MIAGE-BDAV – Bases de données avancées

Modélisation et contraintes

THION Romuald

https://romulusfr.github.io/unc-miage-bdav/

Vendredi 27 octobre 2023

Plan

De la modélisation conceptuelle aux relations

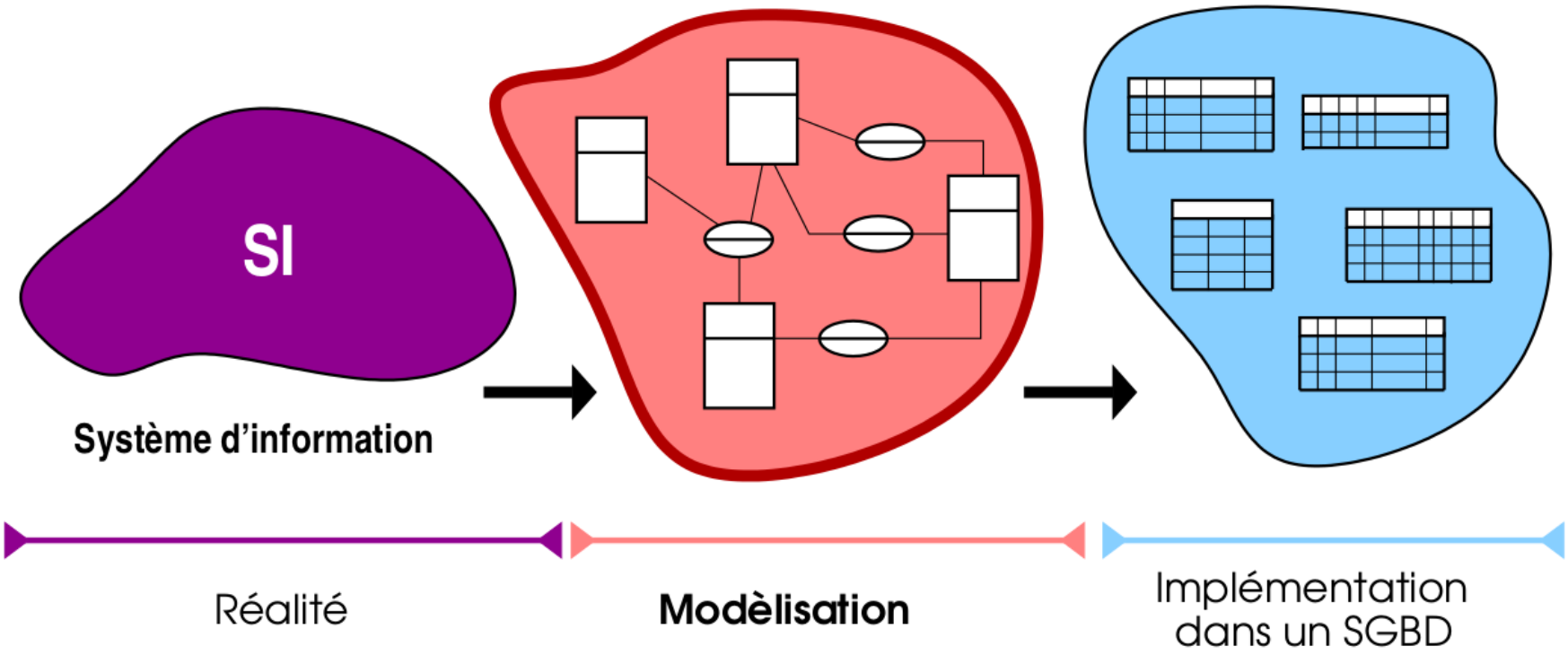
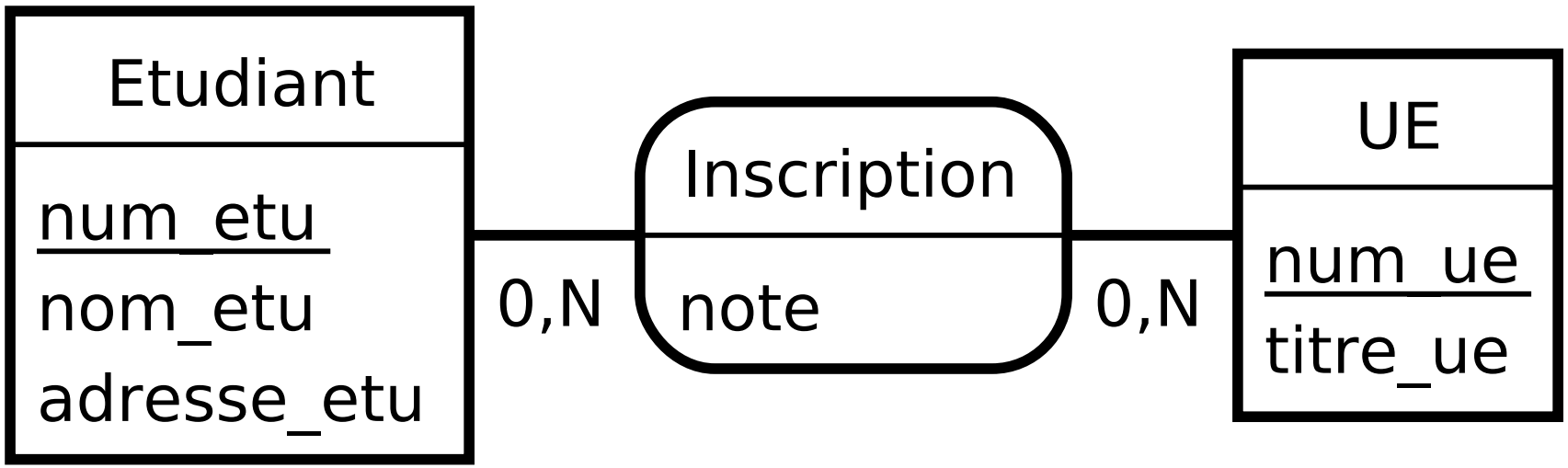


Figure – Modélisation d'un SI, extrait de Only SQL

Les diagrammes Entités-Associations (E/A)



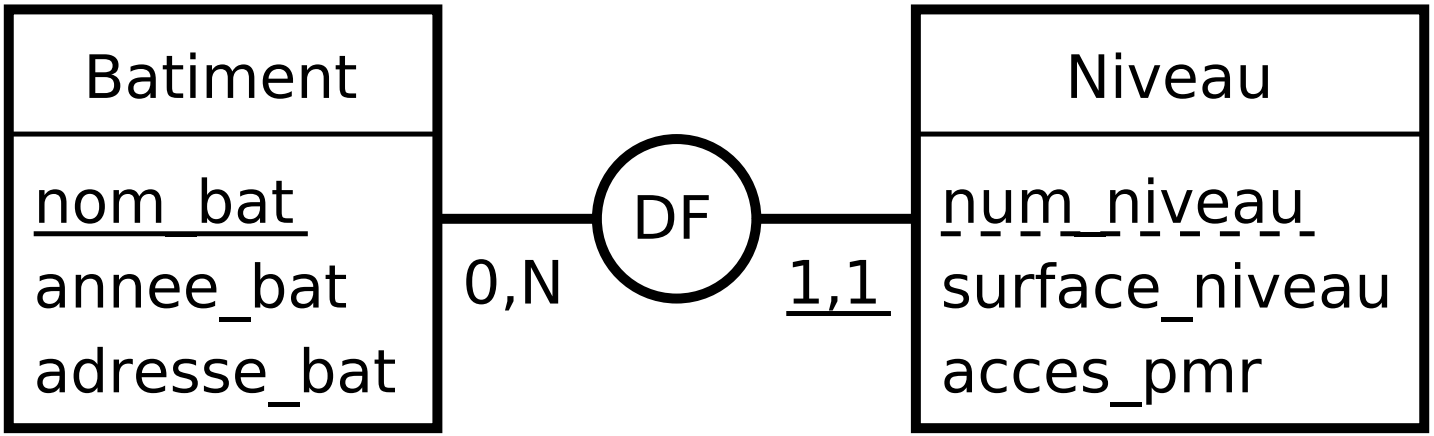
Voir le chapitre *Modèle conceptuel des données* de Only SQL.

L'entité faible

Certaines entités dépendent *intrinsèquement* d'autres :

- les pistes d'un album dépendent de l'album ;
- les salles d'un établissement dépendent de l'étage qui eux mêmes dépendent d'un bâtiment ;

Dans ces cas là, l'identifiant de l'entité dépendante est relatif à l'identifiant de l'identité maître.

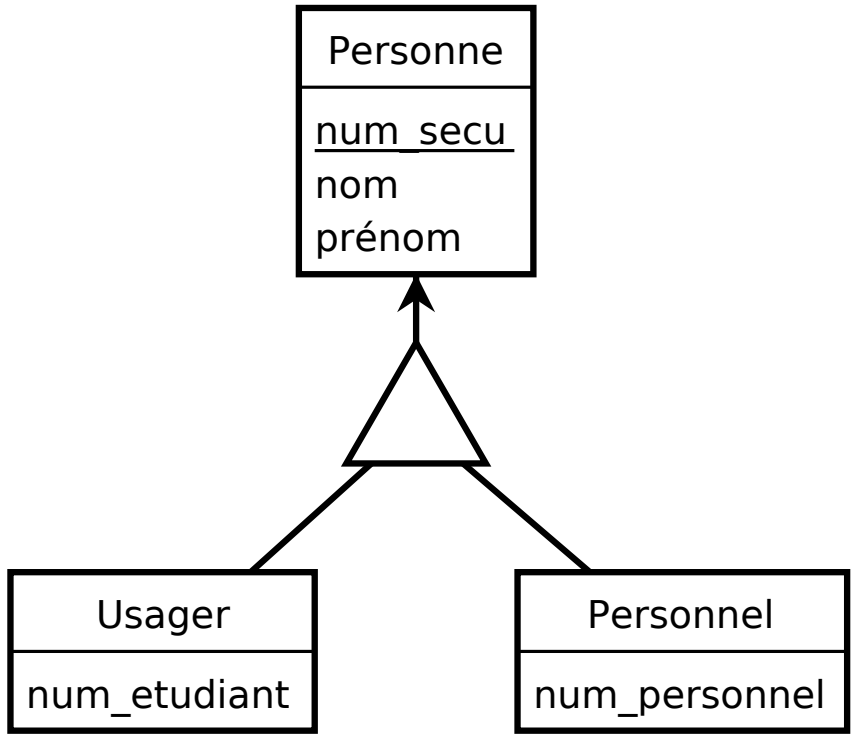


La spécialisation (ou héritage)

Même idée qu'en objet.

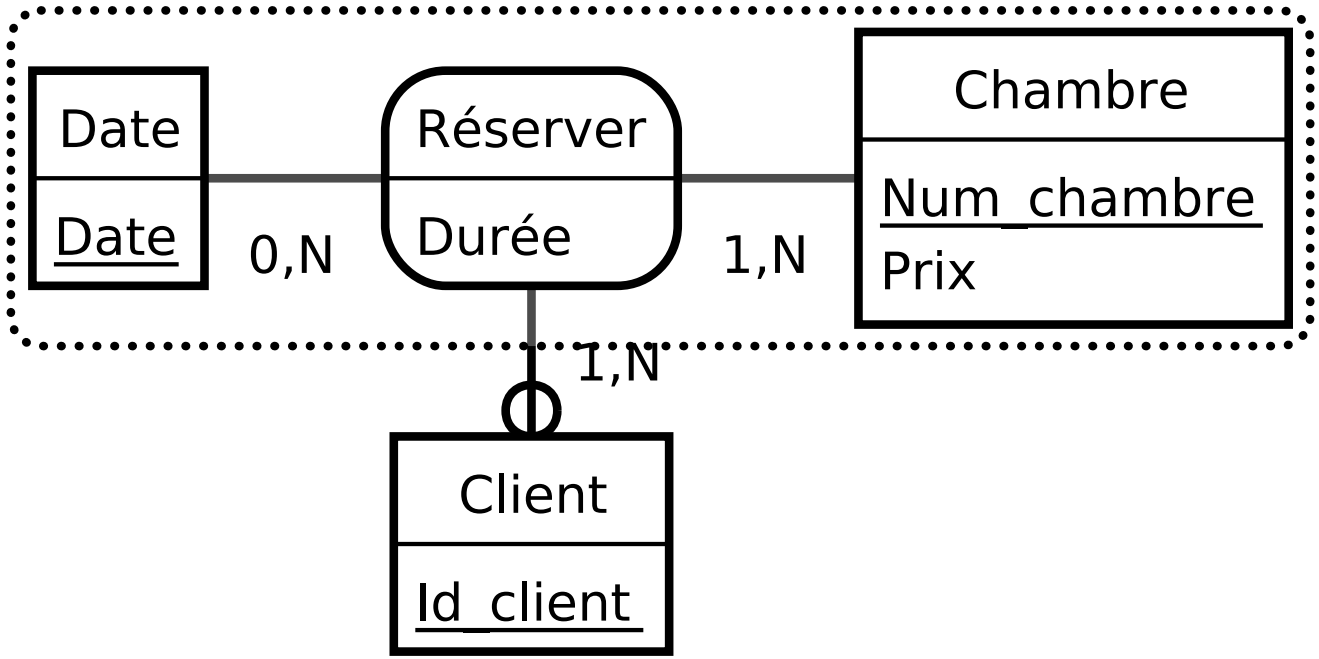
- on peut préciser les contraintes de *totalité* et *d'exclusion* ;
- différents codages dans le modèle relationnels sont possibles

On peut le voir comme un cas dégénéré d'entité faible *sans identifiant de l'entité dépendante*.



L'agrégat (ou pseudo-entité)

Avec les cardinalités *au plus proche* on ne peut pas définir d'association ternaires du type *à une date et chambre fixée il n'y a qu'un seul client*. On aimerait alors associer une association à une autre association. Pour cela, on crée un *agrégat*.



Objectifs

Modéliser

Modéliser consiste à définir un monde abstrait qui coïncide avec les manifestations apparentes du monde réel.

- il s'agit donc de déterminer l'ensemble des attributs, des relations et des contraintes qui constitueront le modèle.

Nous allons voir :

- Quelles sont les propriété attendues d'une bonne modélisation ;
- Comment les obtenir.

Exemple

NumEt	NomEt	Adresse	NumUE	Titre	Note
124	Jean	Paris	F234	Philo I	A
456	Emma	Lyon	F234	Philo I	B
789	Paul	Marseille	M321	Analyse I	C
124	Jean	Paris	M321	Analyse I	A
789	Paul	Marseille	CS24	BD I	B

Table – La relation universelle de tous les attributs de l'univers $\mathcal{U}$

Comment évaluer ce schémas ? Est-il bon ? Pourquoi ? Selon quels critères ?

Anomalie de *modification*

NumEt	NomEt	Adresse	NumUE	Titre	Note
124	Jean	Paris	F234	Philo I	A
456	Emma	Lyon	F234	Philo I	B
789	Paul	Marseille	M321	Analyse I	C
124	Jean	Paris	M321	Analyse I	A
789	Paul	Marseille	CS24	BD I	B

Anomalie de *modification*

Une modification sur une ligne peut nécessiter des modifications sur d'autres lignes.

Anomalie de *suppression*

NumEt	NomEt	Adresse	NumUE	Titre	Note
124	Jean	Paris	F234	Philo I	A
456	Emma	Lyon	F234	Philo I	B
789	Paul	Marseille	M321	Analyse I	C
124	Jean	Paris	M321	Analyse I	A
789	Paul	Marseille	CS24	BD I	B

Anomalie de *suppression*

Certaines informations dépendent de l'existence d'autres informations.

Anomalie d'*insertion*

NumEt	NomEt	Adresse	NumUE	Titre	Note
124	Jean	Paris	F234	Philo I	A
456	Emma	Lyon	F234	Philo I	B
789	Paul	Marseille	M321	Analyse I	C
124	Jean	Paris	M321	Analyse I	A
789	Paul	Marseille	CS24	BD I	B
145	Evariste	Aubenas	???	???	???

Anomalie d'*insertion*

La possibilité d'enregistrer un tuple implique la connaissance de toutes les informations qui lui sont liées : problème de valeurs manquantes.

Comment formaliser tout ça ?

Le moyen qui permet de comprendre et résoudre ces problèmes est l'étude des dépendances et de la normalisation via les dépendances fonctionnelles et les formes normales.

Définition

- La forme la plus fréquemment rencontrée de dépendances.
- Formalisent la notion de clef (identifiant) d'une relation.
- Permettent de définir les « bon » schémas (sans redondance)

Syntaxe des dépendances fonctionnelles

Une *Dépendance Fonctionnelle (DF)* sur un schéma de relation $R \subseteq \mathcal{U}$ est une expression de la forme

$$R : X \rightarrow Y, \text{ avec } X, Y \subseteq R$$

- Une DF $X \rightarrow Y$ est dite triviale si $Y \subseteq X$
- Une DF standard si $X \neq \emptyset$.

Les dépendances fonctionnelles

Soient $\mathcal{U}$ un ensemble d'attributs et $\mathcal{D}$ un domaine et $R \subseteq \mathcal{U}$:

- un tuple t de R est une fonction $R \to \mathcal{D}$
- une instance r de R est un ensemble fini de tuples
- la projection de t sur $X \subseteq R$ notée [a] $t[X]$ est la restriction de t à X

a. On écrit plutôt $t_{|X}$ en mathématiques usuelles

Sémantique des dépendances fonctionnelles

Soit r une instance de relation sur R. Une DF $R : X \to Y$ est *satisfaite* dans r, noté $r \models X \to Y$, *ssi*

$$\forall t_1, t_2 \in r. t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$

On dit aussi que X *détermine (fonctionnellement)* Y dans r.

Les dépendances fonctionnelles

Exemple

r	NumEt	NomEt	Adresse	NumUE	Titre	Note
	124	Jean	Paris	F234	Philo I	A
	456	Emma	Lyon	F234	Philo I	B
	789	Paul	Marseille	M321	Analyse I	C
	124	Jean	Paris	M321	Analyse I	A
	789	Paul	Marseille	CS24	BD I	B

- $r \models \texttt{NumEt} \rightarrow \texttt{NomEt}$ et $r \models \texttt{NumEt}, \texttt{NumUE} \rightarrow \texttt{Note}$
- $r \models \texttt{Adresse} \rightarrow \texttt{NumEt}$ (†)
- $r \not\models \texttt{NumEt} \rightarrow \texttt{NumUE}$ et $r \not\models \texttt{NumUE} \rightarrow \texttt{Note}$

Exercice

Soit R un schéma, r une instance de R et $F = X \rightarrow Y$ une DF sur R.

Cas limites

Quelles sont les contraintes imposées sur r pour les cas suivants ?

- $r \models \emptyset \rightarrow \emptyset$
- $r \models \emptyset \rightarrow R$
- $r \models R \rightarrow \emptyset$
- $r \models R \rightarrow R$

Vérifier la satisfaction en SQL

- Donner une requête SQL qui vérifie si $r \models F$ avec le cas échéant les contre-exemples.
- Tester $AA, \ldots, AH \rightarrow AI$ et $AA, \ldots, AI \rightarrow AJ$ sur le dataset `armstrong.sql`.

Les dépendances fonctionnelles

La notion de clef

Une clef peut-être définie de deux manières *équivalentes* :

- Une clef est un ensemble d'attributs X *qui ne prend jamais deux fois la même valeur* dans r
- Une clef est un ensemble d'attributs qui détermine tout R, c'est-à-dire tel que $r : X \rightarrow R$

Clef primaire (*primary key*)

Une clef primaire est simplement *une* clef parmi les autres (appellées *clefs candidates*), choisie par le concepteur pour sa simplicité ou son aspect naturel.

Les DFs dans les SGBDs

Les contraintes de clefs constituent les principales contraintes d'intégrités des SGBDs :

- définies sur les tables (via les contraintes `UNIQUE` et `PRIMARY KEY`)
- garanties lors de toutes les modifications
- sont un cas particulier des DFs (†)

Contraintes de clefs

```
CREATE TABLE Inscrit (
   -- Traduit la DF NumEt, NumUE -> Note
   PRIMARY KEY (NumEt, NumUE),
   NumEt INTEGER REFERENCES Etudiant(NumEt),
   NumUE INTEGER REFERENCES UE(NumUE),
   Note DECIMAL(4,2) NOT NULL
);
```

Les DFs dans PostgreSQL

PostgreSQL utilise les DFs pour optimiser les requêtes

```
DROP STATISTICS dfs;
CREATE STATISTICS dfs (dependencies) ON
  nom, filiere, categorie, groupe_td, groupe_tp
FROM inscrits ;

ANALYZE inscrits;

SELECT * FROM pg_statistic
WHERE starelid = 'inscrits'::regclass;

SELECT * FROM pg_statistic_ext_data;
```

Les formes normales

La solution aux problèmes des anomalies consiste à normaliser la relation en la décomposant. Cette décomposition s'appuie sur les dépendances fonctionnelles qui existent entre les attributs.

Les formes normales permettent de spécifier formellement la notion *intuitive* de « bon schéma »

L'idée générale est de n'avoir que les clés à vérifier et d'éliminer au maximum des DF qui ne définissent pas des clés.

Inclusion des formes normales

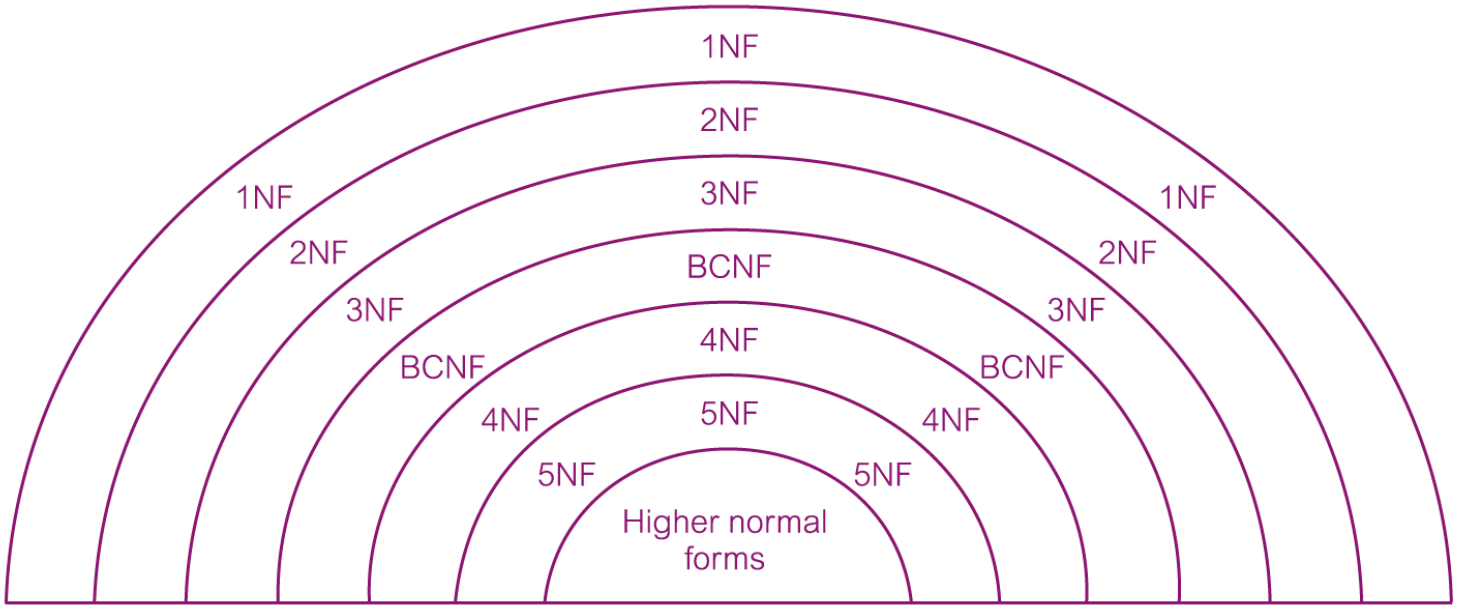


Figure 13.7
Diagrammatic illustration of the relationship between the normal forms.

Figure – Pour les DFs, plusieurs Formes Normales (FN) de plus en plus restrictives (1FN, 2FN, 3FN, FN de Boyce-Codd – FNBC). Au delà, d'autres types de dépendances sont nécessaires.

Les formes normales

Soit R un schéma de relation et F un ensemble de DF définies sur R.
Un schéma de BD est en nFN si tous ses schémas de relations le sont.

Des contraintes de plus en plus restrictives

1FN toutes les valeurs des attributs sont atomiques

2FN R est en 1FN et aucun attribut non clef ne dépend partiellement d'une clef candidate.
E.g., $\{AB \to C, B \to C\}$ *n'est pas en 2FN.*

3FN R est en 2FN et toutes les DFs sont directes : tout attribut non clef dépend directement d'une clé (sans transitivité).
E.g., $\{A \to B, B \to C\}$ *est en 2FN mais pas en 3FN.*

FNBC R est en FN de Boyce-Codd ssi pour chaque DF non-triviale $X \to A$ de F, X contient une clef de R.
E.g., $\{AB \to C, C \to B\}$ *est en 3FN mais pas en FNBC.*

La FNBC : une forme idéale

La FNBC est, *pour les DF*, la forme idéale d'un schéma de BD « *La clef, toute la clef et rien que la clef.* »

Les DFs de la forme $X \rightarrow A$ où X est clef sont exactement celles que les SGBD implémentent avec les `PRIMARY KEY` (ou plus généralement avec les contraintes `UNIQUE` et `NOT NULL`)

La FNBC : une forme idéale

Formalisation de la notion de redondance avec les DF

Soit $r \models F$ une instance de R qui vérifie un ensemble de DFs F. Si ils existent $X \rightarrow A \in F$ et $t_i \neq t_j \in r$ t.q. $t_i[XA] = t_j[XA]$, alors la relation r est redondante [a].

a. On pourrait définir assez similairement les anomalies d'insertion et de mise-à-jour.

Théorème : les propriétés suivantes sont équivalentes

- R est en FNBC par rapport à F
- R n'a pas de problème de redondances par rapport à F
- (R n'a pas de d'anomalies d'insertion par rapport à F)
- (R n'a pas de d'anomalies de mise-à-jour par rapport à F)

Reprise de l'exemple

	NumEt	NomEt	Adresse	NumUE	Titre	Note
t_1	124	Jean	Paris	F234	Philo I	A
t_2	456	Emma	Lyon	F234	Philo I	B
t_3	789	Paul	Marseille	M321	Analyse I	C
t_4	124	Jean	Paris	M321	Analyse I	A
t_5	789	Paul	Marseille	CS24	BD I	B

Vers un bon schéma

Ici, beaucoup de redondances, car $\texttt{NumEt} \rightarrow \texttt{NomEt}, \texttt{Adresse}$ et $\texttt{NumUE} \rightarrow \texttt{Titre}$ et $\texttt{NumEt}, \texttt{NumUE} \rightarrow \texttt{Note}$.

Mais on voit « un bon schéma » commencer à apparaître !

Les dépendances d'inclusion

Définition

Une *Dépendance d'Inclusion* (DI) est une expression de la forme, où R, S sont des symboles de relation et X, Y des séquences d'attributs de même longueur :

$$R[X] \subseteq S[Y]$$

Soient r, s des instances de R et X. La DI $R[X] \subseteq S[Y]$ est satisfaite par r, s *ssi*

$$\forall u \in r, \exists v \in s, u[X] = v[Y]$$

Les dépendances d'inclusion permettent de définir de nouvelles notions de formes normales (e.g., IDNF), mais dans le cas général :

- les problèmes de décision sont plus difficile
- les intéractions avec les DFs sont complexes et rendent le problème de la l'implication indécidable

Les dépendances d'inclusion

Définition

Une DI de la forme $R[X] \subseteq S[Y]$ où de plus Y est une clef de S, c'est-à-dire quand la dépendance fonctionnelle $Y \rightarrow S$ est imposée, est appellée *clef étrangère* (de X vers S).

Les SGBD implémentent ces restrictions des DI :

- introduites avec les mot-clef `FOREIGN KEY` et `REFRENCES`
- permettent de préciser résoudre les conflits en cas de suppression sur S (`CASCADE`, `SET NULL`, `SET DEFAULT`)
- la contrainte que Y soit clef permet d'assurer une vérification *efficace* lors des mises-à-jour de r.

Normaliser

L'activité qui consiste, étant donné un ensemble de DF [1], à avoir un schéma *en bonne forme normale*.

- Soit de façon calculatoire, avec l'aide des DFs et d'algorithmes :
  - *Par synthèse* : on génère les schémas de relation à partir des DFs
  - *Par décomposition* : on raffine successivement $\mathcal{U}$
- Soit par construction, avec les schéma Entités-Associations (E/A)

Exemple filé de la base étudiant

On obtient (par synthèse, décomposition ou par transformation E/A) :

- $R_0(\texttt{NumEt}, \texttt{NomEt}, \texttt{Adresse})$, i.e., `Etudiant` !
- $R_1(\texttt{NumUE}, \texttt{Titre})$, i.e., `UE` !
- $R_2(\texttt{NumEt}, \texttt{NumUE}, \texttt{Note})$, i.e., `Inscrit` !

1. On définit les DFs *a priori*, à partir de la sémantique des données, on ne part pas d'une instance qui pourrait vérifier « par hasard » $r \models \texttt{Adresse} \rightarrow \texttt{NumEt}$!

Quand ne *pas* normaliser ?

La normalisation n'est *pas* une obligation on peut vouloir s'en passer :

- Pour retrouver « toutes » les données (originales), il faut calculer des jointures, qui peuvent être coûteuses :
  - elle sont généralement nombreuses car la décomposition est maximale,
  - leur calcul n'est pas toujours performant, en particulier si les index ne sont pas adaptés.
- La normalisation peut être difficile et donc coûteuse.
- On en a pas nécessairement besoin quand la base n'a pas une très grande durée de vie ou s'il n'y a jamais de modifications.

Les approches de conception

Méthode de conception avec E/A

1. On définit d'abord un Modèle Conceptuel de Données (MCD) dans le formalisme E/A
   - On identifie les entités et entités faibles qui composent la base
   - On identifie les associations entre les entités, en précisant
     - les cardinalités de l'association
     - les attributs de l'association
2. On génère le schéma logique SQL à partir du MCD
   - Pour chaque entité (faible) on crée une table avec une PK (composite avec FK pour les faibles)
   - Pour chaque association, selon les cardinalités
     - Cas *many-to-many* on génère une nouvelle table avec une PK et deux références
     - Cas *many-to-one* on pousse l'association du côté du *one* et on ajoute une référence (sans changer la PK)

Les approches de conception

Les schémas relationnels obtenus par traduction (mécanique) des modèles Entités/Associations sont toujours en FNBC.

Décomposition

Étant donnés un schéma de relation R et un ensemble F de DF, on va décomposer R en R_1, R_2, ..., R_n tels que

- Cette décomposition soit sans perte d'information : on peut retrouver tout instance r de R en combinant *par jointure naturelle* les instances r_i sur les R_i
- Cette décomposition soit sans perte de dépendances : après, jointure des r_i, on peut retrouver toutes les dépendances impliquées par F à partir de leurs projections F_i sur les R_i
- R_1, R_2, ..., R_n soient dans une forme normale maximale

Algorithme de décomposition et synthèse

- $R = \mathcal{U}$ la relation (universelle) à décomposer
- F un *ensemble minimal* de dépendances sur R

Algorithme de décomposition

Algorithme de décomposition en FNBC

Soit $S = \{R\}$. Tant qu'il existe dans $R_i \in S$ qui n'est pas en FNBC :

- On cherche une dépendance non triviale $X \rightarrow Y$ telle que $R_i(X, Y, Z)$ et X n'est pas une clé de R_i.
- On ajoute à Y l'ensemble Z' des attributs de Z fonctionnellement déterminés par X, produisant la dépendance $X \rightarrow Y \cup Z'$.
- On remplace R_i dans S par les deux relations
  - $R_1(X, Y \cup Z')$
  - $R_2(X, Z \setminus Z')$

Propriété : cet algorithme est sans perte d'information mais pas toujours sans perte de dépendances

Algorithme de synthèse

Algorithme de synthèse en 3FN/FNBC

- Générer une relation XY pour chaque DF $X \rightarrow Y$;
- On supprime les schémas de relation qui ne sont pas maximaux par inclusion.
- S'il y a perte de jointure, alors on rajoute une relation composée d'une clé de F.

Propriété : l'algorithme est sans perte d'information et donne un schéma en 3FN ou en FNBC quand c'est possible sans perte de dépendance.