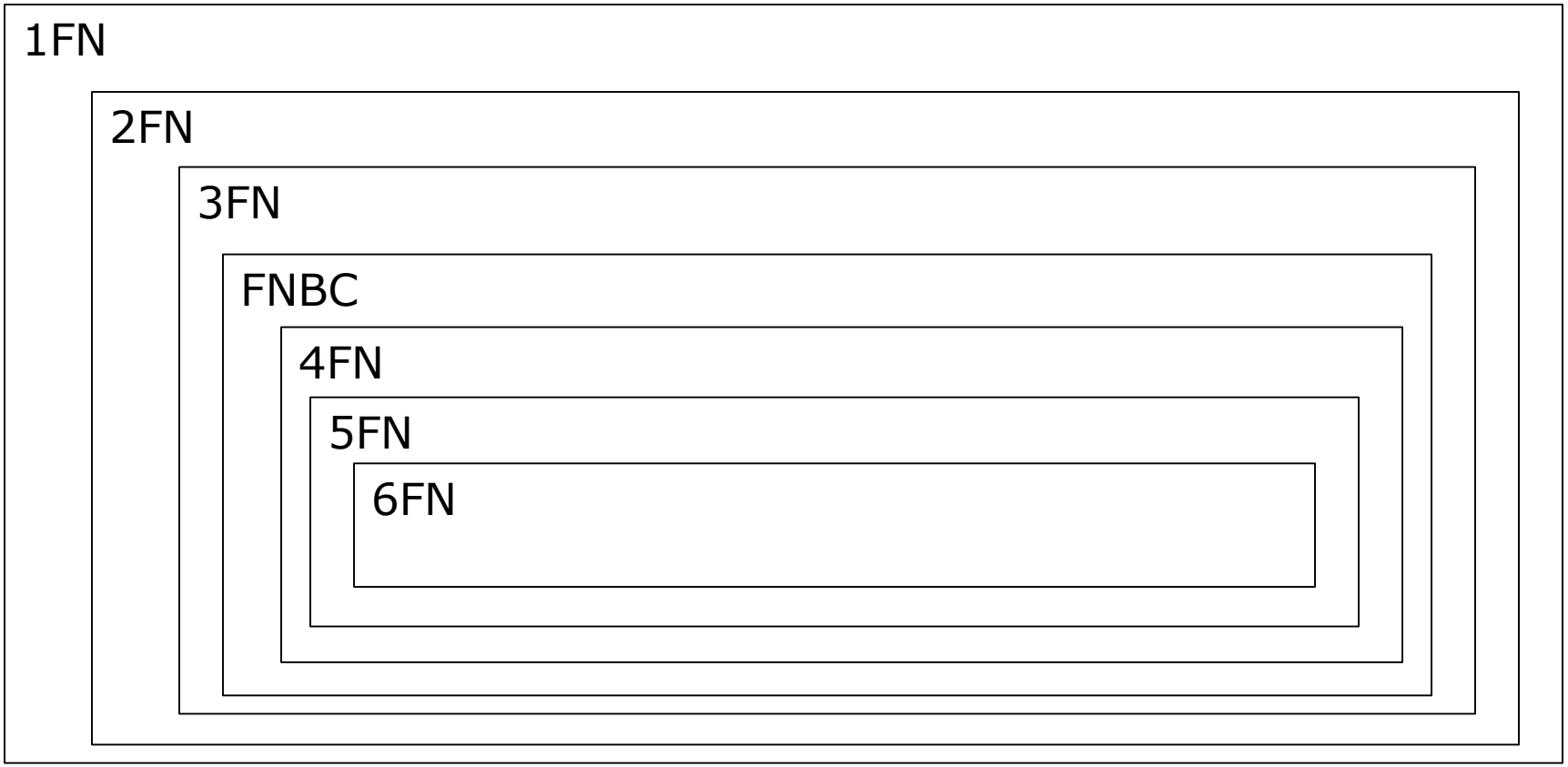
Base de données

Séance 3

1. Normalisation
2. Langage de définition de données (LDD)

Formes normales



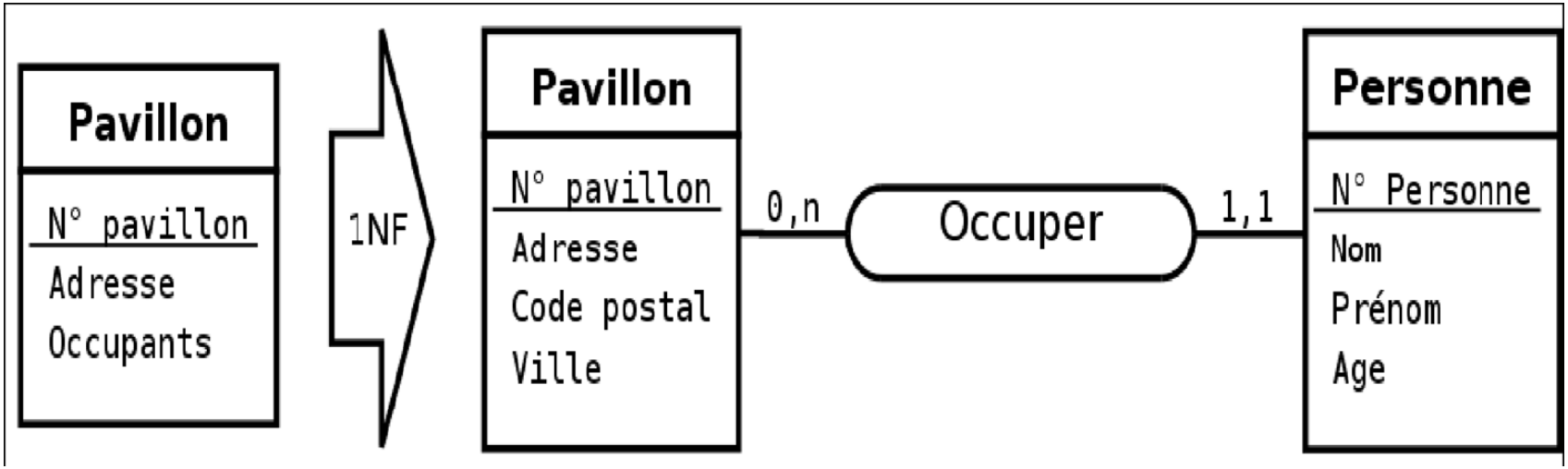
Première forme normale (1FN)

- Une relation est en 1FN si, et seulement si, tout attribut contient une valeur atomique (non multiple, non composée).

- Exemple non en 1FN :
  - Personne(numPersonne, nom, prenom, rueEtVille, prenomsEnfants)

  - Personne(numPersonne, nom, prenom, rue, ville)
  - Prenom(numPrenom, prenom)
  - Prenom_enfant(*numPersonne, numPrenom*)

MCD non en 1FN



Deuxième forme normale (2FN)

- Une relation est en 2FN si, et seulement si
  - elle est en 1FN
  - et si toutes les dépendances fonctionnelles entre la clé et les autres attributs sont élémentaires
- Exemple non en 2FN :
  - Reduction(<u>cru, client</u>, type, remise)

CRU	TYPE	CLIENT	REMISE
CHENAS	A	C1	3%
MEDOC	A	C2	5%
JULIENAS	B	C	4
CHENAS	A	C2	4%

  - Remise(<u>cru, client</u>, remise) et Type(<u>cru</u>, type)

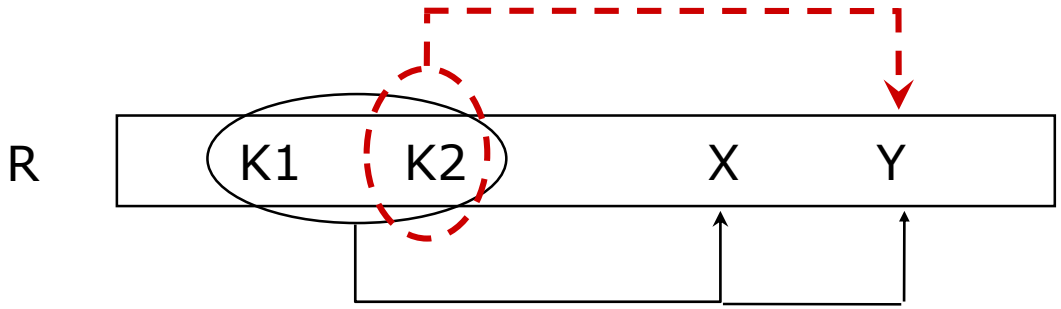
Deuxième forme normale (2FN)

- Une relation est en 2FN si, et seulement si
  - elle est en 1FN
  - et si toutes les dépendances fonctionnelles entre la clé et les autres attributs sont élémentaires
- Exemple non en 2FN :
  - Reduction(cru, client, type, remise)

CRU	TYPE	CLIENT	REMISE
CHENAS	A	C1	3%
MEDOC	A	C2	5%
JULIENAS	B	C	4
CHENAS	A	C2	4%

  - Remise(cru, client, remise) et Type(cru, type)

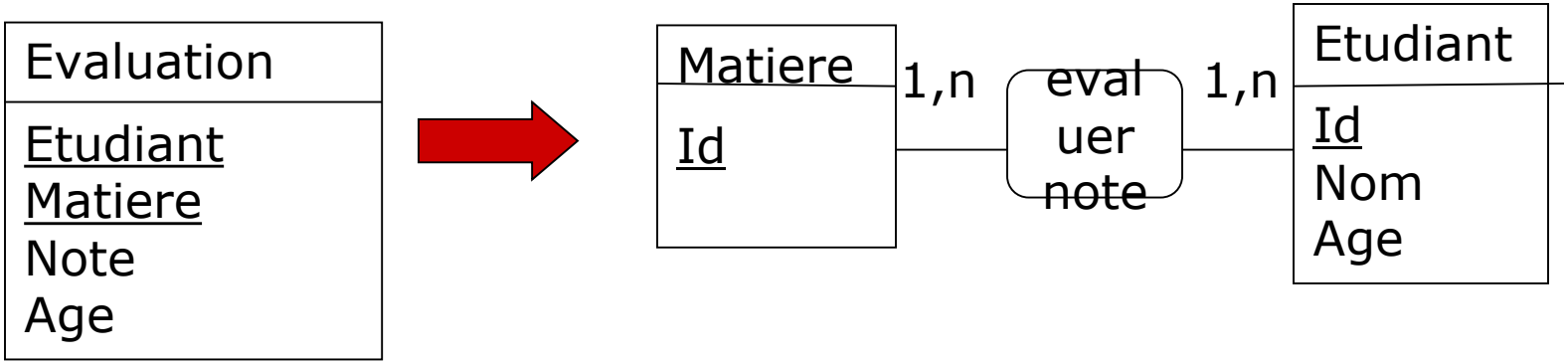
Relation non en 2FN



Une telle relation doit être décomposée en

R1(<u>K1,K2</u>,X) et R2(<u>K2</u>,Y)

MCD non en 2FN



Troisième forme normale (3FN)

- Une relation est en 3FN si, et seulement si
  - elle est en 2FN
  - et si toutes les dépendances fonctionnelles entre la clé et les autres attributs sont directes

- Exemple non en 3FN :
  - Voiture(nv, marque, type, puissance, couleur)

Troisième forme normale (3FN)

- Une relation est en 3FN si, et seulement si
  - elle est en 2FN
  - et si toutes les dépendances fonctionnelles entre la clé et les autres attributs sont directes

- Exemple non en 3FN :
  - Voiture(<u>nv</u>, marque, type, puissance, couleur)

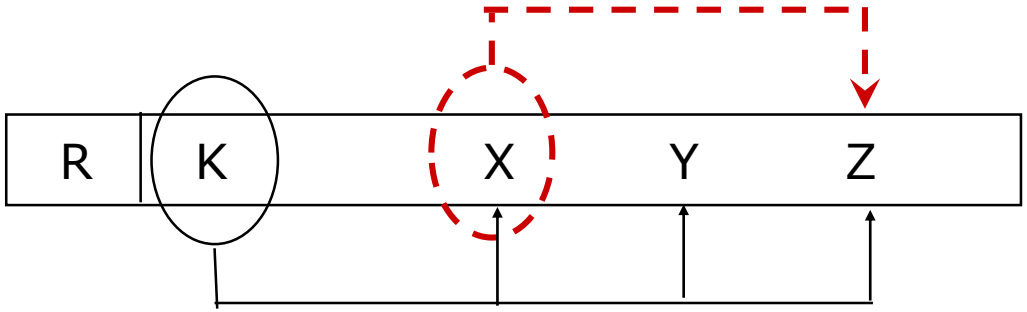
Troisième forme normale (3FN)

- Une relation est en 3FN si, et seulement si
  - elle est en 2FN
  - et si toutes les dépendances fonctionnelles entre la clé et les autres attributs sont directes

- Exemple non en 3FN :
  - Voiture(nv, marque, type, puissance, couleur)

  - Vehicule(nv, type, couleur)
  - Modele(type, marque, puissance)

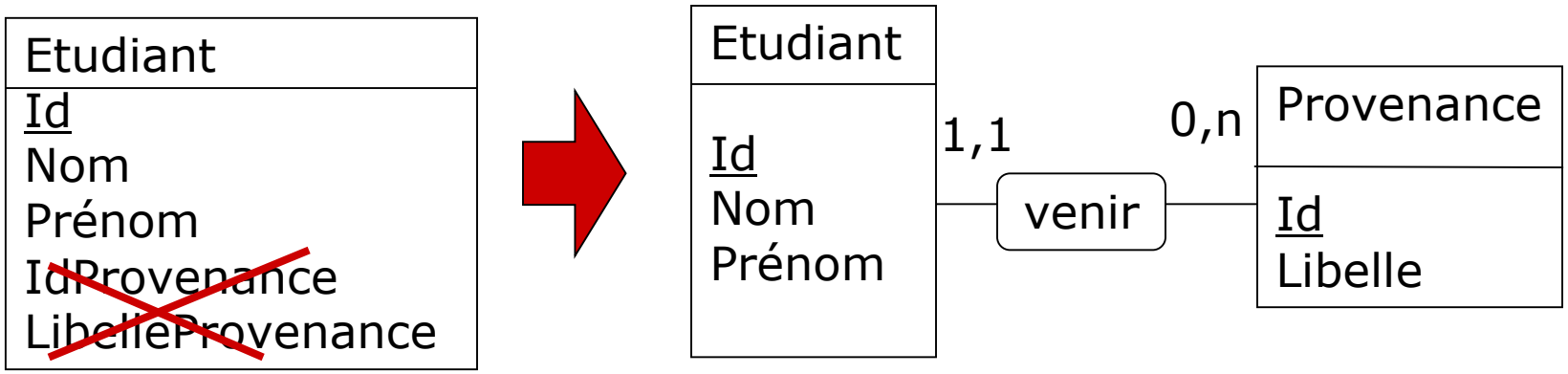
Relation non en 3FN



Une telle relation doit être décomposée en

R1(<u>K, X</u>, Y) et R2(<u>X</u>,Z)

MCD non en 3FN



libelleProvenance ne dépend pas directement de la clé mais d'un autre attribut

Forme normale de Boyce-Codd (BCNF)

- Une relation est en BCNF si, et seulement si
  - elle est en 3FN
  - et si les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé détermine un attribut.

- Exemple
  - Universite(<u>numEtudiant, numMatiere</u>, numEnseignant,note)
  - Si un enseignant n'enseigne qu'une seule matière :

    numEnseignant → numMatiere

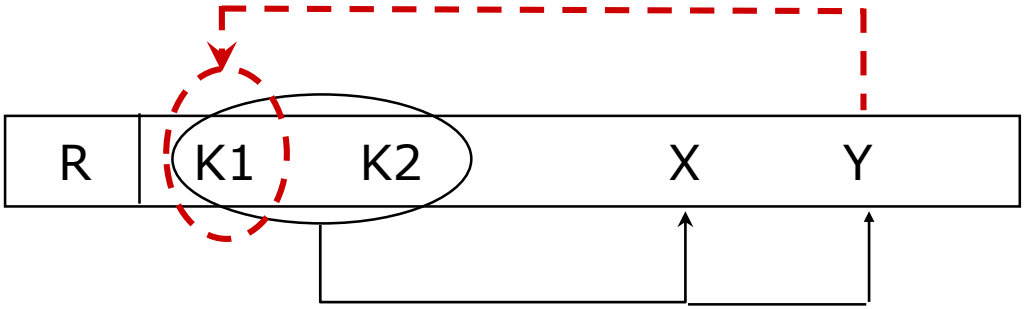
Forme normale de Boyce-Codd (BCNF)

- Universite(numEtudiant, numMatiere, numEnseignant, note) n'est pas en BCNF => redondances de données

numEtudiant	numMatiere	numEnseignant	note
1	5	4	8
2	5	4	14
3	5	4	9,5
1	7	10	13

- Evaluation(numEtudiant, numMatiere, note)
- Enseignement(numEnseignant, numMatiere)

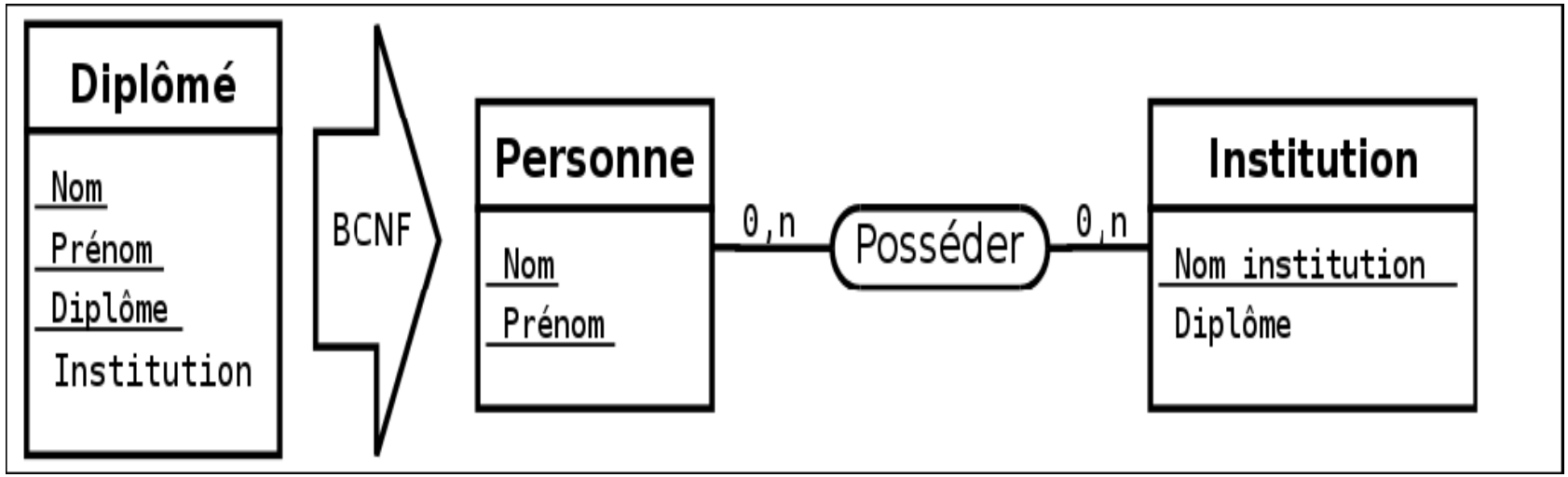
Relation en 3FN mais non en BCNF



Une telle relation peut être décomposée en

R1(<u>K1, K2,</u> X) et R2(<u>Y</u>,K1)

MCD en 3FN mais non en BCNF



Conclusion

- Mnémonique :

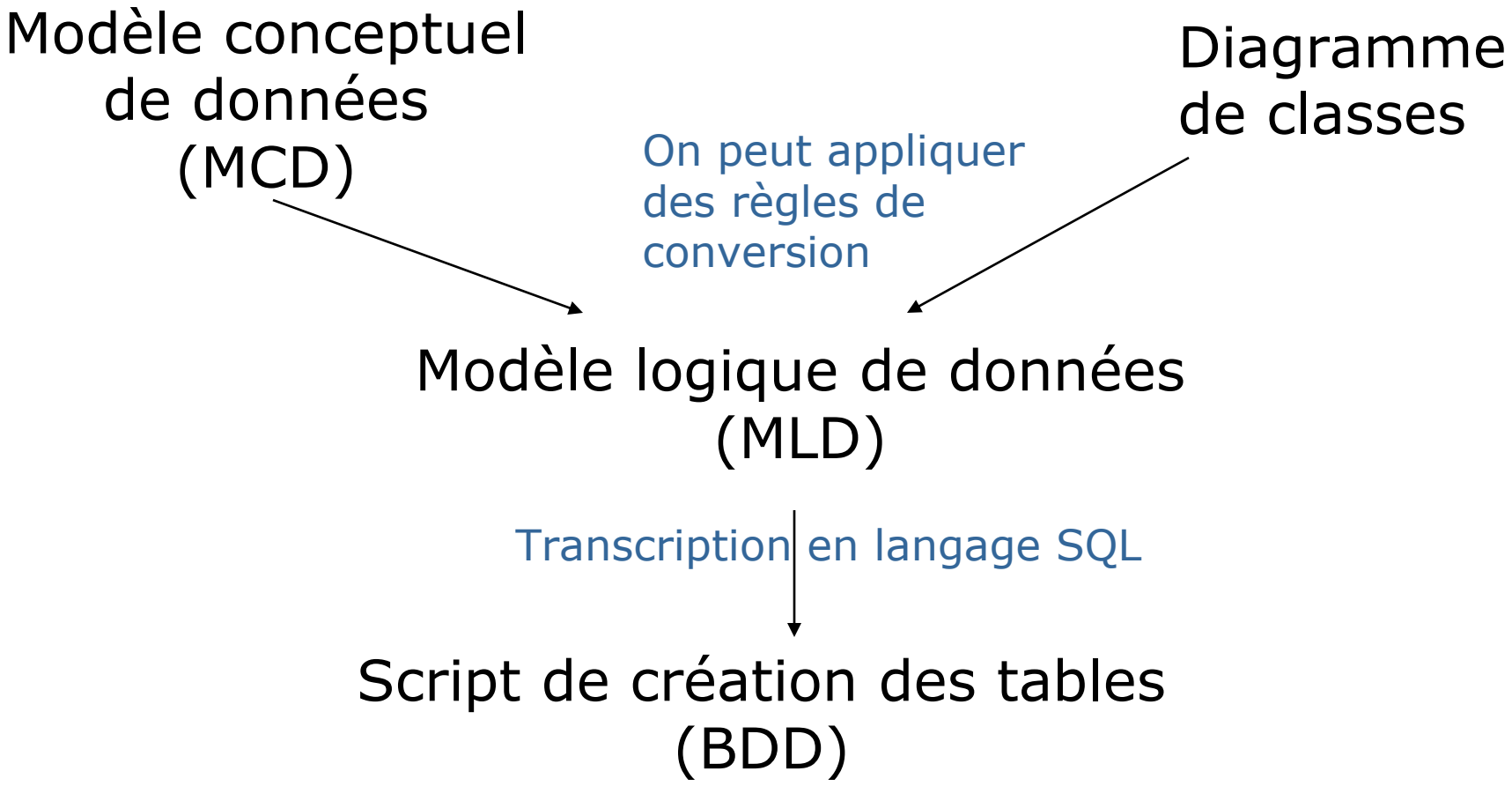
  "The key, the whole key, nothing but the key"
  Chris Date

- Historique :
  - 1FN, 2FN et 3FN par Ted Codd (1971)
  - BCNF par Chris Date (1974)

Base de données

Base de données relationnelle :
Langage de définition de données (LDD)

Du MCD à la base de données



Base de données relationnelle (BDD)

- Les données sont organisées sous forme de **tables** à 2 dimensions dont
  - les lignes sont appelées n-uplet ou enregistrements
  - les colonnes correspondent aux attributs
- Les données sont manipulées par des opérateurs de l'algèbre relationnelle
- L'état cohérent de la base est défini par un ensemble de contraintes d'intégrité
- Le modèle de données est réflexif :
  - Le dictionnaire de données décrit les données sous forme d'une base de données
  - On l'appelle la métabase

Base de données : les objets

- Table
  - Une table enregistre des enregistrements qui décrivent une instance d'une entité
- Vue
  - Les vues sont des résultats d'exploration de données que l'on fait apparaître comme une table.
- Index
  - Un index est une table d'encodage qui optimise l'accès aux données
- User
  - Un objet User représente un utilisateur des données.

Les clefs : primaire et étrangère

- **Clé primaire** :
  - permet d'identifier de manière unique **un** enregistrement dans **la** table
- **Clé étrangère** :
  - permet de vérifier que le champ contient une clef primaire existante d'une autre table.

« *La notion de clef étrangère est directement liée à la notion d'association* »

Langage de définition des données (LDD)

- **Le LDD sert à décrire la nature des données telles qu'elles vont exister**
- CREATE

  Créer des objets de données (des tables, des indexes, des vues)
- DROP

  Supprimer des objets de données
- ALTER

  Modifier la définition des objets de données

Le « moment » d'utilisation du LDD

- Le LDD s'utilise au moment de la constitution de la base de données
- Les instructions DROP s'utilisent principalement :
  - Pour supprimer une table obsolète
  - Avant de recréer une nouvelle version d'une table
- Les instructions ALTER s'utilisent :
  - Pour mettre à jour la structure d'un objet sans perte de données (mise à jour progressive)

SQL : Structured Query Langage

- Le SQL est un langage standardisé qui regroupe les quatre sous-langages :
  - LDD : langage de définition de données
    CREATE, DROP, ALTER
  - LMD : langage de manipulation des données
    SELECT, UPDATE, INSERT, DELETE
  - LCD : langage de contrôle de données
    GRANT, REVOKE
  - LCT : langage de contrôle de transactions
    COMMIT, SAVEPOINT, ROLLBACK

Objet TABLE : Définition

- Structure de données composée de :
  - colonnes
  - lignes appelées n-uplets ou **enregistrements**
- Chaque colonne doit être :
  - nommée
  - typée
- Exemple : table personne, étudiant …

Exemple

Personne (id : INT, nom : CHAR(50), prenom : CHAR(50))

id	**nom**	**prenom**
1	Durand	Caroline
2	Dubois	Jacques
3	Dupont	Lisa
4	Dubois	Rose-Marie

L’instruction CREATE TABLE

- **Attribuer un nom à la table** (le nom doit être unique dans la base de données)
- **Attribuer des colonnes**
  - Le nom de la colonne est unique dans la table
  - Attribuer un type à chaque colonne
  - Définir des contraintes sur ces colonnes
- Spécifier des caractéristiques de stockage
- Spécifier une requête pour créer la table avec des données

L’instruction CREATE TABLE : Syntaxe

CREATE TABLE <*nom*>
(<*définition colonne*> | <*définition contrainte*>,...)
[<*spécification stockage*>]
[<*données provenant d’une requête*>];

L'instruction CREATE TABLE : Exemple

```
CREATE TABLE personne(
    nom      VARCHAR(30),
    age      INTEGER,
    salaire DECIMAL(10,2)
);
```

Définition de colonne : Les types

- Syntaxe générale de type :
  - <*nom type*>[(<*dimensionnement*>)]
- Type alphanumérique :
  - CHAR(n) : chaîne à longueur fixe
  - VARCHAR(n) : chaîne à longueur optimisée
- Exemple :
  - nom VARCHAR(30)

Définition de colonne : Les types

- Types numériques :
  - DECIMAL(p,s): nombre décimal
  - INTEGER : entier
  - SMALLINT : entier court

- Exemples :
  - age SMALLINT
  - salaire DECIMAL(10,2)

Définition de colonne : Les types

- Type de gestion du temps et des dates :
  - DATE : date du calendrier grégorien (YYYY-MM-DD)
  - TIMESTAMP : combiné date temps sous forme compacte : YYYY-MM-DD HH:MM:SS
- Exemple
  - dateCommande DATE

Les contraintes

- Contrainte de colonne
  - Associée à une colonne de la table
- Contrainte de table
  - Associée à plusieurs colonnes de la table, ou à la totalité des colonnes

« *Il est utile de donner aux contraintes un nom qui exprime la raison de la contrainte* »

Les contraintes des colonnes

- S'applique sur **une seule** colonne
  - Nullité de colonne : NULL / NOT NULL
  - Unicité de valeur dans une colonne : UNIQUE
  - Clé primaire : PRIMARY KEY
  - Vérification sur un ensemble de valeur : CHECK
  - Clé étrangère : FOREIGN KEY … REFERENCES

« *Une contrainte définit des restrictions sur les valeurs possibles* »

Exemple de contraintes de colonnes

```
CREATE TABLE personne(
    id INT PRIMARY KEY,
    nom VARCHAR(20),
    salaire DECIMAL(12,2) NOT NULL,
    codePostal DECIMAL(5),
    CONSTRAINT valeur_cp CHECK (codePostal BETWEEN 00001
    AND 99999)
);
```

Valeur par défaut d'une colonne

- Permet d'attribuer une **valeur par défaut** à une colonne si une requête d'ajout de données ne la fournit pas.
- Sans spécification la valeur par défaut est NULL
- Peut être une valeur, une valeur calculée standard,...

```
dateJour DATE DEFAULT '1990-09-01'
```

Les contraintes de table

- S’applique sur plusieurs colonnes
  - Clé primaire, si elle est « composite »
  - Clé étrangère, si elle est « composite »

Contraintes de table

- PRIMARY KEY (colonne1, colonne2, …)
  - désigne la concaténation des attributs cités comme clé primaire de la table
- UNIQUE (colonne1, colonne2, …)
  - désigne la concaténation des attributs cités comme clé secondaire de la table
- FOREIGN KEY (colonne1, colonne2, …) REFERENCES table [(colonne1, colonne2,…)]
- CHECK (condition)
- ON DELETE CASCADE / ON DELETE SET NULL

Exemple de contraintes de table

```
CREATE TABLE personne(
    code VARCHAR(4),
    nom VARCHAR(20),
    salaire DECIMAL(12,2),
    codePostal CHAR(5),
    CONSTRAINT pk_personne PRIMARY KEY(code,nom)
);
```

Modification de la structure d'une table

- On pourrait supprimer la table et la recréer

…?

On perd les données qui sont déjà dedans

Modification de la structure d'une table

- Modifier la structure, c'est :
  - supprimer, modifier les caractéristiques ou ajouter des colonnes
  - supprimer, ajouter, modifier des contraintes
- Syntaxe générale

ALTER TABLE <nom_table>
[<add>][<modify>][<drop>]

Exemples de modification

```
ALTER TABLE tab1 ADD (col1 type1, col2 type2, …);

ALTER TABLE tab1 MODIFY (col1 type1, col2 type2, …);

ALTER TABLE tab1 DROP CONSTRAINT cont1;

ALTER TABLE tab1 ADD CONSTRAINT cont1 FOREIGN KEY
   (col11,…, col1n) REFERENCES tab2(col21, …, col2n);

ALTER TABLE tab1 ADD CONSTRAINT cont1 PRIMARY KEY
   (col1, col2, …);
```

Destruction d'une table

- Syntaxe générale :

DROP TABLE <nom_table>

- Exemple :

```
DROP TABLE personne ;
```

- Problèmes liés à la destruction d'une table :
  - Certaines contraintes peuvent interdire la destruction d'une table

MLD Cours …

- Enseignant(id, nom, prenom)
- Etudiant(id, nom, prenom)
- Salle(id, nbPlaces)
- Matiere(id, intitule)
- Cours(id, nom, *#idEnseignant, #idMatiere, #idSalle*)
- Inscription(*#idEtudiant, #idMatiere*, evaluation)

BDD

```
CREATE TABLE enseignant(
    id INT PRIMARY KEY,
    nom VARCHAR(30),
    prenom VARCHAR(30));

CREATE TABLE etudiant(
    id INT PRIMARY KEY,
    nom VARCHAR(30),
    prenom VARCHAR(30));

CREATE TABLE salle(
    id INT PRIMARY KEY,
    nbPlaces INT CHECK (nbPlaces > 0));

CREATE TABLE matiere(
    id INT PRIMARY KEY,
    intitule VARCHAR(30));
```

BDD : contraintes dans CREATE

```
CREATE TABLE cours(
    id INT PRIMARY KEY,
    nom VARCHAR(30),
    idEnseignant INT,
    idMatiere INT,
    idSalle INT,
    FOREIGN KEY fk_enseignant(idEnseignant)REFERENCES enseignant(id),
    FOREIGN KEY fk_matiere(idMatiere) REFERENCES matiere(id),
    FOREIGN KEY fk_salle(idSalle) REFERENCES salle(id));
```

- Cours(id, nom, *#idEnseignant, #idMatiere, #idSalle*)

BDD : contraintes avec ALTER

```
CREATE TABLE cours(
    id INT,
    nom VARCHAR(30),
    idEnseignant INT,
    idMatiere INT,
    idSalle INT);

ALTER TABLE cours ADD CONSTRAINT pk_cours PRIMARY KEY (id);
ALTER TABLE cours ADD CONSTRAINT fk_enseignant FOREIGN KEY
    (idEnseignant) REFERENCES enseignant(id);
ALTER TABLE cours ADD CONSTRAINT fk_matiere FOREIGN KEY (idMatiere)
    REFERENCES matiere(id);
ALTER TABLE cours ADD CONSTRAINT fk_salle FOREIGN KEY (idSalle)
    REFERENCES salle(id);
```

- Cours(id, nom, *#idEnseignant, #idMatiere, #idSalle*)

BDD

```
CREATE TABLE inscription(
    idEtudiant INT,
    idMatiere INT,
    evaluation DECIMAL(2,1) NOT NULL,
    FOREIGN KEY fk_etudiant(idEtudiant)REFERENCES etudiant(id),
    FOREIGN KEY fk_matiere(idMatiere) REFERENCES matiere(id),
    CONSTRAINT pk_inscription PRIMARY KEY (idEtudiant,idMatiere));
```

- Inscription(*#idEtudiant, #idMatiere*, evaluation)

- **Exercice : CREATE TABLE + ALTER TABLE**