effectuer des calculs qui serviront dans la requête : moyenne, constitution de liste…

4. SQLite

SQLite

- SGBD classique : un SGBD classique s'exécute généralement sur un serveur dédié sur lequel se connecteront des clients afin de lui transmettre des requêtes
- SQLite (http://www.sqlite.org) est un SGBD embarqué : ie à la fois client et serveur.
- Il peut être enveloppé dans une application (ex. firefox, thunderbird, antivirus McAfee, Skype…)
- C'est un SGBD SQL qui gère plusieurs tables, les vues, les transactions… La base de données est stockée dans un seul fichier.

Démarrage rapide

- Créer une base de donnée qui contiendra nos tables :
  1. Ouvrir une console en ligne de commande
  2. Se positionner dans le répertoire de SQLite
  3. Lancer le programme en indiquant le nom de la base de données

```
C:\ici\etla\sqlite3.exe test.db
sqlite>.mode column
sqlite>.headers on
```

- Exécuter des requêtes SQL
- Quelques commandes:

`.help`	*affiche les commandes usuelles*
`.exit`	*quitte*
`.output stdout`	*résultat affiché dans la console*
`.output FILE`	*résultat renvoyé dans un fichier*
`.tables ?Pattern?`	*Liste les tables correspondant au pattern*
`.read FILENAME`	*exécute les commandes SQL contenues dans FILENAME*
`.schema Table`	*affiche la structure de la table Table*

Aller plus loin

- http://sqlpro.developpez.com/cours/sqlaz/

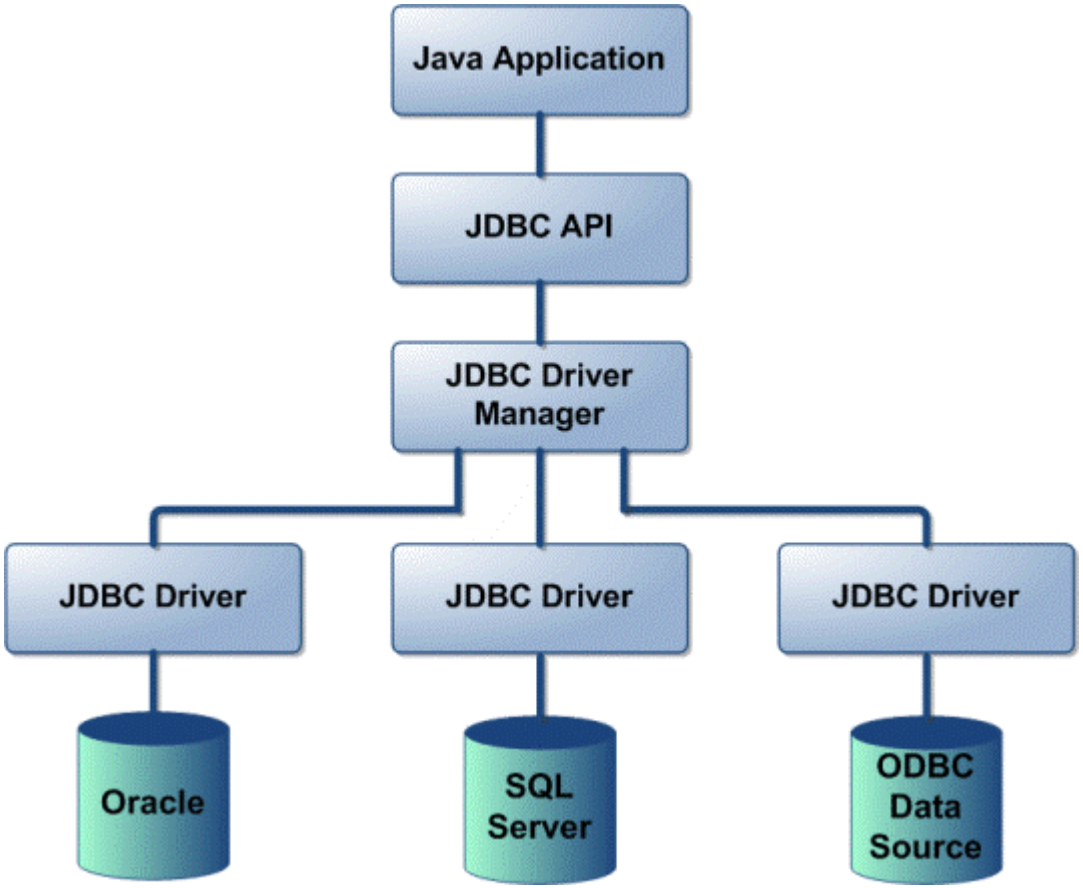
5. JDBC

JDBC : présentation générale

- JDBC = Java DataBase Connectivity
- Cette API permet d'accéder à des bases de données relationnelles depuis un programme Java. L'intérêt est d'avoir du code indépendant du SGBDR sur lequel on se connecte.
- Pour cela, JDBC fournit un ensemble de classes Java liées - via un **Driver -** à un SGBD spécifique.
- Un pilote JDBC est un composant logiciel (middleware) transposant les appels JDBC vers l'API spécifique d'un vendeur de SGBDR.
- **=> Changer de SGBD n'a pas (peu) d'impact sur l'application***

- *sauf cas particuliers ☺

Architecture logicielle d'une application basée sur JDBC



Connexion à un SGBD

- Charge un pilote JDBC
  - On n'instancie pas directement une classe on demande à une autre classe de le faire pour nous…
  - … il y a du travail de back office à faire avant de nous donner l'instance

```
try {
        Class.forName("sun.jdbc.Driver");
        //…
        } catch (ClassNotFoundException ex) {
        /* Pilote non trouvé… /   }
```

À pour effet de charger la classe Driver en cours d'exécution (RunTime), sans avoir à recompiler les sources.

  - Ceci implique que la JVM saura trouver le bon driver en fonction des informations transmises : sun.jbc.Driver
    - Implique qu'il faut chercher une classe Driver
    - dans un sous répertoire sun/jdbc/.
    - visible de la JVM : le répertoire, ou le fichier .jar (fichier zippé contenant des classes java) sont donc dans le CLASSPATH.
    - => Dans Blue J, il faut
      - aller dans le menu : Tools > Preferences > onglet Libraries > et cliquer sur Add (et sélectionner le point .jar par ex.)
      - Réinitialiser la JVM (pour que la librairie soit incluse lors de la compilation) : clic bouton droit sur le « candy bar »  puis reset JVM.

Connexion à un SGBD … suite

- Et on demande une connexion au driver Manager

```
try {
        Class.forName(« sun.jdbc.Driver »);
        Connection cnx =
        DriverManager.getConnection(url,user,pwd);
        //…
        } catch (ClassNotFoundException ex) {
        /* Pilote non trouvé…  */  }
```

- Format de l'url : "<protocol>:<subprotocol>:<subname>"
  - <protocole>: jdbc (of course)
  - <subprotocol> : nom du driver ou au mécanisme de connexion à la base de données (cf. ci après). Cette information permettra au « DriverManager » de savoir quelle classe il faut chercher & charger.
  - <subname> : identifier la source de données (ie BD), syntaxe dépend du sous-protocole et du driver (par ex. connexion réseau ou accès fichier).

Connexion à un SGBD … suite

- Format de l'url : « jdbc:<subprotocol>:<subname>"
- Par exemple :
  (merci http://java.developpez.com/faq/jdbc/?page=connection#urlJDBC)
  - jdbc:odbc:maBase;CacheSize=30;ExtensionCase=LOWER
    Î--- connexion via un driver ODBC maBase sera la source de donnée enregistré dans ODBC (cf. demo au tableau)
  - jdbc:mysql://<host>:<port>/<database_name>?property1=value1 &property2=value2
    - jdbc:mysql://localhost/maBase
      => l'appli java sera sur le serveur ou « tourne » MySQL
    - jdbc:mysql://monSGBD.serveur.com:3306/test
      => connexion « à distance » sur la base test
  - jdbc:sqlite:chemin/ma-bd.db
    - Remarque si on utilise : jdbc:sqlite:ma-bd.db, le chemin considéré par défaut sera l'endroit ou se trouve la classe contenant ce code. Un fichier ma-bd.bd sera créé mais vide (ce qui est ballot si le fichier existe ailleurs).

Communication au travers d'une connexion

- La connexion permet de communiquer avec la base de données :
  - Accès aux instructions
    - createStatement()
    - prepareStatement()
    - prepareCall()
  - Gestion de transactions
    - setAutoCommit()
    - Commit()
    - Rollback()
    - setTransactionIsolation()
  - Informations sur la base
    - getMetaData()

Connexion à un SGBD … suite

- Une façon (simple) de faire est de créer un Statement et lui transmettre la requête SQL.

```
try {
        Class.forName(« sun.jdbc.Driver »);
        Connection cnx =
        DriverManager.getConnection(url,user,pwd);
        Statement stmt = cnx.createStatement();
        stmt.executeUpdate("INSERT INTO TABA VALUES('A','2')”);
    //…
    } catch (ClassNotFoundException ex) {
        // Pilote non trouvé…
    } catch (SQLException ex) {
        // Autres Erreurs SQL…
}
```

Illustration avec SQLite

- D'abord on créé un objet « container » qui représentera un enregistrement.

```
public class Marque {
    private int id;
    private String marque;

    public Marque(int id, String marque) {
        this.id = id;    this.marque = marque;
    }
         public String toString() {
        return "<" + id + "\t" + marque + ">" ;  }
}
```

- Exemple d'usage ?
  1. Les objets seront utilisés pour remplir un tableau modifiable en Java
  2. L'utilisateur modifie des valeurs
  3. La modification est répercutée sur le tableau d'objet
  4. On peut ensuite faire mettre à jour la BD via une boucle avec une requête SQL UPDATE.

En complément des transparents qui suivent :

http://www.tutorialspoint.com/sqlite/sqlite_java.htm
http://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.htm

Illustration avec SQLite

- On se connecte, on liste le contenu de la table « Marque ».

```
import java.sql.*;

public class TestDB
{
        public static void main( String args[] ) {
        Marque[] tab = new Marque[20]; int pos = 0; int id; String marque;
        try {
        Class.forName("org.sqlite.JDBC");
        Connection c = DriverManager.getConnection("jdbc:sqlite:../sqlite/phone.db"
        Statement st = c.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM Marque");
        while(rs.next() ) {
    id = rs.getInt("PK_idMarque"); // ou rs.getInt(1)
    marque = rs.getString("marque"); // ou rs. getString(2)
    tab[pos]=new Marque(id, marque);              pos++;
   }
   System.out.println(" j'ai récupéré "+pos+" enregistrements" );
   for (int i=0; i<pos; i++)       System.out.println(tab[i]);
   } catch ( Exception e ) {
      System.err.println( e.getClass().getName() + ": " + e.getMessage() );
      System.exit(0);             }
    System.out.println("Opened database successfully");
  }  }
```

Illustration avec SQLite : zoom

- Notez l'usage du chemin relatif calculé par rapport à la localisation de la classe java :
  - Classes java : /quelque_part/java/TestDB.class
  - BD : /quelque_part/sqlite/phone.db

```
Connection c = DriverManager.getConnection("jdbc:sqlite:../sqlite/phone.db");
```

- La requête SQL pour lister le contenu de la table Marque est : « `SELECT * FROM Marque;` ». On l'injecte via un objet `statement`.

```
Statement st = c.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM Marque");
```

- La classe `Resultset`, contrairement à ce que son nom pourrait faire croire, ne contient pas l'ensemble des résultats : imaginez les conséquences, en terme de mémoire, si la table contenait des centaines d'enregistrements !
- Elle permet par contre de récupérer 1 à 1 les enregistrements correspondant à la requête avec la méthode `next()` qui renvoie false quand on a parcouru tout l'ensemble des résultats.

Illustration avec SQLite : zoom

```
while(rs.next() ) { /* on traite chaque enregistrement */          }
```

- On doit donc traiter chaque enregistrement : ici on les mets dans un tableau d'objet (car on sait qu'il n'y en a pas beaucoup[1]) qui sont donc des objets « libres » de tout contexte BD, SQL…

```
while(rs.next() ) {
  id = rs.getInt("PK_idMarque"); // ou rs.getInt(1)
  marque = rs.getString("marque"); // ou rs. getString(2)
  tab[pos]=new Marque(id, marque);                pos++;
 }
```

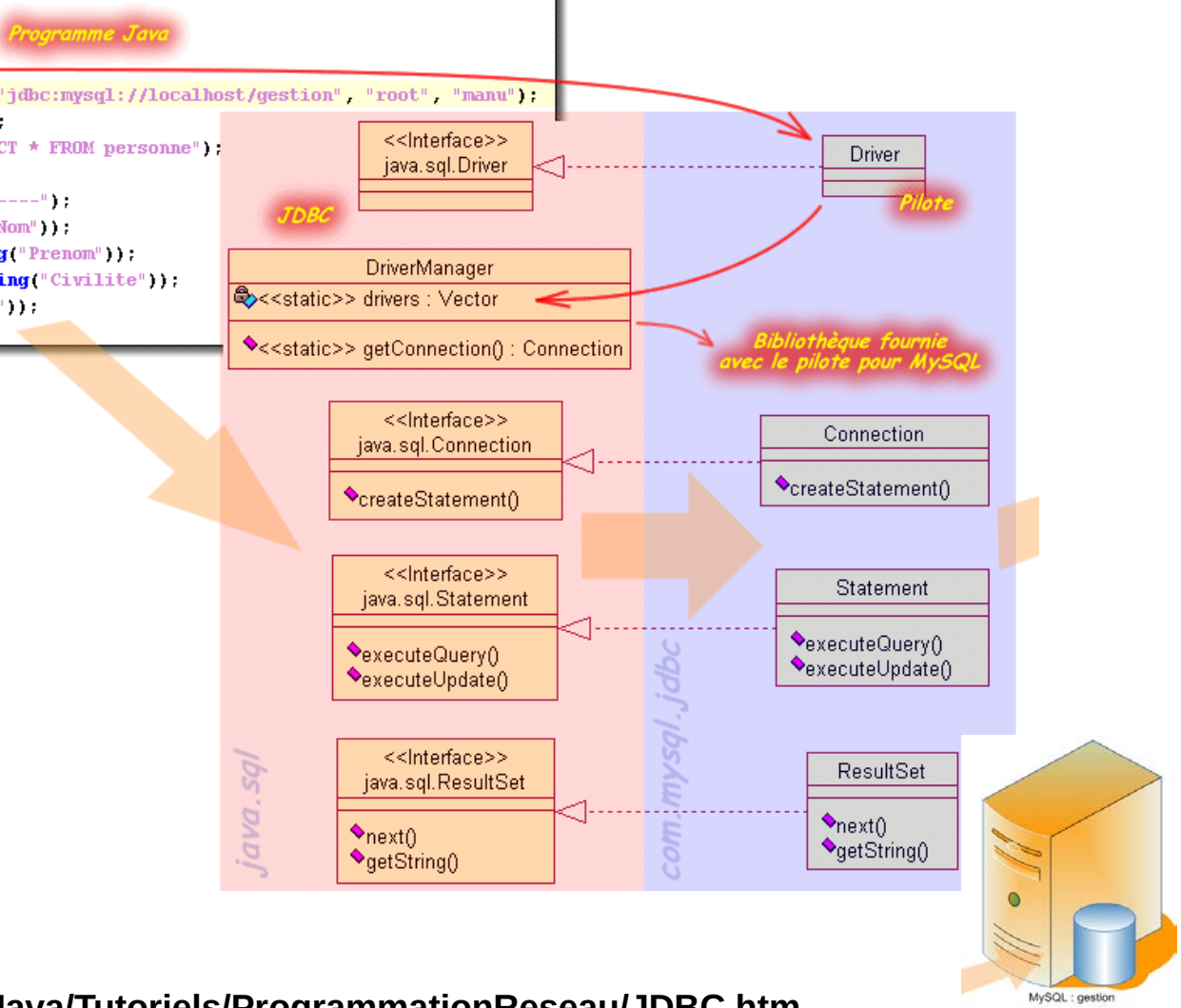
- rs pointe donc sur **1** enregistrement, et il faut alors récupérer la valeur de chaque colonne (soit par son nom, soit par son ordre de déclaration dans la requête SQL CREATE).

(1) On pourrait imaginer :

- soit traiter un enregistrement à la fois (a l'intérieur du while)
- Soit limiter les résultats des requêtes (on ajoute a la fin de la requête : LIMIT [Offset], NbEnregistrement), qui permet de restreindre les résultats de la requête : nbEnregistrement nb maxi d'enregistrement, decale de « offset » enregistrement (pour afficher en plusieurs fois le contenu d'une table

Classes Java Générales & Principe

```
public static void main(String[] args) {
   try {
//      Class.forName("org.gjt.mm.mysql.Driver");
     Class.forName("com.mysql.jdbc.Driver");
     Connection connexion = DriverManager.getConnection("jdbc:mysql://localhost/gestion", "root", "manu");
     Statement instruction = connexion.createStatement();
     ResultSet résultat = instruction.executeQuery("SELECT * FROM personne");
     while (résultat.next()) {
       System.out.println("---------------------------------");
       System.out.println("Nom : "+ résultat.getString("Nom"));
       System.out.println("Prénom : "+ résultat.getString("Prenom"));
       System.out.println("Civilité : "+ résultat.getString("Civilite"));
       System.out.println("Age : "+ résultat.getInt("age"));
     }
```



Source : http://manu.kegtux.org/Java/Tutoriels/ProgrammationReseau/JDBC.htm

Annexe

Apparté sur la définition de chemin en Java

- La manipulation de fichier soulève le problème des formats différents suivant les plateforme (Windows – Mac – Linux : / versus \ etc…).
- Une solution est de contourner le problème en utilisant le protocole FILE: lequel utilise la syntaxe web pour décrire un chemin :
  - FILE:///C:/toto.txt
- L'autre solution est d'utiliser les classes File et FileLocator