Synthèse de bases de données

Titouan Christophe
Florentin Hennecker
Rodrigue Van Brande

10 juin 2015

Table des matières

1 Modèle entité-association

Un schéma conceptuel des données, pas forcément implémenté de cette façon. Selon le contexte, désigne la classe (modèle) ou l'objet (l'instance).

Remarque Le nom **Entité-Relation** est équivalent à **Entité-Relationel** ou encore **Entité-Association**

1.1 Entité

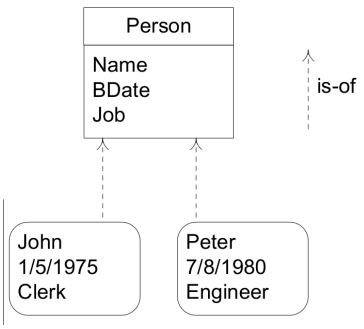


FIGURE 1 – Une entité

1.1.1 Attributs

Les entités ont des attributs, qui ont une cardinalité minimale et maximale

Attributs multivalués Attributs dont la cardinalité maximale est supérieure à 1. Exemple : `Person.degrees`

Attributs dérivés Attributs calculés à partir d'autres attributs, pas stockés dans la base de donnée. Exemple : à partir du champ `Person.birth`, on peu déduire `Person.age`

Attributs composites Attributs formés par l'aggrégation de plusieurs autres attributs. Exemple : `Person.Adress = {Street, City, State, Zipcode, Country}`

Clefs ou identificateurs Attributs ou ensemble d'attributs d'une entité dont les valeurs sont uniques (déterminent une et une seule entité). On les met en évidence dans un schéma entité-association en les soulignant, en ajoutant un symbole *clef* à côté, ou en les mettant dans une section à part dans la boîte de l'entité.

Clefs primaires Seule l'une des clefs peut être soulignée, et est l'identifiant d'un objet pour cette entité.

1.2 Association

FIGURE 2 – Une relation One-to-many

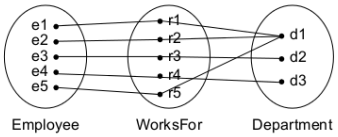


FIGURE 3 – Vue ensembliste de la relation à la Figure 2

— On doit spécifier la multiplicité minimale et maximale pour chaque pair de la relation
— La relation porte un nom (souvent lié au sens et à la sémantique de la relation)
— Chaque entité à un rôle dans la relation
— Une relation peut contenir des attributs
— L'arité (x-ary) d'une relation, ou son **degré**, indique le nombre d'entités participant à la relation
— **One-to-one**, **one-to-many** ou **many-to-many** pour les relations binaires

On peut noter qu'il n'y a pas de différence fondamentale entre un attribut et une relation, juste une question de point de vue. On préfèrera une relation si on souhaite mettre en évidence les liens entre plusieurs objets, et un attribut quand c'est une propriété comme un autre.

1.2.1 Relation n-aire

Une relation n-aire implique plus que 2 entités différentes.

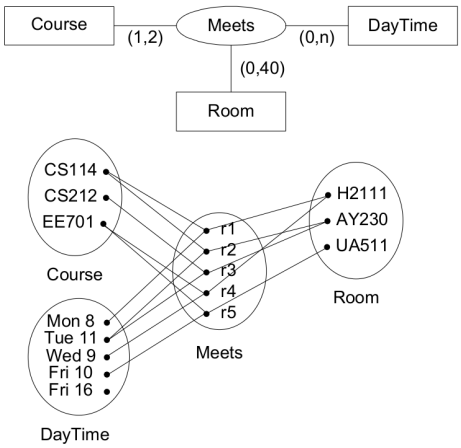


FIGURE 4 – Une relation ternaire et sa vue ensembliste

1.2.2 Transformation des relation n-aires en binaires

Les relations obtenues après transformation **par projection** contiennent moins d'information que la relation initiale. Par exemple, dans la Figure 5, on voit que la relation ternaire indique qu'un fournisseur délivre un produit pour un projet, tandis que les relations binaires indiquent uniquement qu'un fournisseur délivre un produit, qu'un projet utilise un produit, et qu'un projet utilise un fournisseur.

Projection On crée une relation pour chaque paire possible de la relation

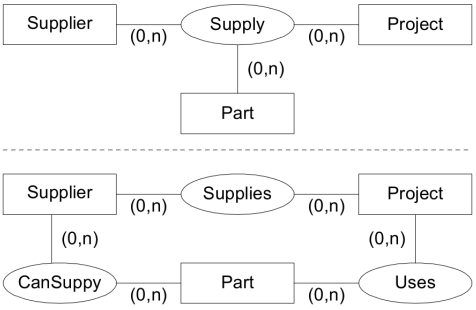


FIGURE 5 – Projection d'une relation ternaire

Insertion On transforme la relation en entité, et on la lie par des relations binaires à toutes les entités de la relation initiale.

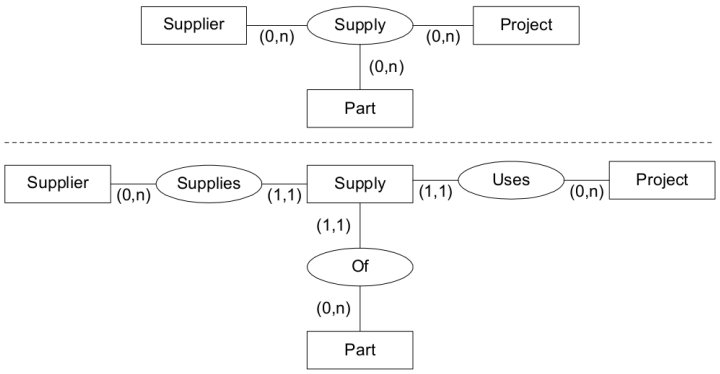


FIGURE 6 – Transformation par insertion

1.3 Entités faibles

Une entité faible n'a aucune clef ne dépendant que de ses propres attributs, et a donc une **clef partielle** qui la distingue des autres entités ayant la même clef étrangère.

Intérêt des entités faibles
- Suppression des attributs redondants
- Réduction des relation n-aires en relations binaires par la méthode d'insertion
- Héritage et polymorphisme (ajout des attributs des entités filles dans des entités faibles)

1.4 Relation génériques

Équivalent des design pattern pour les relations. Elle dénotent la nature d'une relation sur un schéma entité-association.

1.4.1 Classification

Utilisée lorsqu'on instancie un objet. Exemple : on a une liste de destinations au départ d'un aéroport (classe), et les vols chaque jour vers ces destinations (instance).

1.4.2 Généralisation (héritage)

Relation spéciale entre une ou plusieurs sous-entités et une superentité, plus abstraite.

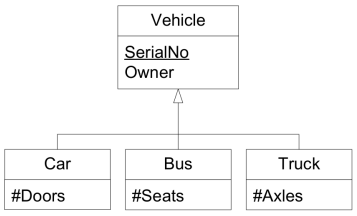


FIGURE 7 – Généralisation

Différent types de recouvrements
- **Partiel** ou **Total** : les entité filles ont-elles des attributs de la superentité ?
- **Exclusif** ou **Recouvrant (overlapping)** : les entités filles partagent-elles des attributs ?
- La spécialisation peut être déterminée par un prédicat (ex : `Person.age < 18 -> Child`)
- Les clefs sont héritées, et de nouvelles clefs peuvent être définies par les entités filles

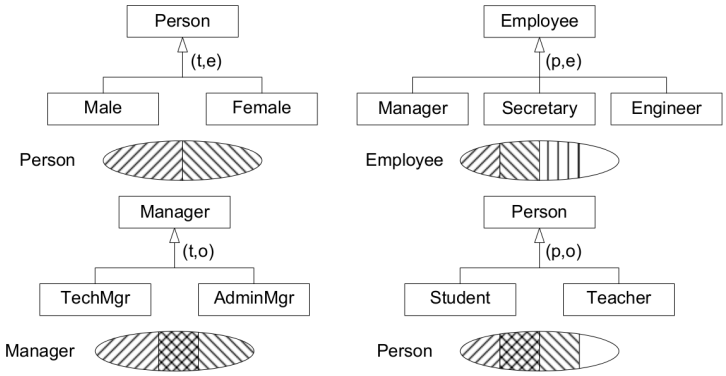


FIGURE 8 – Combinaisons de recouvrements partiels et totaux, exclusifs et recouvrants

Les contraintes de recouvrement influencent l'implémentation.

2 Modèle relationnel

Données structurées en tables bidimensionnels de valeurs simples. Les lignes sont aussi appelées tuples, et les colonnes attributs. Les relations sont des tables avec des restrictions :

— L'ordre des lignes n'a pas d'importance
— L'ordre des colonnes n'a pas d'importance
— La table a au moins une clef (pas de lignes en double exemplaire)

La relation peut être vue comme un prédicat, par exemple comme un ensemble de propriétés.

Les bases de données sont des collections de relations, mais ne pas oublier que les structures relationnelles sont plus riches que les tables, et que le modèle de données ne définit pas que les structures de données, mais aussi des opérations pour les manipuler.

2.1 Schéma

On parle du schéma pour décrire le modèle des données (le nom des colonnes).

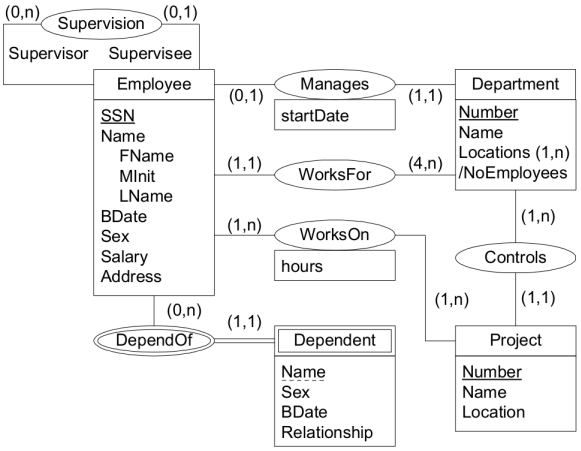


FIGURE 9 – Un modèle entité-association

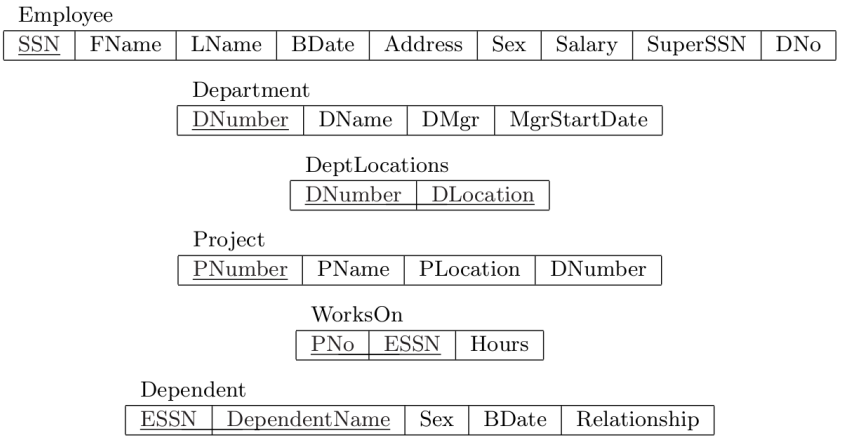


FIGURE 10 – Les tuples correspondant au schéma Figure 9

Lors de la traduction du modèle entité-association en modèle relationnel, on voit que toutes les relations ne sont pas matérialisées par une table (par exemple, un `Department` contient directement la clef de son manager et sa date d'entrée en service), et que certaines tables ne sont pas modélisées par une entité ou une relation, mais par un attribut multivalué (`Department.location`)

2.2 Relations

Le schéma auquel on adjoint les valeurs donnent une relation. Plus formellement, on a

$$R(A_1 : D_1,\ \ldots\ , A_n : D_n) \tag{1}$$

Où R est la relation, les D_i sont un ensemble de valeurs atomiques appelées le **domaine**, les attributs A_i sont distincts dans la relation. $A_i : D_i$ dénote la structure de la relation. Étant donné un tuple [1] $\{A_1 : d_1,\ \ldots\ , A_n : d_n\}$, on a la valeur relationelle

$$\rho \subseteq \{\{A_1 : d_1,\ \ldots\ , A_n : d_n\} \mid d_i \in D_i\} \tag{2}$$

Le domaine définit la comparabilité des valeurs : des attributs ne peuvent être comparés que s'ils sont sur le même domaine. On peut voir le domaine comme des types définis par l'utilisateur.

2.3 Clefs de relation

Une super-clef désigne un ou plusieurs attributs qui, ensemble, identifient un et un seul tuple.

Une clef est une super-clef minimale (si un seul des attributs est enlevé de la super-clef minimale, elle perd sa propriété d'identification unique). Généralement, une relation a plusieurs clefs, qu'on appelle aussi clefs candidates.

La clef primaire est une clef choisie parmi les clefs candidates comme étant la seule qu'on utilisera pour identifier un tuple.

2.4 Contraintes relationnelles

Les contraintes font parties du schéma, et spécifient des prescriptions ou assertions sur les données. Elles permettent de garantir l'intégrité des données.

1. un ensemble de paires attribut :valeur

2.4.1 Intégrité référentielle

Étant donné une contrainte relationelle sur R_1 et R_2, $\exists$ un attribut A_2 de R_2 tq. A_2 a le même domaine que la clef primaire de R_1 et toute valeur de A_2 dans un tuple de R_2 apparaît comme clef primaire dans le tuple de R_1. On appelle A_2 une clef étrangère. A_2 n'est pas forcément une clef de R_2

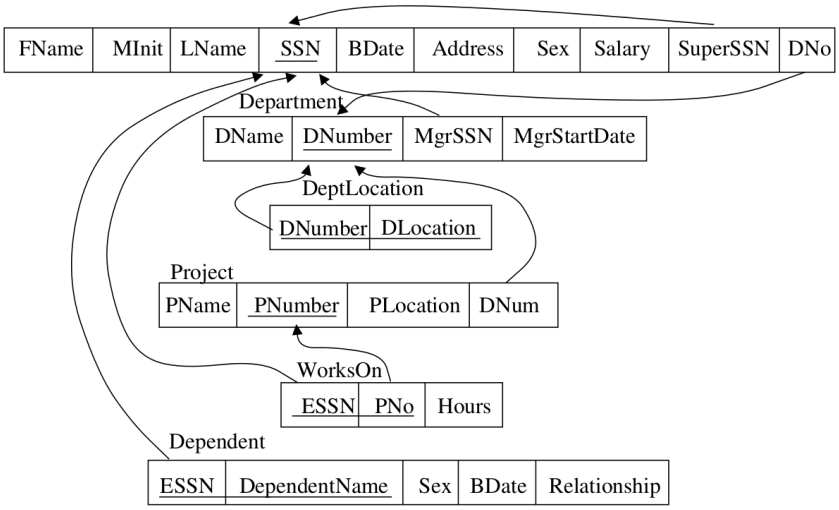


FIGURE 11 – Intégrité référentielle du schéma Figure 9

2.4.2 Contraintes transitionelles

On impose une contrainte sur la mise à jour de la valeur d'un attribut (exemple : une personne ne peut pas changer de sexe, un salaire ne peut qu'augmenter...)

2.4.3 Contraintes structurelles

Intégrité de la clef Toute relation a une clef primaire et éventuellement d'autres clefs

Intégrité de domaine Les valeurs des attributs doivent correspondre au domaine (type et opérations permises)

Intégrité de l'entité Les clefs primaires ne sont pas nulles

Intégrité de colonne Une contrainte spécifique imposée sur une colonne (ex : la date est supérieure à l'an 2000)

Intégrité de ligne Contrainte sur un tuple en particulier (ex : la date de retraite doit être postérieure à la date de premier emploi).

2.4.4 Suppression d'un tuple

Quand la suppression d'un tuple viole une contrainte d'intégrité référentielle, il faut exécuter une des actions suivantes :

- rejeter la suppression
- propager la suppression aux tuples qui référencent le tuple supprimé
- nullifier les valeurs d'attributs des tuples qui référencent le tuple supprimé, sauf si ces attributs font partie de la clé primaire

2.5 Implémentation dans les DBMS

Un DBMS a donc :

Un langage de définition des données (DDL) permettant de créer le schéma

Un langage de manipulation des données (DML) pour accéder aux valeurs. Par exemple, l'algèbre relationelle ou le calcul tuple. SQM redéfinit les opérations correspondantes et définit quelques extensions (les fonctions d'aggrégation par exemple).

Des opération d'altération permettant de modifier ou supprimer des valeurs et des schémas. Ces opérations doivent tenir compte des contraintes (en émettant une erreur si l'une d'elle est violée par exemple).

Des vues (éventuellement) décrivant des relations dérivées (relations inférées à partir d'autres). Elles peuvent être enregistrées sur le disque (snapshots), mais ça peut poser des problème de redondance.

2.6 Traduction depuis un schéma E-R

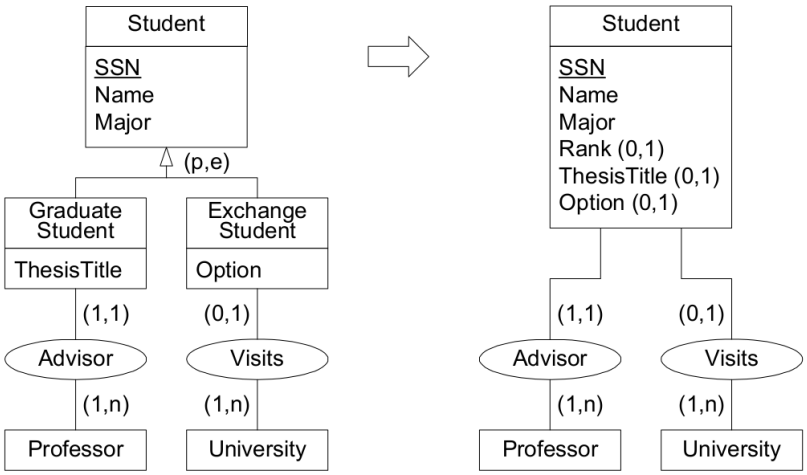
2.6.1 Suppression des généralisations

Tout dans la superentité Tous les champs de toutes les entités filles sont dans la superentité. Les attributs inexistants pour une entité fille sont nuls. On rajoute un champ pour déterminer le type d'enfant du tuple (dans l'exemple ci-dessous, `Rank`).

Il faut parfois également rajouter des contraintes (ci-dessous, par exemple : seulement les étudiants avec `Rank == GradStud` ont un advisor).

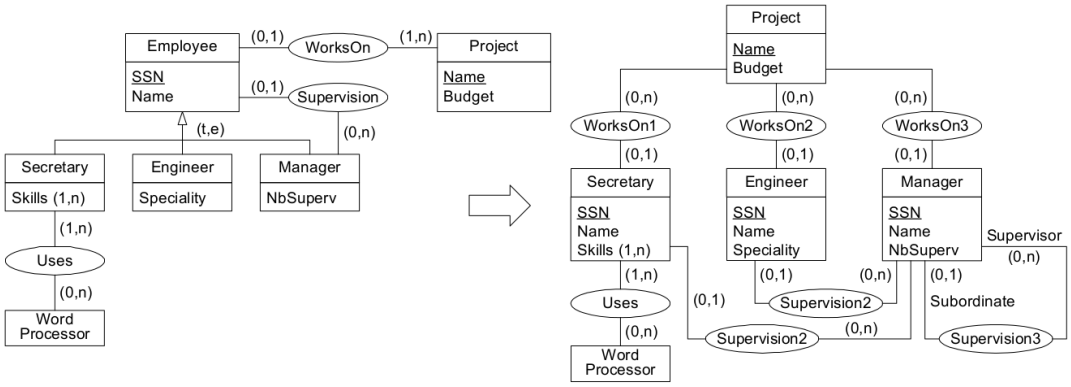
Avantage Facile et rapide à mettre en place, toujours utilisable.

Inconvénients Gaspillage d'espace (beaucoup d'attributs nuls), les opérations sur les entités filles doivent être exprimée sur la superentit (programme plus complexe).



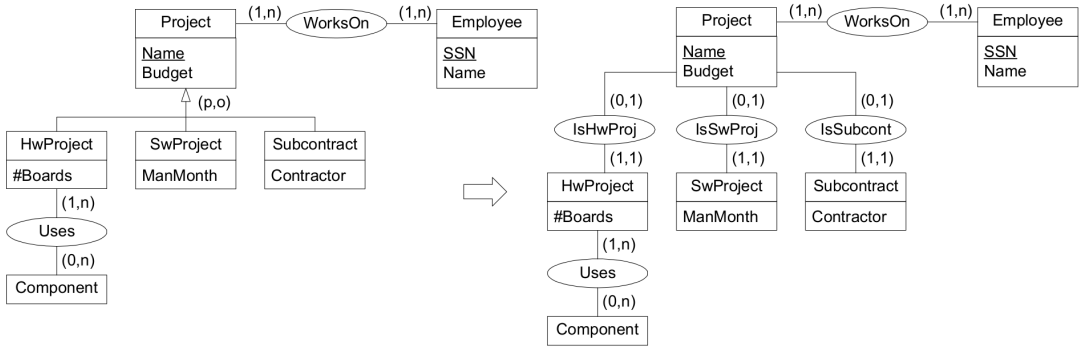
Garder les entités filles On ne crée que les entités feuilles dans l'arbre d'héritage, il n'existe pas de table pour la superentité. Chaque entité fille contient ses attributs propres et ceux de sa superentité. **Il faut garantir avec une contrainte que les clefs sont uniques parmis l'ensemble de toutes les entités filles.**

Inconvénients Beaucoup d'attribut redondants et de relations, rendant les applications plus complexes. Utilisable avec les relations totales-exclusives uniquement.



Transformation en relations Une table pour chaque entité. On crée une relation one-to-one entre une entité et sa superentité. Les filles sont des entités faibles.

Inconvénients Redondance (beaucoup de relations signifient la même chose), il faut des contraintes pour exprimer le type de généralisation (recouvrement). Les opération deviennent plus complexes (en O(n) de la profondeur de l'arbre d'héritage).



2.6.2 Règles de traduction

Entités Pour toutes les entités non-faibles E créer une relation R qui a tous les attributs de E. On identifie les clefs, et les attributs composites sont stocké champ par champ.

Entité faibles Création d'une relation R pour chaque entité faible E, contenant tous ses attributs simples, et les clefs des entités référencées comme clefs étrangères.

Relation one-to-one Choisir une des deux entité comme faible, et y insérer une clef étrangère qui est la clef primaire de l'autre.

Relation one-to-many L'entité du côté many est transformée en une relation R possèdant une clef étrangère qui référence la clef primaire de l'autre relation

Relation many-to-many On crée une relation pour chaque entité, et une relation avec une clef étrangère qui référencent les deux autres relations (on passe de 2 à 3 tables).

Attributs multivalués

Avec une relation supplémentaire On traduit une entité E contenant un attribut multivalué en relation R. On crée une relation contenant une clef étrangère vers la clef primaire de R et une des valeurs de l'attribut.

Department

DName	DNumber	DMgr	{DLocations}
Research	5	333445555	{Bellaire,Sugarland,Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

⇓ 1NF Normalization

Department

DName	DNumber	DMgr
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DeptLocations

DNumber	DLocation
5	Bellaire
5	Sugarland
5	Houston
4	Stafford
1	Houston

FIGURE 12 – Transformation des attributs multivalués en ajoutant une relation

Avec une seule relation On ajoute une ligne par valeur de l'attribut multivalué pour un même objet.

Department

DName	DNumber	DMgr	{DLocations}
Research	5	333445555	{Bellaire,Sugarland,Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

⇓ 1NF Normalization

Department

DName	DNumber	DLocation	DMgr
Research	5	Bellaire	333445555
Research	5	Sugarland	333445555
Research	5	Houston	333445555
Administration	4	Stafford	987654321
Headquarters	1	Houston	888665555

FIGURE 13 – Transformation des attributs multivalués en une seule relation

2.6.3 Perte d'information

La traduction amène à une perte de précision : une relation (dans le modèle relationnel) peut traduire plusieurs entités et relations du modèle E-R. Une bonne correspondance entre les deux modèle nécessite des bonnes contraintes d'intégrité et référentielles, ainsi qu'une documentation mettant le modèle E-R en relation avec la structure relationelle.

3 Algèbre relationnel

L'algèbre relationnel est un ensemble d'opérateurs qui agissent sur une ou plusieurs relations et qui produisent une relation en sortie.

3.1 Sélection (ou restriction)

On ne prend que les tuples qui satisfont une (ou plusieurs) condition(s).

$$\sigma_{condition}(R) = \{r \in R \,|\, condition(r)\}$$

Intuition : la sélection est une tranche horizontale d'une relation.

Exemple :

R

A	B	C
1	a	1
2	a	1
3	c	2
4	d	2

$\rightarrow \sigma_{C=2}(R) \rightarrow$

A	B	C
3	c	2
4	d	2

3.2 Projection

Garder seulement un ensemble d'attributs d'une relation.

$$\pi_{attributs}(R)$$

Intuition : la projection est une tranche verticale d'une relation.

Exemple :

R

A	B	C
1	a	1
2	a	1
3	c	2
4	d	2

$\rightarrow \pi_{B,C}(R) \rightarrow$

B	C
a	1
c	2
d	2

Suppression des doublons La projection fait en plus une suppression des doublons.

3.3 Combinaison d'opérations

Forme imbriquée de type $\pi_{FName,LName}(\sigma_{DNo<3 \vee DNo=5}(Employee))$

Forme séquentielle

$$Temp \leftarrow \sigma_{DNo=5}(Employee)$$
$$R(FirstName, LastName) \leftarrow \pi_{FName,LName}(Temp)$$

3.4 Opérations ensemblistes

Union $R \cup S$: tous les tuples dans R, dans S ou dans R et S (sans doublon)

Intersection $R \cap S$: tous les tuples dans R et dans S

Différence $R - S$: tous les tuples dans R mais pas dans S

Ces opérations ne se font que sur des relations dont les attributs sont de même types un à un.

Exemple :

Student

FName	LName
Susan	Yao
Ramesh	Shah
Barbara	Jones
Amy	Ford
Jimmy	Wang

Prof

FName	LName
John	Smith
Ricardo	Brown
Susan	Yao
Francis	Johnson
Ramesh	Shah

Student ∪ Prof

FName	LName
Susan	Yao
Ramesh	Shah
Barbara	Jones
Amy	Ford
Jimmy	Wang
John	Smith
Ricardo	Brown
Francis	Johnson

Student ∩ Prof

FName	LName
Susan	Yao
Ramesh	Shah

Student - Prof

FName	LName
Barbara	Jones
Amy	Ford
Jimmy	Wang

3.5 Renommage

Dans certains cas (voir produit cartésien), il faut renommer certains attributs d'une relation. On le fait de la manière suivante :

$$\alpha_{originalAttributeName:newAttributeName}(R)$$

Exemple :

R

A	B	C
1	a	1
2	a	1
3	c	2
4	d	2

$\rightarrow \alpha_{C,D}(R)$ ou $R(A, B, D) \leftarrow R \rightarrow$

A	B	D
1	a	1
2	a	1
3	c	2
4	d	2

3.6 Produit cartésien

Le produit cartésien $R \times S$ associe chaque tuple d'une relation avec tous les tuples d'une autre relation. Puisque tous les attributs de chaque relation se retrouvent dans le résultat, si certains attributs ont les mêmes noms, il faut les renommer.

Exemple :

R

A	B
A1	B1
A2	B2

S

C	D
C1	D1
C2	D2

R × S

A	B	C	D
A1	B1	C1	D1
A1	B1	C2	D2
A2	B2	C1	D1
A2	B2	C2	D2

3.7 Jointure

La jointure (aussi appelée jointure θ) n'est en fait qu'un produit cartésien suivi d'une sélection. La condition exprimée sur la jointure est effectivement la condition de la sélection.

$$R \bowtie_{condition} S = \{concat(r, s) \mid r \in R \ \wedge \ s \in S \ \wedge \ condition(r, s)\}$$

N'oublions pas que certains attributs de R ou de S pourraient devoir être renommés.

Les notations des paragraphes suivants proviennent de wikipedia :

Exemple :

R

A	B
a	b
c	b
d	e

S

C	D
b	c
e	a
b	d

R $\bowtie_{B=C}$ S

A	B	C	D
a	b	b	c
a	b	b	d
c	b	b	c
c	b	b	d
d	e	e	a

Jointure naturelle $R \bowtie S$: pour tous les attributs qui se trouvent en même temps dans R et dans S, on garde les tuples qui ont les mêmes valeurs pour ces attributs dans R et dans S.

Exemple :

R

A	B
a	b
c	b
d	e

S

C	D
b	c
e	a
b	d

R ∗ S

A	B	C
a	b	c
a	b	d
c	b	c
c	b	d
d	e	a

Equijoin $R \bowtie_{A=B} S$: Jointure θ avec une seule condition qui est une égalité.

L'equijoin tel que très floutement défini dans le cours :

Equijoin : $R *_{A=B} S$ quand la condition de join est une simple égalité

Left Outer Join $\supset\!\bowtie$

Right Outer Join $\bowtie\!\subset$

Full Outer Join $\supset\!\bowtie\!\subset$

3.8 Division

$$R(A, B) \div S(B)$$

Tous les tuples de $\pi_A(R)$ qui apparaissent dans R avec au moins chaque élément de S.

R

A	B
a1	b1
a1	b2
a1	b3
a1	b4
a2	b1
a2	b3
a3	b2
a3	b3
a3	b4
a4	b1
a4	b2
a4	b3

S

B
b1
b2
b3

T = R ÷ S

A
a1
a4

3.9 Outer Union

Si R et S sont partiellement compatibles, l'outer union garde tous les attributs et le annule s'ils ne sont pas applicables pour ce tuple.

4 Calcul relationnel

Cette section traite de la structure logique des Relational Query Languages. On utilise la logique du premier ordre.

4.1 Calcul relationnel tuple

La structure générale d'une requête en calcul tuple relationnel est :

$$\{t_1.A_1, t_2.A_2, ..., t_n.A_n \mid F(t_1, ...t_n, t_{n+1}, ...t_m)\}$$

avec :

- $t_1, t_2, ...t_m$ les variables tuple associées à F avec une relation via un prédicat de relation [2]
- A_i : les attributs de la relation associée à t_i
- F : la formule logique qui contient $t_1, ..., t_m$
- $t_1, ...; t_n$ les variables libres de F ("variables de query")
- $t_{n+1}, ..., t_m$ les variables quantifiées dans F

4.1.1 Quantificateur existentiel

Voici un join (le quantificateur existentiel est généralement utilisé pour les join) :

$$\{e.Lname, e.Address \mid Employee(e) \\ \wedge \exists d(Department(d) \wedge d.Manager =' KrazyBen' \wedge d.DNumber = e.DNo)\})$$

4.1.2 Quantificateur universel

Le quantificateur universel est très utile pour les divisions. Il doit toujours être associé avec l'implication. Pourquoi ? Prenons l'exemple de cours prérequis. On veut tous les étudiants qui ont passé les prérequis de Plasticine. Si on a cette formule : $\forall p(Prereq(p) \wedge \exists etudiant...$ et que Plasticine n'a aucun prérequis, la formule renvoie un ensemble vide alors que tous les étudiants ont passé les "aucun" prérequis.

$$\{e.Lname \mid Employee(e) \\ \wedge \forall d(Dependent(d) \rightarrow e.SSN = d.ESSN)\})$$

4.2 Calcul relationnel domaine

La structure générale d'une requête en calcul relationnel domaine est :

$$\{x_1, s_2, ..., x_n \mid F(x_1, ...x_n, x_{n+1}, ...x_m)\}$$

avec $R(A_i : x_i, ...A_j : x_j)$.

4.2.1 Quantificateur existentiel

Voici un exemple simple :

$$\{fn, ln \mid \exists fn, ln \\ (Employee(FName : fn, LName : ln) \\ \wedge\ fn =' Krazy' \\ \wedge\ ln =' Ben')\}$$

Plutôt que d'"assigner" tous les champs et d'écrire une condition dessus, si on n'a besoin d'un champ que pour une condition, on peut écrire ceci :

$$\{fn \mid \exists fn \\ (Employee(FName : fn, LName :' Ben') \\ \wedge\ fn =' Krazy')\}$$

Voici un exemple dans le cas d'un join (nom et addresse de tous les employés du département plasticine) :

$$\{fn, ln, a \mid \exists d \\ (Employee(FName : fn, LName : ln, Address : a, DNo : d) \\ \wedge\ (Department(DName :' Plasticine', DNumber : d)))\}$$

2. wat

Quantificateur universel Il fonctionne de la même manière qu'en calcul tuple relationnel.

5 SQL

Note : cette partie n'est que sommaire, puisqu'elle a été étudiée en détail dans le projet

La forme générale d'une requête SQL est `SELECT attributes FROM relations WHERE condition`.

L'opérateur `SELECT x1 FROM x2` est équivalent à $\pi_{x_1}(\sigma(x_2))$ en algèbre relationelle. L'opérateur `WHERE` est équivalent à une σ.

5.1 Aggrégation

SQL propose des fonctions d'aggrégation, optimisées algorithmiquement (mais pas nécessairement relationellement). On peut par exemple `SELECT SUM(salary) FROM employees;`. L'opérateur `GROUP BY` permet de grouper les éléments d'un résultat. En lui ajoutant une clause `HAVING`, on peut spécifier des options au regroupement.

5.2 Ordre opérationnel

Dans le cas d'une requête complète

```
1 SELECT ...
  FROM ...
  WHERE condition1
  GROUP BY attribut(s)
  HAVING condition2
```

1. Évaluation du `SELECT ... FROM ... WHERE ...` sans la partie `SELECT`
2. Partition du résultat dans les différentes classes pour chaque attribut apparaissant dans le `GROUP BY`
3. Application du `HAVING` à chaque classe aggrégée
4. Exécution du `SELECT`

6 Normalisation

6.1 Motivation

Une bonne conception de base de données peut être obtenue par
— l'intuition et le savoir-faire du développeur
— une préconception dans un modèle de données plus expressif (ex : E-R)
— par la théorie de normalisation.

Il faut prendre en compte à la fois le niveau logique (clarté de la sémantique et des données, facilité de recherche et altération, ...), et le niveau physique (espace de stockage, efficacité d'accès, ...).

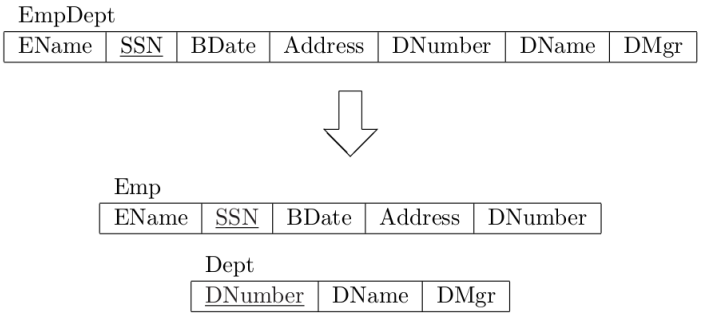


FIGURE 14 – Exemple de mauvaise conception et d'une meilleure normalisation

À la Figure 14, on modélise un employé d'un département. Soit on met tout dans une même relation, soit on en fait deux tables. Il y a plusieurs inconvénients à tout mettre dans la même table.

Anomalies d'insertion Dans l'exemple ci-dessus, on devrait vérifier, avant d'insérer un élément, que les données concernant le département sont cohérentes avec les autres employés du même département. Il n'est pas possible de créer un département sans employé, puisque le numéro d'employé (`SSN`) est une clef.

Anomalies de modification et de suppression Si on supprime le dernier employé d'un département, le département n'existe plus (peut poser problème si d'autres tables référencent un département). Si on change le manager d'un département (`DMgr`), il faut le modifier dans plusieurs lignes, ...

Ces anomalies peuvent mener à des résultats de jointure erronés par exemple.

6.2 Dépendances fonctionnelles

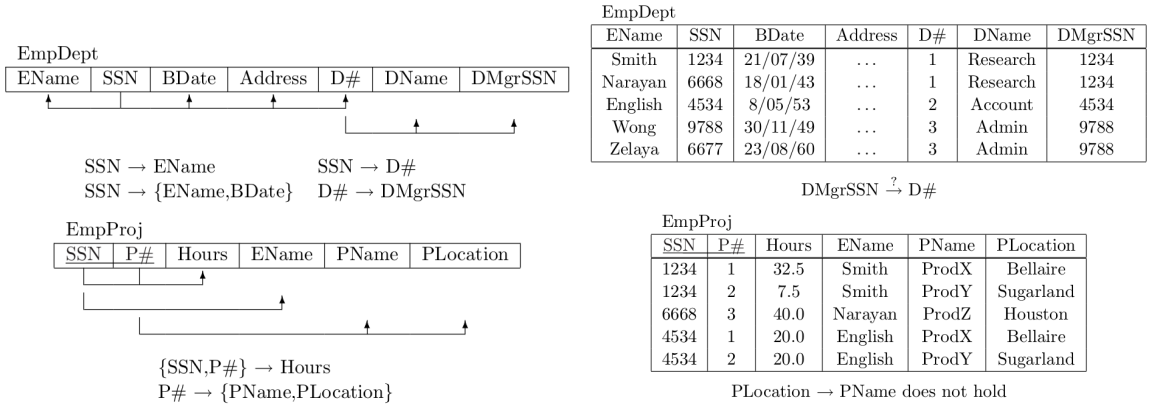
La dépendance fonctionnelle $X \to Y$ s'applique à la relation $R(A_1, \ldots, A_n)$, avec $X, Y \subseteq \{A_1, ..., A_n\}$, si

$$\forall t_1, t_2 \ : \ t_1[X] = t_2[X] \to t_1[Y] = t_2[Y]$$

ou encore si

$$\nexists t_1, t_2 \ : \ t_1[X] = t_2[X]$$

Il y a donc une dépendance fonctionnelle de X à Y si tous les tuples égaux sur X sont aussi égaux sur Y. On dit que **Y dépend de X**.



EmpDept						
EName	SSN	BDate	Address	D#	DName	DMgrSSN
Smith	1234	21/07/39	...	1	Research	1234
Narayan	6668	18/01/43	...	1	Research	1234
English	4534	8/05/53	...	2	Account	4534
Wong	9788	30/11/49	...	3	Admin	9788
Zelaya	6677	23/08/60	...	3	Admin	9788

EmpProj					
SSN	P#	Hours	EName	PName	PLocation
1234	1	32.5	Smith	ProdX	Bellaire
1234	2	7.5	Smith	ProdY	Sugarland
6668	3	40.0	Narayan	ProdZ	Houston
4534	1	20.0	English	ProdX	Bellaire
4534	2	20.0	English	ProdY	Sugarland

FIGURE 15 – Exemples de dépendances fonctionnelles

6.2.1 Inférence de nouvelles dépendances fonctionnelles

Seules les dépendances les plus évidentes sont explicitement spécifiées, mais on peut en déduire d'autres à partir de celles-ci par déduction formelle.

La closure F^+ d'un ensemble de dépendances F est l'ensemble des dépendances qu'on peut inférer de F.

$$X \to Y \in F^+ \Leftrightarrow F \models X \to Y$$

Règles d'inférence d'Armstrong Ces formules peuvent être prouvées à l'aide de la définition des dépendances.

Réflexivité $Y \subseteq X \models X \to Y$

Augmentation $X \to Y \models XZ \to YZ$ (et $XZ \to Y$)

Transitivité $(X \to Y) \wedge (Y \to Z) \models X \to Z$

Formules dérivées Ces formules peuvent être prouvées à partir des formules précédentes

Décomposition $X \to YZ \models (X \to Y) \wedge (X \to Z)$

Union $(X \to Y) \wedge (X \to Z) \models X \to YZ$

Pseudo transitivité $(X \to Y) \wedge (WY \to Z) \models WX \to Z$

6.2.2 Equivalence et couverture

Soit deux ensembles de dépendances fonctionnelles F et G.

Couverture F couvre G si $(\forall g \in G \ : \ F \models g)$ (càd $G^+ \subseteq F^+$)

Équivalence F et G sont équivalents si F couvre G et G couvre F (càd $F^+ = G^+$)

Ensemble minimal F est un ensemble de dépendances fonctionelles minimal si aucun de ses sous-ensemble n'est équivalent à lui même. On l'appelle aussi la couverture minimale.

6.2.3 Algorithme pour trouver la couverture minimale

1. $G := F$
2. On remplace chaque dépendance $X \to \{Y_1, \ \ldots \ , Y_n\}$ par n dépendances $X \to Y_1, \ \ldots \ , X \to Y_n$ dans G
3. Pour chaque dépendance $X \to A$ dans G, pour chaque attribut B qui est un élément de X, si $(G - \{X \to A\}) \cup \{(X - \{B\}) \to A\}$ est équivalent à G, on remplace $X \to A$ par $(X - \{B\}) \to A$
4. Enlever toutes les dépendances $x = X \to A$ telle que $G - \{x\}$ est équivalent à G

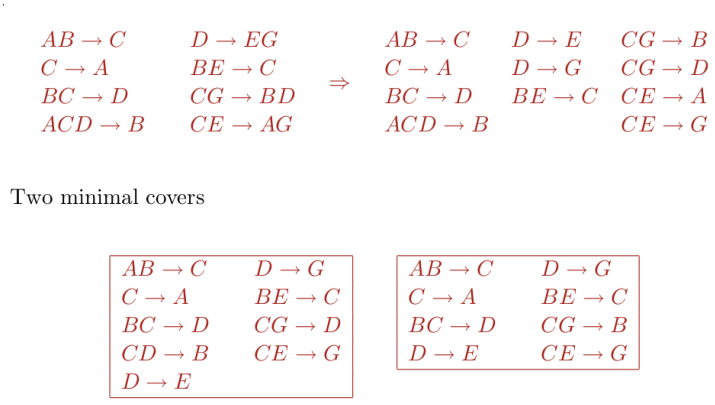


FIGURE 16 – Exemple de réduction à l'ensemble de contraintes minimal

6.3 Forme normale

Une condition (ou contrainte d'intégrité) qui certifie qu'un schéma relationel est dans une forme particulière. Plus on monte dans les formes normales, plus petites sont les relations, et nécessiteront donc plus de jointures.

6.3.1 Première forme normale

— Les relations sont définies dans le modèle relationnel
— Les attributs composites et multivalués, ainsi que les relations imbriquées sont interdites

6.3.2 Deuxième forme normale

Une dépendance entièrement fonctionelle est une dépendance fonctionelle $X \to Z$ telle que la suppression de n'importe quel attribut de X invalide la dépendance.

Un attribut premier est un attribut qui est membre d'une clef.

Définition Une relation R est en deuxième forme normale si tout attribut non-premier dépend entièrement d'une clef. On peut remarquer qu'une relation où toutes les clefs sont des attributs simples est automatiquement en 2FN.

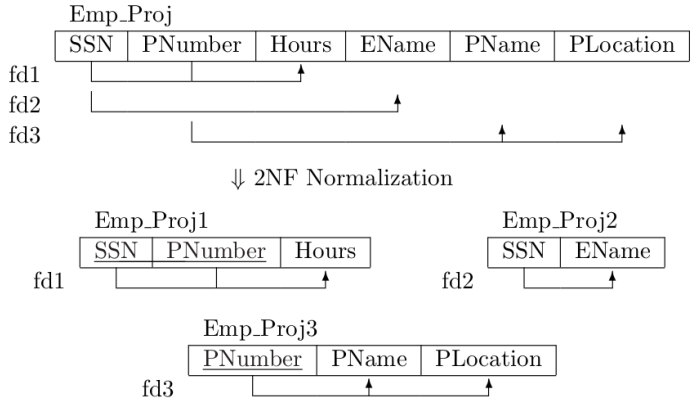


FIGURE 17 – Exemple de mise en deuxième forme normale

6.3.3 Troisième forme normale

Une dépendance transitive $X \to Z$ est décomposable en $(X \to Y) \wedge (Y \to Z)$ (où Z n'est pas un sous ensemble de la clef).

3 définitions pour qu'une relation R soit en 3FN
— R est en 2FN et tous les attributs non-premiers dépendent transitivement d'une super-clef
— Pour toutes les dépendances $X \to A \in F^+$, X est une clef ou A est un attribut premier
— Tous les attributs non-premier sont entièrement dépendants de chaque clef et ne sont pas transitivement dépendant de chaque clef

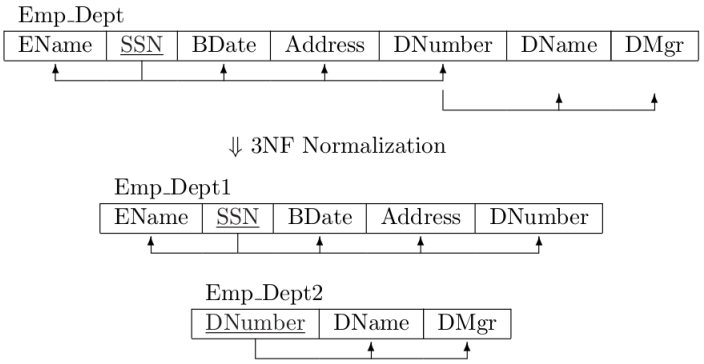


FIGURE 18 – Exemple de mise en troisième forme normale

6.3.4 Forme normale de Boyce-Codd (BCNF)

LA forme de Boyce-Codd est une troisième forme normale améliorée, où toutes les dépendances résultent des clefs. Une relation à 2 attributs est d'office en BCNF.

Intuition de la BCNF Les dépendances fonctionelles concernent la clef, entière et seulement la clef.

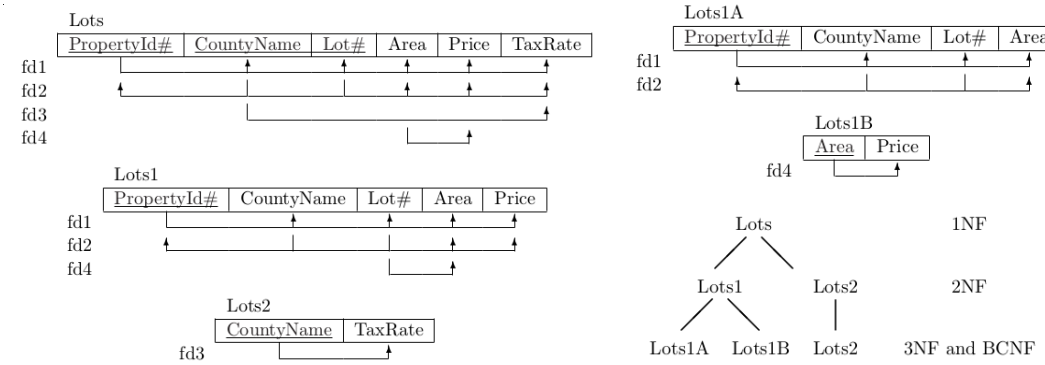


FIGURE 19 – Exemple de raffinement successif en formes normales

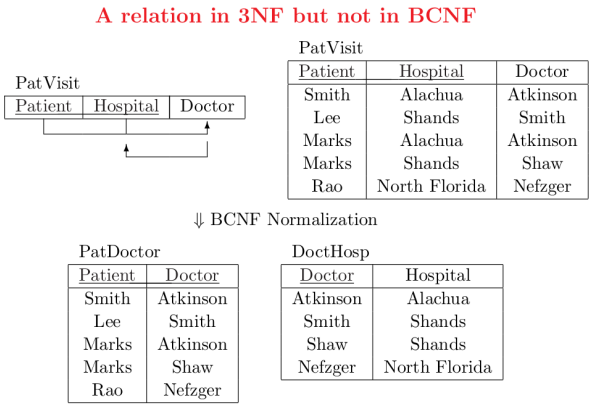


FIGURE 20 – Contre-Exemple de BCNF

6.3.5 Quatrième forme normale

Dépendance multivaluée Une dépendance $X \twoheadrightarrow Y$ est multivaluée si on peut définir un ensemble de valeurs admissibles pour Y pour X connu.

Une dépendance multivaluée triviale est le cas particulier $(X \subseteq Y) \vee (X \cup Y = R)$.

Définition Une relation est en 4NF si elle est en 3NF et que pour chacune de ses dépendances multivaluées non triviales $X \twoheadrightarrow Y$, X est une super-clef.

6.3.6 Cinquième forme normale

Une relation est en 5NF si elle est en 4NF et qu'il n'est plus possible de la décomposer sans perte d'informations.