

Bases de Données I

Université de Mons

Jef Wijsen

2024–2025

Préface

Merci de signaler toute erreur, de toute sorte (orthographe, grammaire, typographie, contenu...), à `jef.wijsen@umons.ac.be`. Des suggestions pour améliorer les explications seront également appréciées.

Je tiens à remercier un nombre d'étudiants pour leur lecture attentive et leurs remarques : Damien Galant, Cédric Pilatte, James Main, Alix Declerck.

Le site web <http://informatique.umons.ac.be/ssi/bdI.shtml> contient des anciens examens et des slides.

Table des matières

I	Le Modèle Relationnel	5
1	Preliminaires	6
2	La Structure du Modèle Relationnel	7
2.1	Relations	7
2.2	Bases de données	8
2.3	Clé primaire et clé étrangère	9
2.4	La contrainte UNIQUE	10
2.5	Le bouche-trou NULL	10
2.6	La perspective “Unnamed”	11
3	L’Algèbre Relationnelle	12
3.1	Syntaxe	12
3.2	Sémantique	13
3.3	Exemples	13
3.4	Inclusion et équivalence	14
3.5	Non-redondance des opérateurs	15
3.6	Fermeture transitive	16
3.7	La perspective “Unnamed”	16
3.7.1	Syntaxe	16
3.7.2	Sémantique	17
3.7.3	Exemple	17
4	Le Calcul Relationnel	18
4.1	Syntaxe	18
4.2	Sémantique	19
4.3	Indépendance du domaine	20
4.4	Des requêtes dites safe	22
4.5	Discussions approfondies	23
4.6	Le calcul relationnel TRC	24
4.6.1	TRC par l’exemple	24
4.6.2	Syntaxe	26
4.6.3	Sémantique	26
4.6.4	Du TRC au SQL	27
5	SQL	28
5.1	La base de données	28
5.2	Création de domaines et de tables	28

5.3	Retrouver des informations	29
5.4	Mises à jour	33
5.5	Intégration de SQL à des langages de programmation	34
5.6	Vues	35
5.6.1	Définition des Vues	35
5.6.2	Interrogation au travers de vues	35
5.6.3	Mise à jour au travers de vues	35
6	Les Dépendances Fonctionnelles	36
6.1	Introduction	36
6.2	Syntaxe et sémantique	36
6.3	Fermeture d'un ensemble d'attributs	37
6.4	Autres dépendances	38
6.4.1	Dépendances de jointure et dépendances multivaluées	38
6.4.2	Dépendances d'inclusion	40
7	Les Formes Normales	42
7.1	La redondance	42
7.2	BCNF	42
7.3	Décompositions	43
7.4	Décomposition en BCNF	44
7.5	3NF	45
7.6	Dépendances fonctionnelles singulières	46
7.7	Décomposition en 3NF	46
7.8	1NF, 2NF, 4NF, 5NF	48
7.8.1	1NF et 2NF	48
7.8.2	4NF	49
7.8.3	5NF	50
8	Le Modèle Entité-Association	52
8.1	Énoncé	52
8.2	Le diagramme entité-association	53
8.3	Traduction vers le modèle relationnel	53
II	Gestion des Transactions	55
9	Two-Phase Locking (2PL)	56
9.1	Cadre théorique	56
9.2	Les exécutions sérielles	57
9.3	Les exécutions sérialisables	57
9.4	L'écriture aveugle	58
9.5	Spécification du protocole 2PL	58
9.5.1	Le comportement d'une transaction	59
9.5.2	L'exécution globale	60
9.6	Implémentation du protocole 2PL	61
9.7	Le verrou mortel	62
9.8	Strict 2PL	63
10	Résistance aux Pannes et Reprise	65

10.1	Le buffer	65
10.2	Undo/Redo	66
10.3	Le journal	67
10.4	Procédure de Reprise	67
10.5	Exemple	67
10.6	Checkpointing	68
10.7	Undo/No-Redo et Redo/No-Undo	69

III Exercices 70

A Les Grandes Découvertes en Bases de Données 124

A.1	Introduction	124
A.2	Les BD Hiérarchiques	124
A.3	Les BD de Type Réseau	125
A.4	Les BD Relationnelles	127
A.5	Le Web, une BD ?	130
A.5.1	Un Manque de Structure	130
A.5.2	Traiter le Futur Web comme BD	130

Première partie

Le Modèle Relationnel

Chapitre 1

Préliminaires

L'ensemble vide est dénoté par $\emptyset$ ou $\{\}$.

Soient A et B deux ensembles. On écrit $A \subsetneq B$ si $A \subseteq B$ et $A \neq B$.

On dénote par $|A|$ le nombre d'éléments dans A .

Le *produit cartésien* de A et B , dénoté par $A \times B$, est l'ensemble

$$A \times B := \{(a, b) \mid a \in A, b \in B\}.$$

Une *relation binaire* sur A est un sous-ensemble de $A \times A$.

Soit R une relation binaire sur A . On dit que R est *transitive* si pour tout $a, b, c \in A$, si $(a, b) \in R$ et $(b, c) \in R$, alors $(a, c) \in R$. La *fermeture transitive* de R est la plus petite (par rapport à $\subseteq$) relation binaire qui contient R et qui est transitive.

Soit n un entier non-négatif. L'ensemble A^n est défini comme suit :

$$A^n := \{\langle a_1, \dots, a_n \rangle \mid a_1, \dots, a_n \in A\}.$$

Une *fonction totale* f de A vers B , dénotée $f : A \rightarrow B$, est un sous-ensemble de $A \times B$ tel que pour tout $a \in A$, il existe exactement un élément $b \in B$ tel que $(a, b) \in f$; cet unique élément b est alors dénoté par $f(a)$. On appelle A le *domaine* de f .

Soit $f : A \rightarrow B$ une fonction totale et $A' \subseteq A$. La *restriction* de f à l'ensemble A' , dénotée par $f[A']$, est la fonction totale $f' : A' \rightarrow B$ telle que pour chaque $a \in A'$, $f'(a) = f(a)$.

Un cas particulier est la restriction de f à l'ensemble vide : $f[\emptyset] = \{\}$.

Exemple 1. Soient $A = \{a, b, c\}$ et $B = \{1, 2, 3, 4\}$. Soit $f = \{(a, 1), (b, 2), (c, 2)\}$. Alors f est une fonction totale de A vers B .

Noter la différence entre $f(a) = b$ et $f[\{a\}] = \{(a, b)\}$. □

Chapitre 2

La Structure du Modèle Relationnel

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation).

E.F. Codd, 1970 [5, page 377]

Le *modèle relationnel* a été introduit en 1970 par E.F. Codd [5]. Un premier prototype de système de base de données relationnelle, appelé “System R”, a été développé à partir de 1974 [3]. C’était le début d’une technologie révolutionnaire qui est maintenant omniprésente : le SGBDR, Système de Gestion de Base de Données Relationnelle.

Les ouvrages suivants présentent, de façon rigoureuse, les fondements théoriques des bases de données relationnelles :

- “The Theory of Relational Databases” par D. Maier [11] ;
- “Principles of Database and Knowledge-Base Systems” par J.D. Ullman [12, 13] ;
- “Foundations of Databases” par S. Abiteboul, R. Hull et V. Vianu [1].

Les livres [1] et [11] sont gratuitement disponibles en ligne.

2.1 Relations

Supposons

- un ensemble infini **dom** dont les éléments sont appelés des *valeurs* (ou des *constantes*) ;
- un nombre d’ensembles non vides, appelés *domaines*. Chaque domaine est un sous-ensemble de **dom** ;
- un nombre infini d’*attributs* $A, B, C, A_1, B_1, C_1, \dots$; et
- une fonction *Dom* qui fait correspondre, à chaque attribut, un domaine.

Un *schéma-de-relation* (ou simplement *schéma* si aucune confusion n’est possible) est alors un ensemble fini $\mathcal{A}$ d’attributs.

Prénom	Genre
Jean	♂
Anne	♀

FIGURE 2.1 – Une relation présentée sous forme de table.

Un *tuple* sur ce schéma-de-relation est une fonction totale $t : \mathcal{A} \rightarrow \mathbf{dom}$ telle que pour tout attribut $A \in \mathcal{A}$, $t(A) \in \text{Dom}(A)$. Une *relation* sur ce schéma-de-relation est un ensemble fini de tuples sur ce schéma.

Exemple 2. Soit N l'ensemble des prénoms autorisés en Belgique. Soient **Prénom** et **Genre** deux attributs avec $\text{Dom}(\text{Prénom}) = N$ et $\text{Dom}(\text{Genre}) = \{\sigma, \varphi\}$. Alors $\{\text{Prénom}, \text{Genre}\}$ est un schéma-de-relation. Deux tuples sur ce schéma-de-relation sont $\{(\text{Prénom}, \text{Jean}), (\text{Genre}, \sigma)\}$ et $\{(\text{Prénom}, \text{Anne}), (\text{Genre}, \varphi)\}$. L'ensemble qui contient ces deux tuples est une relation. $\square$

Les relations seront souvent présentées sous forme de table. La table de la figure 2.1 présente la relation de l'exemple 2. Ensuite, il est habituel de parler de “la première rangée” pour dénoter le tuple $\{(\text{Prénom}, \text{Jean}), (\text{Genre}, \sigma)\}$, ou de “la première colonne”. Noter cependant que des relations et des schémas-de-relation sont des ensembles non-ordonnés.

Dans la théorie, on va souvent négliger les domaines, en supposant que pour chaque attribut A , $\text{Dom}(A) = \mathbf{dom}$.

Si $A_1, A_2, \dots, A_\ell$ sont des attributs, on écrira souvent $A_1 A_2 \dots A_\ell$ pour dénoter l'ensemble $\{A_1, A_2, \dots, A_\ell\}$. Si X et Y sont des ensembles d'attributs, on écrira souvent XY pour dénoter l'union $X \cup Y$.

Par exemple, si A et B sont des attributs, et X est un ensemble d'attributs, alors $X \cup \{A, B\}$ peut être abrégé comme XAB .

2.2 Bases de données

Supposons un nombre infini de *noms de relation* $R, S, T, R_1, S_1, T_1, \dots$, et une fonction *sorte* qui fait correspondre, à chaque nom de relation, un schéma-de-relation. La notation $R[A_1, \dots, A_n]$ est utilisée pour dénoter que R est un nom de relation avec $\text{sorte}(R) = \{A_1, \dots, A_n\}$.

Un *schéma-de-base-de-données* (ou simplement *schéma* si aucune confusion n'est possible) est alors un ensemble fini $\mathcal{R}$ de noms de relation. Une *base de données* (ou *instance de base de données*) sur ce schéma-de-base-de-données est une fonction totale avec domaine $\mathcal{R}$ qui fait correspondre, à chaque nom de relation R dans $\mathcal{R}$, une relation sur $\text{sorte}(R)$. Le symbole $\mathcal{I}$ sera souvent utilisé pour dénoter une base de données ; pour $R \in \mathcal{R}$, on dénote par $R^{\mathcal{I}}$ la relation qui correspond à R .

Il est habituel de présenter une base de données comme un ensemble de tables, comme illustré dans la figure 2.2. Le schéma de cette base de données est $\{\text{VINS}, \text{ABUS}\}$ avec $\text{sorte}(\text{VINS}) = \{\text{Cru}, \text{Millésime}, \text{Qualité}\}$ et $\text{sorte}(\text{ABUS}) = \{\text{Buveur}, \text{Cru}, \text{Millésime}\}$.

VINS	Cru	Millésime	Qualité	ABUS	Buveur	Cru	Millésime
	Volnay	1983	A		Jean	Volnay	1983
	Volnay	1979	B		Jean	Volnay	1979
	Chablis	1983	A		Pierre	Volnay	1979
	Julienas	1986	C		Pierre	Julienas	1986

FIGURE 2.2 – Une base de données présentée sous forme de tables. Exemple issu de [8].

2.3 Clé primaire et clé étrangère

On fait correspondre, à chaque nom de relation R , une *clé primaire*, en utilisant la syntaxe suivante :

$$R \text{ PRIMARY KEY}(A_1, A_2, \dots, A_k), \quad (2.1)$$

avec $A_1, A_2, \dots, A_k$ des attributs distincts appartenant à $\text{sorte}(R)$. Soit $\mathcal{I}$ une base de données dont le schéma contient R . On dit que $\mathcal{I}$ satisfait (ou respecte) la clé primaire de R si pour tout $t_1, t_2 \in R^{\mathcal{I}}$, si $t_1[A_1 A_2 \dots A_k] = t_2[A_1 A_2 \dots A_k]$, alors $t_1 = t_2$. C'est-à-dire, la base de données $\mathcal{I}$ respecte la clé primaire si pour chaque tuple $t \in R^{\mathcal{I}}$, il n'existe pas d'autre tuple dans $R^{\mathcal{I}}$ avec la même valeur pour (les attributs de) la clé primaire; chaque tuple $t \in R^{\mathcal{I}}$ est donc identifié de façon unique par sa valeur pour la clé primaire. Il est habituel de souligner les attributs qui constituent la clé primaire.

La clé primaire de R peut être utilisée ailleurs pour faire référence aux tuples d'une relation $R^{\mathcal{I}}$. La syntaxe est comme suit :

$$S \text{ FOREIGN KEY}(B_1, B_2, \dots, B_k) \text{ REFERENCES } R, \quad (2.2)$$

avec S un nom de relation et $B_1, B_2, \dots, B_k$ des attributs appartenant à $\text{sorte}(S)$. Soit $\mathcal{I}$ une base de données dont le schéma contient R et S . On dit que $\mathcal{I}$ satisfait (ou respecte) la clé étrangère (2.2) si pour tout tuple $s \in S^{\mathcal{I}}$, il existe un tuple (unique) $t \in R^{\mathcal{I}}$ tel que pour chaque $i \in \{1, \dots, k\}$, $s(B_i) = t(A_i)$, où les attributs $A_1, \dots, A_k$ constituent la clé primaire (2.1). C'est-à-dire, chaque tuple de $S^{\mathcal{I}}$ fait référence à un tuple unique de $R^{\mathcal{I}}$ en copiant sa valeur pour la clé primaire. Noter que la longueur k de la clé étrangère est identique à la longueur de la clé primaire de R . Il est cependant possible que $B_i \neq A_i$ (pour un ou plusieurs $i \in \{1, \dots, k\}$). Il est aussi possible que $S = R$.

Les clés primaires et étrangères constituent un mécanisme élégant pour relier des tuples de différentes relations, tout en évitant des références "artificielles" telles que les pointeurs ou les auto-incréments. Les clés primaires et étrangères sont des *contraintes* que l'on ajoute à un schéma-de-bases-de-données pour restreindre les bases de données qui sont admissibles.

Exemple 3. Pour la base de données de la figure 2.2, on peut raisonnablement imposer les clés primaires et la clé étrangère suivantes :

```
VINS PRIMARY KEY(Cru, Millésime)
ABUS PRIMARY KEY(Buveur, Cru, Millésime)
ABUS FOREIGN KEY(Cru, Millésime) REFERENCES VINS
```

La clé primaire de VINS exprime qu'aucun cru ne peut avoir deux qualités différentes pour une même année. $\square$

VOITURES	Plaque	Châssis	Type	Année
	CGD.123	13459ABC	Renault 19	1992
	SAP.346	CBA54321	Peugeot 404	1994

ACCESSOIRES	Plaque	Accessoire
	CGD.123	radio
	CGD.123	système antivol

FIGURE 2.3 – Base de données. Chaque voiture a une plaque et un numéro de châssis qui lui sont propres.

Exemple 4. Cet exemple artificiel montre l'importance de l'ordre des attributs dans la spécification des clés primaires et étrangères.

```

R[A, B]
R PRIMARY KEY(A, B)
R FOREIGN KEY(B, A) REFERENCES R

```

Une base de données $\mathcal{I}$ (dont le schéma contient R) respecte ces contraintes si et seulement si pour tout $t_1 \in R^{\mathcal{I}}$, il existe $t_2 \in R^{\mathcal{I}}$ tel que $t_2(B) = t_1(A)$ et $t_2(A) = t_1(B)$. Noter que la clé primaire dans cet exemple sera toujours respectée, car une relation est un ensemble (sans doublons). $\square$

2.4 La contrainte UNIQUE

La syntaxe de la contrainte UNIQUE est comme suit :

```
R UNIQUE(A1, A2, ..., Ak),
```

avec R un nom de relation et $A_1, A_2, \dots, A_k$ des attributs distincts appartenant à $\text{sorte}(R)$. Cette contrainte est satisfaite par une bases de données $\mathcal{I}$ (dont le schéma contient R) si pour tout $t_1, t_2 \in R^{\mathcal{I}}$, si $t_1[A_1 A_2 \dots A_k] = t_2[A_1 A_2 \dots A_k]$, alors $t_1 = t_2$.

Exemple 5. La base de données de la figure 2.3 enregistre les accessoires présents dans différentes voitures. La contrainte UNIQUE exprime que chaque voiture a un numéro de châssis qui lui est unique.

```

VOITURES [Plaque, Châssis, Type, Année]
VOITURES PRIMARY KEY(Plaque)
VOITURES UNIQUE(Châssis)
ACCESSOIRES [Plaque, Accessoire]
ACCESSOIRES PRIMARY KEY(Plaque, Accessoire)
ACCESSOIRES FOREIGN KEY(Plaque) REFERENCES VOITURES

```

$\square$

2.5 Le bouche-trou NULL

Une valeur dans un tuple peut être absente pour plusieurs raisons : la valeur est inexistante, la valeur est inconnue... Dans ces situations, les systèmes de base de données permettent généralement de remplacer la valeur absente par le symbole NULL. Dans la base de données de la

CITOYENS	NN	Prénom	Nom	NPermisConduire
	94.04.01-001.31	Alex	Astaire	123456789
	15.09.11-001.47	Bart	Bretelle	NULL
	91.06.22-001.31	Cloé	Camp	NULL

FIGURE 2.4 – Base de données avec des NULLs.

figure 2.4, NN dénote le numéro national, et NPermisConduire le numéro de permis de conduire. Le numéro national de Bart Bretelle indique qu’il est né à la date du 11 septembre 2015 ; il est trop jeune pour avoir un permis de conduire. La situation est moins claire pour Cloé Camp : soit elle n’a pas de permis de conduire, soit elle a un permis de conduire dont le numéro est manquant dans la base de données actuelle.

Le bouche-trou NULL n’étant pas une valeur (i.e., $\text{NULL} \notin \mathbf{dom}$), il est nécessaire d’étendre les définitions des sections 2.1–2.4 pour tenir compte des NULLs. Par exemple, est-ce-que la relation VOITURES de la figure 2.3 pourrait contenir deux tuples distincts t_1, t_2 tels que $t_1(\text{Châssis}) = \text{NULL}$ et $t_2(\text{Châssis}) = \text{NULL}$, tout en respectant la contrainte VOITURES UNIQUE(Châssis) ?

Cependant, cette extension avec NULL est hors de portée du présent syllabus.

2.6 La perspective “Unnamed”

Le formalisme expliqué précédemment utilise des attributs pour dénoter une valeur dans un tuple ; on parle de la perspective “Named”, dans le sens où les attributs portent un nom.

Dans la perspective “Unnamed”, les attributs sont remplacés par des indices $1, 2, 3, \dots$.

Soit n un entier non-négatif. Une *relation d’arité n* est un sous-ensemble fini de $\mathbf{dom}^n$.

On définit la fonction *arité* qui fait correspondre, à chaque nom de relation R , un entier non-négatif tel que $\text{arité}(R) = |\text{sorte}(R)|$.

Soit $\mathcal{R}$ un schéma-de-base-de-données. Une *base de données* sur $\mathcal{R}$ est une fonction totale avec domaine $\mathcal{R}$ qui fait correspondre, à chaque nom de relation R dans $\mathcal{R}$, une relation d’arité $\text{arité}(R)$.

Les définitions de clé primaire, clé étrangère et la contrainte UNIQUE peuvent facilement être adaptées à la perspective “Unnamed”.

Dès que l’on ajoute un ordre aux attributs dans la perspective “Named”, une correspondance évidente émerge entre les perspectives “Named” et “Unnamed”. Cet ordre est généralement l’ordre dans lequel on liste les attributs dans la notation $R[A_1, \dots, A_n]$. Alors, un tuple “Named” $\{(A_1, a_1), \dots, (A_n, a_n)\}$ correspond, un-à-un, à sa version “Unnamed” $\langle a_1, \dots, a_n \rangle$.

Chapitre 3

L'Algèbre Relationnelle

Soit $\mathcal{R}$ un schéma-de-base-de-données (i.e., un ensemble fini de noms de relation). On peut concevoir une *requête* comme un programme informatique qui prend en entrée une base de données $\mathcal{I}$ sur $\mathcal{R}$ et qui donne en sortie une relation. Le “langage de programmation” introduit dans ce chapitre est l'*algèbre relationnelle*. Ce langage est moins puissant que les langages Java ou Python, mais est beaucoup plus simple et se prête mieux aux analyses théoriques.

Dans un premier temps, on introduira l'algèbre relationnelle dans la perspective “Named”. Cette algèbre est dénotée par le sigle SPJRUD, qui contient la première lettre de chaque opérateur de l'algèbre.

3.1 Syntaxe

Soit $\mathcal{R}$ un schéma-de-bases-de-données.

Noms de relation Chaque nom de relation R dans $\mathcal{R}$ est une expression algébrique (atomique).

Sélection “attribut égal constante” Si E est une expression algébrique, $A \in \text{sorte}(E)$, et $a \in \mathbf{dom}$, alors $\sigma_{A=a}(E)$ est une expression algébrique avec $\text{sorte}(\sigma_{A=a}(E)) = \text{sorte}(E)$.

Sélection “attribut égal attribut” Si E est une expression algébrique et $A, B \in \text{sorte}(E)$, alors $\sigma_{A=B}(E)$ est une expression algébrique avec $\text{sorte}(\sigma_{A=B}(E)) = \text{sorte}(E)$.

Projection Si E est une expression algébrique et $X \subseteq \text{sorte}(E)$, alors $\pi_X(E)$ est une expression algébrique avec $\text{sorte}(\pi_X(E)) = X$.

Jointure Si E_1 et E_2 sont des expressions algébriques, alors $E_1 \bowtie E_2$ est une expression algébrique avec $\text{sorte}(E_1 \bowtie E_2) = \text{sorte}(E_1) \cup \text{sorte}(E_2)$.

Renommage Si E est une expression algébrique, $A \in \text{sorte}(E)$, et B est un attribut tel que $B \notin \text{sorte}(E)$, alors $\rho_{A \mapsto B}(E)$ est une expression algébrique avec $\text{sorte}(\rho_{A \mapsto B}(E)) = (\text{sorte}(E) \setminus \{A\}) \cup \{B\}$.

Union Si E_1 et E_2 sont des expressions algébriques avec $\text{sorte}(E_1) = \text{sorte}(E_2)$, alors $E_1 \cup E_2$ est une expression algébrique avec $\text{sorte}(E_1 \cup E_2) = \text{sorte}(E_1)$. Noter que le symbole $\cup$ est utilisé ici pour marquer la différence avec l'union ensembliste $\cup$. Si aucune confusion n'est possible, on peut écrire $E_1 \cup E_2$ au lieu de $E_1 \cup E_2$.

Différence Si E_1 et E_2 sont des expressions algébriques avec $\text{sorte}(E_1) = \text{sorte}(E_2)$, alors $E_1 - E_2$ est une expression algébrique avec $\text{sorte}(E_1 - E_2) = \text{sorte}(E_1)$.

Des parenthèses sont utilisées pour préciser l'ordre d'évaluation. En absence de parenthèses, la jointure est prioritaire sur l'union et la différence. Par exemple, l'expression $\pi_A(R) \bowtie S \cup T$ doit être évaluée comme $(\pi_A(R) \bowtie S) \cup T$.

3.2 Sémantique

Soit $\mathcal{I}$ une base de données sur $\mathcal{R}$.

Si E est une expression algébrique, on dénote par $\llbracket E \rrbracket^{\mathcal{I}}$ la relation qui est le résultat de l'évaluation de E sur $\mathcal{I}$. La définition de $\llbracket E \rrbracket^{\mathcal{I}}$ est comme suit :

- Si R est un nom de relation, alors $\llbracket R \rrbracket^{\mathcal{I}} := R^{\mathcal{I}}$. Donc, le résultat de l'expression R est la relation dans la base de données qui correspond à R . Noter que pour deux bases de données différentes, $\mathcal{I}$ et $\mathcal{J}$, il est possible que $\llbracket R \rrbracket^{\mathcal{I}} \neq \llbracket R \rrbracket^{\mathcal{J}}$.
- $\llbracket \sigma_{A=a}(E) \rrbracket^{\mathcal{I}} := \{t \in \llbracket E \rrbracket^{\mathcal{I}} \mid t(A) = a\}$.
- $\llbracket \sigma_{A=B}(E) \rrbracket^{\mathcal{I}} := \{t \in \llbracket E \rrbracket^{\mathcal{I}} \mid t(A) = t(B)\}$.
- $\llbracket \pi_X(E) \rrbracket^{\mathcal{I}} := \{t[X] \mid t \in \llbracket E \rrbracket^{\mathcal{I}}\}$.
- Soit $X = \text{sorte}(E_1)$ et $Y = \text{sorte}(E_2)$.
Alors $\llbracket E_1 \bowtie E_2 \rrbracket^{\mathcal{I}} := \{t \mid t[X] \in \llbracket E_1 \rrbracket^{\mathcal{I}} \text{ et } t[Y] \in \llbracket E_2 \rrbracket^{\mathcal{I}}\}$.
- Soit $X = \text{sorte}(E) \setminus \{A\}$.
Alors $\llbracket \rho_{A \mapsto B}(E) \rrbracket^{\mathcal{I}} := \{t \mid \text{il existe } s \in \llbracket E \rrbracket^{\mathcal{I}} \text{ tel que } s[X] = t[X] \text{ et } s[A] = t[B]\}$.
- $\llbracket E_1 \cup E_2 \rrbracket^{\mathcal{I}} := \llbracket E_1 \rrbracket^{\mathcal{I}} \cup \llbracket E_2 \rrbracket^{\mathcal{I}}$, avec $\cup$ l'union ensembliste.
- $\llbracket E_1 - E_2 \rrbracket^{\mathcal{I}} := \llbracket E_1 \rrbracket^{\mathcal{I}} \setminus \llbracket E_2 \rrbracket^{\mathcal{I}}$, avec $\setminus$ la différence ensembliste.

Une expression algébrique E est aussi appelée une *requête*, et $\llbracket E \rrbracket^{\mathcal{I}}$ est la *réponse* à la requête E sur la base de données $\mathcal{I}$.

3.3 Exemples

Les buveurs exigeants Disons qu'un buveur est exigeant s'il ne boit que des vins de qualité A. Pour la base de données de la figure 2.2, on souhaite encoder la requête "*Quels buveurs sont exigeants ?*". Soit

$$E_1 := \pi_{\{\text{Buveur}, \text{Qualité}\}}(\text{ABUS} \bowtie \text{VINS}).$$

Un tuple $\{(\text{Buveur}, b), (\text{Qualité}, q)\}$ dans l'évaluation de E_1 signifie que b est un buveur qui a déjà bu un vin de qualité q . L'expression suivante renvoie alors les buveurs qui ont déjà bu un vin qui n'est pas de qualité A :

$$E_2 := \pi_{\{\text{Buveur}\}}(E_1 - \sigma_{\text{Qualité}=\text{"A"}}(E_1)).$$

Si on substitue dans cette expression la définition de E_1 , on obtient :

$$E_2 = \pi_{\{\text{Buveur}\}}(\pi_{\{\text{Buveur}, \text{Qualité}\}}(\text{ABUS} \bowtie \text{VINS}) - \sigma_{\text{Qualité}=\text{"A"}}(\pi_{\{\text{Buveur}, \text{Qualité}\}}(\text{ABUS} \bowtie \text{VINS}))).$$

Cependant, cette substitution nuit à la lisibilité.

L'expression demandée E_3 doit renvoyer tout buveur qui n'est pas dans l'évaluation de E_2 . Donc,

$$E_3 := \pi_{\text{Buveur}}(\text{ABUS}) - E_2.$$

L'expression E_2 peut être simplifiée si on introduit l'abréviation suivante. Si $A \in \text{sorte}(R)$, alors $\sigma_{A \neq "a"}(R)$ est une abréviation pour $R - \sigma_{A="a"}(R)$. Avec cette abréviation, E_2 peut s'écrire comme

$$E_2 := \pi_{\{\text{Buveur}\}}(\sigma_{\text{Qualité} \neq "A"}(E_1)).$$

La table sans colonnes La définition de $\pi_X(E)$ permet $X = \{\}$. Quelle requête est encodée par $E_4 := \pi_{\{\}}(E_3)$?

S'il existe au moins un buveur exigeant, alors l'évaluation de E_4 renvoie la relation $\{\{\}\}$, i.e., la relation qui contient le tuple vide. S'il n'existe pas de buveur exigeant, alors l'évaluation de E_4 renvoie la relation vide $\{\}$. Il est habituel d'interpréter $\{\{\}\}$ et $\{\}$ respectivement comme **true** et **false**. L'expression E_4 encode alors la requête booléenne "*Existe-t-il des buveurs exigeants ?*".

Les buveurs ayant bu plusieurs crus Pour la base de données de la figure 2.2, on souhaite encoder la requête "*Quels buveurs ont déjà bu deux ou plusieurs crus différents ?*". Soit

$$E_1 := \pi_{\{\text{Buveur}, \text{Cru}\}}(\text{ABUS}) \bowtie \rho_{\text{Cru} \rightarrow \text{Vin}}(\pi_{\{\text{Buveur}, \text{Cru}\}}(\text{ABUS})).$$

La requête finale E_2 est comme suit :

$$E_2 := \pi_{\{\text{Buveur}\}}(E_1 - \sigma_{\text{Cru}=\text{Vin}}(E_1)).$$

3.4 Inclusion et équivalence

Dans les définitions suivantes, un schéma-de-bases-de-données $\mathcal{R}$ est sous-entendu.

Soient E_1 et E_2 deux expressions algébriques. On dira que l'expression E_1 est incluse dans E_2 , dénoté par $E_1 \sqsubseteq E_2$, si pour toute base de données $\mathcal{I}$, $\llbracket E_1 \rrbracket^{\mathcal{I}} \subseteq \llbracket E_2 \rrbracket^{\mathcal{I}}$.

On dira que les expressions E_1 et E_2 sont équivalentes, dénoté par $E_1 \equiv E_2$, si $E_1 \sqsubseteq E_2$ et $E_2 \sqsubseteq E_1$.

Exemple 6. Soit R et S deux noms de relation avec $\text{sorte}(R) = \text{sorte}(S)$. Soit $X \subseteq \text{sorte}(R)$. On peut facilement prouver que $\pi_X(R \cup S) \equiv \pi_X(R) \cup \pi_X(S)$. $\square$

Il a été prouvé [1, Corollary 6.3.2] que l'équivalence des expressions algébriques est *indécidable*. C'est-à-dire, il n'existe pas d'algorithme qui prend en entrée deux expressions algébriques, et détermine si, oui ou non, ces deux expressions sont équivalentes.

Proposition 1 (Trakhtenbrot's theorem). *L'équivalence des expressions algébriques en SPJRUD est indécidable.*

R	$\left \begin{array}{cc} A & B \\ \hline a & 1 \end{array} \right.$	R	$\left \begin{array}{cc} A & B \\ \hline a & 1 \\ a & 2 \end{array} \right.$
-----	--	-----	---

FIGURE 3.1 – Bases de données $\mathcal{I}$ (à gauche) et $\mathcal{J}$ (à droite).

3.5 Non-redondance des opérateurs

Un exercice intéressant est de démontrer qu’aucun des six opérateurs de l’algèbre SPJRU n’est redondant. Démontrons ici que la différence est un opérateur non-redondant. Dénотons par SPJRU l’algèbre relationnelle qui ne contient pas la différence.

La démonstration fera appel aux bases de données $\mathcal{I}$ et $\mathcal{J}$ de la figure 3.1, et l’expression algébrique

$$E_0 := \pi_A(\sigma_{B=\text{“1”}}(R)) - \pi_A(\sigma_{B=\text{“2”}}(R)).$$

On a $\llbracket E_0 \rrbracket^{\mathcal{I}} = \{(A, a)\}$ et $\llbracket E_0 \rrbracket^{\mathcal{J}} = \{\}$. Remarquons que $\mathcal{J}$ ajoute un tuple à $\mathcal{I}$, mais que $\llbracket E_0 \rrbracket^{\mathcal{J}} \subsetneq \llbracket E_0 \rrbracket^{\mathcal{I}}$. On appelle cette requête *non-monotone* : l’insertion de tuples dans la base de données peut avoir comme effet la suppression de tuples de la réponse. Intuitivement, la différence est non-redondante car c’est le seul opérateur permettant de ressortir cet effet non-monotone.

De façon plus formelle, démontrons qu’aucune expression algébrique de l’algèbre SPJRU (i.e., l’algèbre sans la différence) n’est équivalente à E_0 . Dans le paragraphe suivant, on prouve que pour toute expression algébrique E de l’algèbre SPJRU, $\llbracket E \rrbracket^{\mathcal{I}} \subseteq \llbracket E \rrbracket^{\mathcal{J}}$. À partir de $\llbracket E \rrbracket^{\mathcal{I}} \subseteq \llbracket E \rrbracket^{\mathcal{J}}$ et $\llbracket E_0 \rrbracket^{\mathcal{J}} \subsetneq \llbracket E_0 \rrbracket^{\mathcal{I}}$, il est correct de conclure que soit $\llbracket E \rrbracket^{\mathcal{I}} \neq \llbracket E_0 \rrbracket^{\mathcal{I}}$, soit $\llbracket E \rrbracket^{\mathcal{J}} \neq \llbracket E_0 \rrbracket^{\mathcal{J}}$ (soit les deux), et donc $E \neq E_0$.

On prouve maintenant, par induction, que pour toute expression algébrique E de l’algèbre SPJRU, $\llbracket E \rrbracket^{\mathcal{I}} \subseteq \llbracket E \rrbracket^{\mathcal{J}}$. La base de l’induction est $E = R$, où R est un nom de relation. Par notre choix de $\mathcal{I}$ et $\mathcal{J}$ (voir figure 3.1), on a bien que $\llbracket R \rrbracket^{\mathcal{I}} = R^{\mathcal{I}}$ est un sous-ensemble de $\llbracket R \rrbracket^{\mathcal{J}} = R^{\mathcal{J}}$. Les autres possibilités pour E sont comme suit :

- E est une sélection $\sigma_{C=\text{“c”}}(E_1)$. Noter que même si $\text{sorte}(R) = \{A, B\}$, il est possible que $C \in \text{sorte}(E_1)$, car E_1 peut contenir un renommage. L’hypothèse d’induction est $\llbracket E_1 \rrbracket^{\mathcal{I}} \subseteq \llbracket E_1 \rrbracket^{\mathcal{J}}$, c’est-à-dire, l’évaluation de E_1 sur $\mathcal{I}$ est un sous-ensemble de l’évaluation de E_1 sur $\mathcal{J}$. Il est alors évident que l’évaluation de $\sigma_{C=\text{“c”}}(E_1)$ sur $\mathcal{I}$ est un sous-ensemble de l’évaluation de $\sigma_{C=\text{“c”}}(E_1)$ sur $\mathcal{J}$, i.e., $\llbracket \sigma_{C=\text{“c”}}(E_1) \rrbracket^{\mathcal{I}} \subseteq \llbracket \sigma_{C=\text{“c”}}(E_1) \rrbracket^{\mathcal{J}}$.
- E est une sélection $\sigma_{C=D}(E_1)$. Raisonnement similaire au premier point.
- E est une projection $\pi_X(E_1)$. Raisonnement similaire au premier point.
- E est un renommage $\rho_{C \mapsto D}(E_1)$. Raisonnement similaire au premier point.
- E est une jointure $E_1 \bowtie E_2$. L’hypothèse d’induction est $\llbracket E_1 \rrbracket^{\mathcal{I}} \subseteq \llbracket E_1 \rrbracket^{\mathcal{J}}$ et $\llbracket E_2 \rrbracket^{\mathcal{I}} \subseteq \llbracket E_2 \rrbracket^{\mathcal{J}}$. Il est alors évident que $\llbracket E_1 \bowtie E_2 \rrbracket^{\mathcal{I}} \subseteq \llbracket E_1 \bowtie E_2 \rrbracket^{\mathcal{J}}$.
- E est une union $E_1 \cup E_2$. Raisonnement similaire au point précédent.

Le paragraphe précédent démontre en fait que l’algèbre SPJRU ne permet pas d’encoder des requêtes non-monotones. La notion de monotonie est définie, de façon générale, comme

suit.

Soit E une expression algébrique relative à un schéma-de-base-de-données $\mathcal{R}$. On dit que E est *monotone* si la propriété suivante est vérifiée pour toutes bases de données $\mathcal{I}$ et $\mathcal{J}$ sur $\mathcal{R}$: si $R^{\mathcal{I}} \subseteq R^{\mathcal{J}}$ pour tout $R \in \mathcal{R}$, alors $\llbracket E \rrbracket^{\mathcal{I}} \subseteq \llbracket E \rrbracket^{\mathcal{J}}$. Une expression algébrique est *non-monotone* si elle n'est pas monotone.

Proposition 2. *Chaque requête de l'algèbre SPJRU (i.e., l'algèbre sans la différence) est monotone.*

La Proposition 2 explique pourquoi la requête “*Quels buveurs sont exigeants ?*” de la section 3.3 a besoin de la différence. Supposons un buveur b qui est exigeant selon la base de données actuelle. Insérons ensuite dans la relation ABUS un tuple qui associe à b un vin de qualité C . Dans la nouvelle base de données, le buveur b ne sera plus exigeant. Ce scénario montre que la requête “*Quels buveurs sont exigeants ?*” est non-monotone, et donc pas exprimable en SPJRU.

3.6 Fermeture transitive

Voir chapitre 1 pour la définition de fermeture transitive. Il est connu [1, page 436] qu'il n'existe pas de requête en SPJRUD qui calcule la fermeture transitive d'une relation binaire. Cependant, l'algèbre permet de tester si une relation binaire est transitive. Soit R un nom de relation avec $\text{sorte}(R) = \{A, B\}$. Soit

$$E_1 := \pi_{AB}(\rho_{B \rightarrow C}(R) \bowtie \rho_{A \rightarrow C}(R)).$$

L'expression E_1 renvoie un tuple $\{(A, a), (B, b)\}$ dès que la base de données contient deux tuples comme suit :

R	A	B
	a	c
	c	b
	$\vdots$	$\vdots$

L'expression

$$E_2 := \pi_{\{\}}(E_1 - R)$$

teste si E_1 contient un tuple qui n'est pas dans R . Si c'est le cas, la relation R n'est pas transitive. Si E_2 retourne la relation vide, la relation R est transitive. Noter que selon l'encodage de **true** et **false** expliqué dans la section 3.3, la requête E_2 demande en fait “*Est-ce que la relation R n'est pas transitive ?*”.

3.7 La perspective “Unnamed”

Dans cette section, on définit l'algèbre relationnelle dans la perspective “Unnamed”. Un exercice intéressant est de démontrer que cette algèbre a exactement la même expressivité que l'algèbre SPJRUD.

3.7.1 Syntaxe

Soit $\mathcal{R}$ un schéma-de-bases-de-données.

Noms de relation Chaque nom de relation R dans $\mathcal{R}$ est une expression algébrique d'arité $\text{arité}(R)$.

Sélection “attribut égal constante” Si E est une expression algébrique d’arité n , $1 \leq i \leq n$, et a une constante, alors $\sigma_{i=a}(E)$ est une expression algébrique d’arité n .

Sélection “attribut égal attribut” Si E est une expression algébrique d’arité n et $1 \leq i, j \leq n$, alors $\sigma_{i=j}(E)$ est une expression algébrique d’arité n .

Projection Si E est une expression algébrique d’arité n et $1 \leq i_1, i_2, \dots, i_k \leq n$, alors $\pi_{i_1, i_2, \dots, i_k}(E)$ est une expression algébrique d’arité k .

Produit cartésien Si E et F sont des expressions algébriques d’arités m et n , respectivement, alors $E \times F$ est une expression algébrique d’arité $m + n$.

Union Si E et F sont deux expressions algébriques de même arité n , alors $E \cup F$ est une expression algébrique d’arité n .

Différence Si E et F sont deux expressions algébriques de même arité n , alors $E - F$ est une expression algébrique d’arité n .

3.7.2 Sémantique

Soit $\mathcal{I}$ une base de données sur $\mathcal{R}$. La définition de $\llbracket E \rrbracket^{\mathcal{I}}$ est comme suit :

- Si R est un nom de relation, alors $\llbracket R \rrbracket^{\mathcal{I}} = R^{\mathcal{I}}$.
- $\llbracket \sigma_{i=a}(E) \rrbracket^{\mathcal{I}} = \{ \langle a_1, a_2, \dots, a_n \rangle \in \llbracket E \rrbracket^{\mathcal{I}} \mid a_i = a \}$.
- $\llbracket \sigma_{i=j}(E) \rrbracket^{\mathcal{I}} = \{ \langle a_1, a_2, \dots, a_n \rangle \in \llbracket E \rrbracket^{\mathcal{I}} \mid a_i = a_j \}$.
- $\llbracket \pi_{i_1, i_2, \dots, i_k}(E) \rrbracket^{\mathcal{I}} = \{ \langle a_{i_1}, a_{i_2}, \dots, a_{i_k} \rangle \mid \langle a_1, a_2, \dots, a_n \rangle \in \llbracket E \rrbracket^{\mathcal{I}} \}$.
- $\llbracket E \times F \rrbracket^{\mathcal{I}} = \{ \langle a_1, a_2, \dots, a_m, b_1, b_2, \dots, b_n \rangle \mid \langle a_1, \dots, a_m \rangle \in \llbracket E \rrbracket^{\mathcal{I}}, \langle b_1, \dots, b_n \rangle \in \llbracket F \rrbracket^{\mathcal{I}} \}$.
- $\llbracket E \cup F \rrbracket^{\mathcal{I}} = \llbracket E \rrbracket^{\mathcal{I}} \cup \llbracket F \rrbracket^{\mathcal{I}}$, où $\cup$ est l’union ensembliste.
- $\llbracket E - F \rrbracket^{\mathcal{I}} = \llbracket E \rrbracket^{\mathcal{I}} \setminus \llbracket F \rrbracket^{\mathcal{I}}$, où $\setminus$ est la différence ensembliste.

3.7.3 Exemple

La requête E_3 retourne les buveurs exigeants :

$$\begin{aligned} E_1 &:= \pi_{1,6}(\sigma_{2=4}(\sigma_{3=5}(\text{ABUS} \times \text{VINS}))) \\ E_2 &:= \pi_1(E_1 \setminus \sigma_{2=\text{A}}(E_1)) \\ E_3 &:= \pi_1(\text{ABUS}) - E_2 \end{aligned}$$

Noter que la projection permet de permuter et de dupliquer des colonnes. Par exemple,

$$R \begin{array}{c|cc} 1 & 2 & \\ \hline a & b & \\ c & d & \end{array} \quad \pi_{2,1,1}(R) \begin{array}{c|ccc} 1 & 2 & 3 & \\ \hline b & a & a & \\ d & c & c & \end{array}$$

Chapitre 4

Le Calcul Relationnel

Le calcul relationnel sera introduit dans la perspective “Unnamed”. L’idée est d’utiliser la logique du premier ordre pour interroger une base de données. Prenons la requête “*Quels buveurs sont exigeants ?*” introduite dans la section 3.3. En calcul relationnel, cette requête peut être encodée comme suit :

$$\{b \mid \underbrace{\forall c \forall m (\text{ABUS}(b, c, m) \rightarrow \text{VINS}(c, m, \text{“A”}))}_{\text{partie gauche}} \wedge \underbrace{\exists c' \exists m' \text{ABUS}(b, c', m')}_{\text{partie droite}}\}.$$

La partie droite de la conjonction vérifie que b est bien un buveur (et pas, par exemple, une cafetière). La partie gauche de la conjonction vérifie que chaque vin bu par b est bien de qualité A.

Notez dès le début l’importance de la partie droite. Substituons b par la valeur “Senseo”.¹ Comme Senseo n’est pas un buveur, la prémisse $\text{ABUS}(\text{“Senseo”}, c, m)$ de l’implication sera fausse pour tout c et m . Mais alors la partie gauche de la conjonction est vraie!² La partie droite de la conjonction joue le rôle de garde-fou : puisque $\exists c' \exists m' \text{ABUS}(\text{“Senseo”}, c', m')$ est fausse, la conjonction sera finalement fausse.

4.1 Syntaxe

Supposons un nombre infini de *variables* $x, y, z, x_1, y_1, z_1 \dots$ ³ Aucune variable n’appartient à **dom**. Soit $\mathcal{R}$ un schéma-de-base-de-données.

- Chaque variable x est un terme.
- Chaque constante $c \in \mathbf{dom}$ est un terme.
- Si e_1 et e_2 sont des termes, alors $e_1 = e_2$ est une formule (atomique).
- Si R est un nom de relation d’arité k , et $e_1, \dots, e_k$ son des termes, alors $R(e_1, \dots, e_k)$ est une formule (atomique).

1. Senseo est une marque de cafetière.

2. Rappelez-vous : si la prémisse d’une implication est fausse, l’implication est vraie, quel que soit le conséquent !

3. Pour des raisons de lisibilité, on utilise parfois des variables b, c, m pour dénoter, respectivement, des buveurs, crus et millésimes.

- Si φ_1 et φ_2 sont des formules, alors $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ et $\neg\varphi_1$ sont des formules.
- Si φ est une formule et x une variable, alors $\exists x\varphi$ et $\forall x\varphi$ sont des formules.

L'expression $\varphi \rightarrow \psi$ est une abréviation pour $\neg\varphi \vee \psi$. L'expression $\varphi \leftrightarrow \psi$ est une abréviation pour $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.

L'ensemble des *variables libres* d'une formule φ , dénoté par $free(\varphi)$, est défini comme suit :

- $free(e_1 = e_2) = \{e_1, e_2\} \setminus \mathbf{dom}$;
- $free(R(e_1, \dots, e_k)) = \{e_1, \dots, e_k\} \setminus \mathbf{dom}$;
- $free(\varphi_1 \wedge \varphi_2) = free(\varphi_1 \vee \varphi_2) = free(\varphi_1) \cup free(\varphi_2)$;
- $free(\neg\varphi_1) = free(\varphi_1)$;
- $free(\exists x\varphi) = free(\forall x\varphi) = free(\varphi) \setminus \{x\}$.

Si $free(\varphi) = \{x_1, \dots, x_k\}$, on écrira $\varphi(x_1, \dots, x_k)$.

4.2 Sémantique

La définition mathématique de la sémantique peut sembler plutôt rébarbative. Pourtant, rien n'est plus logique que la logique elle-même $\odot$. Soit $\mathcal{I}$ une base de données sur $\mathcal{R}$. Intuitivement, une formule $R(a_1, \dots, a_k)$ est vraie si $\langle a_1, \dots, a_k \rangle \in R^{\mathcal{I}}$; $a = a$ est vraie et $a = b$ est fausse ; $\varphi_1 \wedge \varphi_2$ est vraie si φ_1 est vraie et φ_2 est vraie ; $\varphi_1 \vee \varphi_2$ est vraie si φ_1 est vraie ou φ_2 est vraie ; $\neg\varphi_1$ est vraie si φ_1 n'est pas vraie ; $\exists x\varphi(x)$ est vraie si φ est vraie pour au moins une valeur de x ; $\forall x\varphi(x)$ est vraie si φ est vraie pour toute valeur de x .

De façon plus formelle, le *domaine actif* d'une base de données $\mathcal{I}$, dénoté par $adom(\mathcal{I})$, est l'ensemble de toutes les constantes qui apparaissent dans $\mathcal{I}$.

Soit $\mathcal{I}$ une base de données sur le schéma-de-base-de-données $\mathcal{R}$. Fixons un *domaine d'interprétation* $\mathbf{d}$ tel que

$$adom(\mathcal{I}) \subseteq \mathbf{d} \subseteq \mathbf{dom}. \quad (4.1)$$

On définit, pour chaque formule $\varphi(x_1, \dots, x_k)$ et $\langle a_1, \dots, a_k \rangle \in \mathbf{d}^k$, la notion de

$$\mathcal{I}, \mathbf{d} \models \varphi(a_1, \dots, a_k),$$

qui exprime que $\varphi(a_1, \dots, a_k)$ est vraie dans $\mathcal{I}$, relativement au domaine d'interprétation $\mathbf{d}$. On écrira $\vec{x}$ et $\vec{a}$ pour respectivement $\langle x_1, \dots, x_k \rangle$ et $\langle a_1, \dots, a_k \rangle$.

Le tuple $\vec{a}$ donne lieu à une *valuation* $\nu : \mathbf{d} \cup \{x_1, \dots, x_n\} \rightarrow \mathbf{d}$ telle que pour chaque $i \in \{1, \dots, n\}$, $\nu(x_i) = a_i$ et pour chaque $c \in \mathbf{d}$, $\nu(c) = c$. Intuitivement, la valeur de x_i est a_i , et la valeur d'une constante est la constante elle-même.

- $\mathcal{I}, \mathbf{d} \models t_1 = t_2$ si $\nu(t_1) = \nu(t_2)$;
- $\mathcal{I}, \mathbf{d} \models R(e_1, \dots, e_k)$ si $\langle \nu(e_1), \dots, \nu(e_k) \rangle \in R^{\mathcal{I}}$;

- si $\varphi(\vec{x}) \equiv \varphi_1(\vec{x}) \wedge \varphi_2(\vec{x})$, alors $\mathcal{I}, \mathbf{d} \models \varphi(\vec{a})$ si $\mathcal{I}, \mathbf{d} \models \varphi_1(\vec{a})$ et $\mathcal{I}, \mathbf{d} \models \varphi_2(\vec{a})$; ⁴
- si $\varphi(\vec{x}) \equiv \varphi_1(\vec{x}) \vee \varphi_2(\vec{x})$, alors $\mathcal{I}, \mathbf{d} \models \varphi(\vec{a})$ si $\mathcal{I}, \mathbf{d} \models \varphi_1(\vec{a})$ ou $\mathcal{I}, \mathbf{d} \models \varphi_2(\vec{a})$;
- si $\varphi(\vec{x}) \equiv \neg \varphi_1(\vec{x})$, alors $\mathcal{I}, \mathbf{d} \models \varphi(\vec{a})$ si $\mathcal{I} \not\models \varphi_1(\vec{a})$;
- si $\varphi(\vec{x}) \equiv \exists y \psi(y, \vec{x})$, alors $\mathcal{I}, \mathbf{d} \models \varphi(\vec{a})$ s'il existe $c \in \mathbf{d}$ tel que $\mathcal{I}, \mathbf{d} \models \psi(c, \vec{a})$;
- si $\varphi(\vec{x}) \equiv \forall y \psi(y, \vec{x})$, alors $\mathcal{I}, \mathbf{d} \models \varphi(\vec{a})$ si $\mathcal{I}, \mathbf{d} \models \psi(c, \vec{a})$ pour tout $c \in \mathbf{d}$.

Une requête en calcul relationnel est une formule

$$\{x_1, \dots, x_k \mid \varphi(x_1, \dots, x_k)\}.$$

La réponse à cette requête contient tout tuple $\langle a_1, \dots, a_k \rangle$ tel que $\mathcal{I}, \mathbf{d} \models \varphi(a_1, \dots, a_k)$.

Une légère extension consiste à permettre que la même variable apparaisse plusieurs fois à gauche de la barre verticale. Par exemple (voir aussi la section 3.7.3),

$$\{x_1, x_2, x_2 \mid R(x_2, x_1)\}.$$

Noter que le domaine d'interprétation $\mathbf{d}$ intervient à trois endroits dans le calcul de la réponse à une requête :

1. les tuples de la réponse sont choisis dans $\mathbf{d}^k$;
2. dans le test $\mathcal{I}, \mathbf{d} \models \exists y \psi(y, \vec{a})$, on cherche la valeur de y dans $\mathbf{d}$; et
3. dans le test $\mathcal{I}, \mathbf{d} \models \forall y \psi(y, \vec{a})$, on prend, comme valeur pour y , toute valeur de $\mathbf{d}$.

La question qui vient à l'esprit est : “Est-ce que la réponse à une requête peut varier selon le choix de $\mathbf{d}$?”.

4.3 Indépendance du domaine

Malheureusement, le calcul relationnel défini précédemment a ses défauts. Prenons la requête :

$$\{b \mid \forall c \forall m (\text{ABUS}(b, c, m) \rightarrow \text{VINS}(c, m, \text{“A”}))\}. \quad \textcircled{1}$$

Comme expliqué dans l'introduction de ce chapitre, la formule à droite de la barre verticale est vraie si l'on remplace b par les valeurs “Senseo” ou “Chablis”, qui appartiennent donc à la réponse, à moins que “Senseo” et “Chablis” soient des valeurs dans $\mathbf{d}$. ⁵ Cette requête dépend donc du choix du domaine d'interprétation $\mathbf{d}$. En plus, si $\mathbf{d}$ est un ensemble infini, la réponse à cette requête sera infinie (parce que chaque base de données est finie).

Prenons une autre requête :

$$\{c \mid \forall m \text{VINS}(c, m, \text{“A”})\}. \quad \textcircled{2}$$

Intuitivement, on cherche les crus qui ont toujours été de qualité A. La réponse est certainement finie, car c doit apparaître dans la table VINS. Cependant, pour chaque cru c , on peut toujours trouver une valeur pour m dans $\mathbf{d}$ telle que $\text{VINS}(c, m, \text{“A”})$ est faux dans la base de données : il suffit de choisir une valeur qui n'est pas un millésime. Donc, la réponse à cette requête

4. On suppose ici que $\text{free}(\varphi_1) = \text{free}(\varphi_2)$. Noter que l'on peut toujours “introduire” une nouvelle variable libre z dans une formule φ en remplaçant φ par $(z = z) \wedge \varphi$.

5. Pour la base de données de la figure 2.2, “Chablis” appartient au domaine actif et donc au domaine d'interprétation.

sera toujours vide. Une solution pourrait consister à restreindre le domaine de la variable m à un ensemble fini de millésimes. De nouveau, la réponse dépendrait du domaine choisi, qui ne correspond peut-être pas au domaine envisagé par l'utilisateur qui soumet la requête. En plus, la qualité d'un cru peut manquer pour certaines années. Vu que la base de données de la figure 2.2 n'enregistre pas la qualité du Chablis en 1979, est-il correct de dire que le Chablis a toujours été de qualité A ? Les requêtes suivantes ne posent pas ces défauts, car elles contrôlent elles-même le domaine de m . La requête suivante exige que c ait été de qualité A dans *tout millésime m pour lequel la qualité de c a été enregistrée* :

$$\{c \mid \exists m' \exists q' \text{VINS}(c, m', q') \wedge \forall m \forall q (\text{VINS}(c, m, q) \rightarrow q = \text{"A"})\}.$$

La requête suivante est plus sévère ; elle exige que c a été de qualité A dans *tout millésime m qui apparaît dans la colonne Millésime* :

$$\{c \mid \exists m' \exists q' \text{VINS}(c, m', q') \wedge \forall c' \forall m \forall q (\text{VINS}(c', m, q) \rightarrow \text{VINS}(c, m, \text{"A"})\}.$$

Comme déjà illustré par la requête $\odot 1$, contrairement à ce que l'on pouvait penser, une interprétation relative au domaine actif n'est pas toujours naturelle. Soit $\mathcal{I}$ la base de données suivante :

R	Modèle	Couleur
	Clio	bleu
	Laguna	bleu
	Laguna	rouge

Considérons la requête “*Quels modèles sont disponibles en chaque couleur ?*”. La réponse attendue est $\{\langle \text{Laguna} \rangle\}$. Prenons la requête :

$$\{m \mid \exists c' R(m, c') \wedge \forall c R(m, c)\}. \quad \odot 3$$

Une interprétation relative au domaine actif ne donne pas la bonne réponse : puisque $\langle \text{Laguna}, \text{Clio} \rangle$ n'est pas un tuple de R et “Clio” appartient au domaine actif, la sous-formule $\forall c R(\text{"Laguna"}, c)$ n'est pas vraie. La bonne requête est :

$$\{m \mid \exists c' R(m, c') \wedge \forall m' \forall c (R(m', c) \rightarrow R(m, c))\},$$

ce qui est équivalent à :

$$\{m \mid \exists c' R(m, c') \wedge \neg \exists m' \exists c (R(m', c) \wedge \neg R(m, c))\}.$$

On dira qu'une requête $\{\vec{x} \mid \varphi(\vec{x})\}$ est *indépendante du domaine* [d'interprétation] (en anglais : *domain independent*) si la réponse à la requête ne dépend pas du choix de $\mathbf{d}$ que l'on fait dans (4.1). De façon précise, une requête $\{\vec{x} \mid \varphi(\vec{x})\}$ est indépendante du domaine si pour toute base de données $\mathcal{I}$, pour tout $\mathbf{d}_1, \mathbf{d}_2$ tels que $\text{adom}(\mathcal{I}) \subseteq \mathbf{d}_1, \mathbf{d}_2 \subseteq \mathbf{dom}$, la réponse relative à $\mathbf{d}_1$ est identique à la réponse relative à $\mathbf{d}_2$. En conséquence, la réponse à une requête qui est indépendante du domaine peut être calculée en prenant $\text{adom}(\mathcal{I})$ comme domaine d'interprétation.⁶

Suite aux explications précédentes, on peut conclure que les requêtes $\odot 1$, $\odot 2$ et $\odot 3$ ne sont pas indépendantes du domaine. Voici d'autres exemples de requêtes qui ne sont pas indépendantes du domaine.

6. Techniquement, cette analyse n'est correcte que pour des requêtes sans constantes. Si une requête contient des constantes, celles-ci doivent aussi faire partie du domaine d'interprétation. Par exemple, la requête $\{x \mid R(x, 1) \wedge \neg R(x, 0)\}$, avec les constantes 0 et 1, devrait être interprétée relative à $\text{adom}(\mathcal{I}) \cup \{0, 1\}$.

Exemple avec disjonction

$$\{x \mid R(a) \vee R(b) \vee R(x)\} \quad \textcircled{4}$$

Soit $\mathcal{I}_1$ la base de données suivante :

R	A
	a
	c

On a $\text{adom}(\mathcal{I}_1) = \{a, c\}$. L'interprétation relative à $\{a, c\}$ donne la réponse $\{\langle a \rangle, \langle c \rangle\}$. Soit d une constante telle que $d \notin \{a, c\}$. L'interprétation relative à $\{a, c, d\}$ donne la réponse $\{\langle a \rangle, \langle c \rangle, \langle d \rangle\}$. La requête $\textcircled{4}$ n'est donc pas indépendante du domaine.

Exemple avec négation

$$\{x \mid \neg R(x)\} \quad \textcircled{5}$$

Soit $\mathcal{I}_2$ la base de donnée suivante :

R	A
	a

On a $\text{adom}(\mathcal{I}_2) = \{a\}$. L'interprétation relative à $\{a\}$ donne la réponse vide. L'interprétation relative à $\{a, d\}$, $d \neq a$, donne la réponse $\{\langle d \rangle\}$.

Exemple avec quantification universelle

$$\{x \mid \forall y R(x, y)\} \quad \textcircled{6}$$

Soit $\mathcal{I}_3$ la base de donnée suivante :

R	A	B
	a	a
	a	b

On a $\text{adom}(\mathcal{I}_3) = \{a, b\}$. L'interprétation relative à $\{a, b\}$ donne la réponse $\{\langle a \rangle\}$. L'interprétation relative à $\{a, b, d\}$ donne la réponse vide, parce que $\langle a, d \rangle \notin R^{\mathcal{I}_3}$.

4.4 Des requêtes dites safe

La section 4.3 a amplement illustré les défauts des requêtes qui ne sont pas indépendantes du domaine. Désormais, une requête qui n'est pas indépendante du domaine sera considérée comme erronée. Malheureusement, il est connu [1, Corollary 6.3.2] qu'il n'existe pas d'algorithme qui prend en entrée une requête en calcul relationnel et qui détermine si, oui ou non, cette requête est indépendante du domaine (voir section 4.5).

La solution consistera à imposer des restrictions syntaxiques qui garantissent l'indépendance du domaine. Ces restrictions peuvent prendre différentes formes. On introduit ici des restrictions qui sont directement inspirées de l'algèbre relationnelle.

La notion de variable libre reste inchangée et n'est pas répétée ci-après.

Nom de relation Si R est un nom de relation d'arité n , et $x_1, \dots, x_n$ sont des variables distinctes, alors $R(x_1, \dots, x_n)$ est une formule safe (atomique). La restriction que $x_1, \dots, x_n$ soient distinctes fait que chaque variable correspond à exactement un attribut de $\text{sorte}(R)$.

Sélection Si φ est une formule safe, a est une constante, et $x, y \in \text{free}(\varphi)$, alors $\varphi \wedge (x = "a")$ et $\varphi \wedge (x = y)$ sont des formules safe.

Projection Si φ est une formule safe et $x \in \text{free}(\varphi)$, alors $\exists x\varphi$ est une formule safe.

Jointure Si φ_1 et φ_2 sont des formules safe, alors $\varphi_1 \wedge \varphi_2$ est une formule safe.

Union Si φ_1 et φ_2 sont des formules safe avec $\text{free}(\varphi_1) = \text{free}(\varphi_2)$, alors $\varphi_1 \vee \varphi_2$ est une formule safe. La restriction que φ_1 et φ_2 aient exactement les mêmes variables libres provient de la restriction que l'union $E_1 \cup E_2$ est seulement définie si $\text{sorte}(E_1) = \text{sorte}(E_2)$. Noter que la formule dans $\odot 4$ n'est donc pas safe.

Différence Si φ_1 et φ_2 sont des formules safe avec $\text{free}(\varphi_1) = \text{free}(\varphi_2)$, alors $\varphi_1 \wedge \neg\varphi_2$ est une formule safe. La restriction que φ_1 et φ_2 aient exactement les mêmes variables libres provient de la restriction que la différence $E_1 - E_2$ est seulement définie si $\text{sorte}(E_1) = \text{sorte}(E_2)$. Intuitivement, la formule φ_1 est un garde-fou qui fixe le domaine des variables libres. Noter que la formule dans $\odot 5$ n'est donc pas safe.

Une requête $\{\vec{x} \mid \varphi(\vec{x})\}$ est safe si la formule $\varphi(\vec{x})$ est safe. Dénotons par SRC (Safe Relational Calculus) la restriction du calcul relationnel aux formules et requêtes safe.

Un exercice intéressant est de démontrer les résultats suivants (voir aussi les transparents) :

- Chaque requête en SPJRUD est équivalente à une requête en SRC.
- Chaque requête en SRC est équivalente à une requête en SPJRUD.
- Chaque requête en SRC est indépendante du domaine (une conséquence des deux assertions qui précèdent).

SRC impose une syntaxe très restrictive aux formules. Malgré ces restrictions, l'assertion suivante est vraie :

- Chaque requête qui est indépendante du domaine est équivalente à une requête en SRC (voir section 4.5).

Bien que les restrictions syntaxiques ne diminuent donc pas l'expressivité, elles rendent le langage SRC rébarbatif à utiliser. Par exemple, la requête $\{x \mid R(x, x)\}$ doit s'écrire comme $\{x \mid \exists y (R(x, y) \wedge x = y)\}$ en SRC. Dans [1, Section 5.4], on trouve des restrictions moins sévères du calcul relationnel qui garantissent aussi l'indépendance du domaine.

4.5 Discussions approfondies

Sur la base de la Proposition 1, on peut facilement prouver qu'il est indécidable de déterminer si, oui ou non, une formule en calcul relationnel est indépendante du domaine. Considérons une requête de la forme :

$$\{\vec{x} \mid \psi(\vec{x}) \wedge \neg(\varphi_1 \leftrightarrow \varphi_2)\}, \quad (4.2)$$

avec φ_1 et φ_2 des formules booléennes (i.e., sans variables libres) en SRC, et ψ une formule en calcul relationnel telle que $\{\vec{x} \mid \psi(\vec{x})\}$ n'est pas indépendante du domaine. Deux cas sont possibles :

1. Si $\varphi_1 \equiv \varphi_2$, alors $\neg(\varphi_1 \leftrightarrow \varphi_2)$ est faux, et la requête (4.2) retourne la réponse vide sur toute base de données, quel que soit le domaine d'interprétation. Dans ce cas, la requête est donc indépendante du domaine.
2. Si $\varphi_1 \not\equiv \varphi_2$, alors $\neg(\varphi_1 \leftrightarrow \varphi_2)$ est vrai, et la requête (4.2) est équivalente à $\{\vec{x} \mid \psi(\vec{x})\}$, une requête qui n'est pas indépendante du domaine.

En conclusion, la requête (4.2) est indépendante du domaine si et seulement si $\varphi_1 \equiv \varphi_2$. Mais on sait, par la Proposition 1, que l'équivalence de φ_1 et φ_2 est indécidable. Noter ici que la Proposition 1 peut être appliquée au langage SRC, car on sait qu'il est possible de "traduire"

des requêtes entre SRC et SPJRUD. Il est alors correct de conclure qu'il est indécidable de déterminer si, oui ou non, une formule en calcul relationnel est indépendante du domaine.

On explique maintenant pourquoi chaque requête qui est indépendante du domaine est équivalente à une requête en SRC. Noter d'abord qu'il existe une requête $\{x \mid \alpha(x)\}$ en SRC qui prend en entrée une base de données et qui retourne son domaine actif. Par exemple, supposons que le schéma-de-base-de-données consiste en deux noms de relations, R et S , chacun d'arité 2. Alors $\alpha(x)$ est la formule suivante :

$$\alpha(x) := (\exists y R(x, y) \vee \exists y R(y, x)) \vee (\exists y S(x, y) \vee \exists y S(y, x)).$$

Soit $\{\vec{x} \mid \psi(\vec{x})\}$ une requête en calcul relationnel qui est indépendante du domaine. La requête en SRC équivalente peut être construite comme suit.

- Assurer que toute formule atomique respecte les contraintes syntaxiques imposées par SRC. Par exemple, remplacer $R(x, "a")$ par $R(x, y) \wedge (y = "a")$; remplacer $R(x, x)$ par $R(x, y) \wedge (y = x)$;
- remplacer chaque sous-formule $x = "a"$ de ψ par $\alpha(x) \wedge (x = "a")$;
- remplacer chaque sous-formule $x = y$ de ψ par $(\alpha(x) \wedge \alpha(y)) \wedge (x = y)$;
- remplacer chaque négation $\neg\varphi(x_1, \dots, x_n)$ par $(\alpha(x_1) \wedge \dots \wedge \alpha(x_n)) \wedge \neg\varphi(x_1, \dots, x_n)$;
- assurer que les parties gauche et droite d'une disjonction aient les mêmes variables libres. Par exemple, remplacer $\varphi_1(x, y) \vee \varphi_2(y, z)$ par $(\varphi_1(x, y) \wedge \alpha(z)) \vee (\varphi_2(y, z) \wedge \alpha(x))$.

On peut vérifier que ces étapes aboutissent à une requête en SRC qui donne les mêmes réponses que $\{\vec{x} \mid \psi(\vec{x})\}$ avec, comme domaine d'interprétation, le domaine actif. Comme $\{\vec{x} \mid \psi(\vec{x})\}$ est indépendante du domaine, une interprétation relative au domaine actif retourne, par définition, la réponse correcte.

4.6 Le calcul relationnel TRC

Dans le calcul introduit précédemment, les variables représentent des valeurs de **dom**. On introduit maintenant brièvement une variante du calcul relationnel dans lequel les variables représentent des tuples. Dans la littérature, on utilise souvent les dénominations DRC et TRC pour distinguer ces deux approches :

Domain Relational Calculus (DRC) : Les variables représentent des valeurs.

Tuple Relational Calculus (TRC) : Les variables représentent des tuples.

4.6.1 TRC par l'exemple

Prenons le schéma-de-bases-de-données suivant, repris du livre [7] :

S[S#, Sname, Status, City]
P[P#, Pname, Color, Weight, City]
SP[S#, P#, Qty]

Les tables S et P enregistrent respectivement des fournisseurs et des produits. La table SP enregistre quel fournisseur peut fournir quel produit, et en quelle quantité. Chaque quantité est un nombre entier strictement positif.

Requête 7.7.4

Trouver tout couple de villes tel qu'un fournisseur habitant la première ville puisse fournir un produit stocké dans la deuxième ville.

La requête en SQL :

```
SELECT  s.City, p.City
FROM    S AS s, SP AS r, P AS p
WHERE   s.S# = r.S#
AND     r.P# = p.P# ;
```

La requête en TRC :

$$\begin{aligned} & \{s.4, p.5 \mid S(s) \wedge P(p) \wedge \exists r (SP(r) \wedge s.1 = r.1 \wedge r.2 = p.1)\} \\ & \quad \equiv \\ & \{s.4, p.5 \mid \exists r (S(s) \wedge P(p) \wedge SP(r) \wedge s.1 = r.1 \wedge r.2 = p.1)\} \end{aligned}$$

Pour améliorer encore la lisibilité, on remplace les indices par les attributs :

$$\begin{aligned} & \{s.City, p.City \mid S(s) \wedge P(p) \wedge \exists r (SP(r) \wedge s.S\# = r.S\# \wedge r.P\# = p.P\#)\} \\ & \quad \equiv \\ & \{s.City, p.City \mid \exists r (S(s) \wedge P(p) \wedge SP(r) \wedge s.S\# = r.S\# \wedge r.P\# = p.P\#)\} \end{aligned}$$

Remarquer que cette dernière requête en TRC est très proche de la requête en SQL.

Requête 7.7.15

Trouver les noms de fournisseurs qui peuvent fournir tout produit rouge.

En SQL :

```
SELECT  s.Sname
FROM    S AS s
WHERE   NOT EXISTS ( SELECT  *
                     FROM    P AS p
                     WHERE   p.Color = "red"
                     AND     NOT EXISTS ( SELECT  *
                                         FROM    SP AS r
                                         WHERE   r.S# = s.S#
                                         AND     r.P# = p.P# ) ) ;
```

En TRC :

$$\begin{aligned} & \{s.2 \mid S(s) \wedge \forall p \in P (p.3 = \text{"red"} \rightarrow \exists r \in SP (r.1 = s.1 \wedge r.2 = p.1))\} \\ & \quad \equiv \\ & \{s.2 \mid S(s) \wedge \neg \exists p \in P (p.3 = \text{"red"} \wedge \neg \exists r \in SP (r.1 = s.1 \wedge r.2 = p.1))\} \end{aligned}$$

Avec les attributs :

$$\begin{aligned} & \{s.Sname \mid S(s) \wedge \forall p \in P (p.Color = \text{"red"} \rightarrow \exists r \in SP (r.S\# = s.S\# \wedge r.P\# = p.P\#))\} \\ & \quad \equiv \\ & \{s.Sname \mid S(s) \wedge \neg \exists p \in P (p.Color = \text{"red"} \wedge \neg \exists r \in SP (r.S\# = s.S\# \wedge r.P\# = p.P\#))\} \end{aligned}$$

4.6.2 Syntaxe

Formules On suppose des variables $r, s, t, \dots$, et une fonction qui fait correspondre, à chaque variable, une arité dans $\{0, 1, 2, \dots\}$.

- Chaque constante $c \in \mathbf{dom}$ est un terme.
- Si t est une variable d'arité n et $i \in \{1, 2, \dots, n\}$, alors $t.i$ est un terme.
- Si e_1 et e_2 sont des termes, alors $e_1 = e_2$ est une formule (atomique).
- Si R est un nom de relation et r une variable de même arité que R , alors $R(r)$ est une formule (atomique).
- Si φ_1 et φ_2 sont des formules, alors $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ et $\neg\varphi_1$ sont des formules.
- Si φ est une formule et r une variable, alors $\exists r\varphi$ et $\forall r\varphi$ sont des formules.

Requêtes Une requête est une expression de la forme

$$\{L \mid \varphi\}$$

avec L une liste de termes et φ une formule telles que

- pour chaque terme $r.i$ dans L , la variable r est une variable libre de φ ; et
- chaque variable libre de φ apparaît dans un terme de L .

Abréviations On introduit un nombre d'abrégations :

- $\varphi_1 \rightarrow \varphi_2$ est une abréviation de $\neg\varphi_1 \vee \varphi_2$;
- $\exists r \in R(\varphi)$ est une abréviation de $\exists r (R(r) \wedge \varphi)$;
- $\forall r \in R(\varphi)$ est une abréviation de $\forall r (R(r) \rightarrow \varphi)$;
- $u_1 \neq u_2$ est une abréviation de $\neg(u_1 = u_2)$.

Noter que ces abréviations sont cohérentes :

$$\begin{aligned}
\forall r \in R(\varphi) &\equiv \neg \neg \forall r \in R(\varphi) && (\text{logique : } \neg \neg p \equiv p) \\
&\equiv \neg \neg \forall r (R(r) \rightarrow \varphi) && (\text{abréviation}) \\
&\equiv \neg \exists r \neg (\neg R(r) \vee \varphi) && (\text{logique : } \neg \forall r \psi \equiv \exists r \neg \psi \text{ et } p \rightarrow q \equiv \neg p \vee q) \\
&\equiv \neg \exists r (R(r) \wedge \neg \varphi) && (\text{De Morgan}) \\
&\equiv \neg \exists r \in R(\neg \varphi) && (\text{abréviation})
\end{aligned}$$

4.6.3 Sémantique

La sémantique du TRC peut être définie de façon rigoureuse, comme on l'a fait pour le DRC dans la section 4.2. Ici, on se contente de donner, de façon informelle, les différences avec le calcul TRC.

Soit $\mathcal{I}$ une base de données et $\mathbf{d}$ un domaine d'interprétation.

- Une variable d'arité n prend ses valeurs dans $\mathbf{d}^n$ (et non dans $\mathbf{d}$).
- $R(r)$ est vrai si le tuple r appartient à la relation $R^{\mathcal{I}}$.

- $\exists r \varphi$ est vrai s'il existe un tuple $r \in \mathbf{d}^n$ pour lequel φ vrai, avec n l'arité de r .
- $\forall r \varphi$ est vrai si pour tout tuple $r \in \mathbf{d}^n$, φ est vrai, avec n l'arité de r .

Dans le calcul TRC, comme dans le calcul DRC, certaines requêtes ne sont pas indépendantes du domaine :

$$\{r.1 \mid R(r) \vee \exists s (S(s))\}; \quad \odot$$

$$\{r.1 \mid \neg R(r)\}. \quad \odot$$

SQL est un mix de l'algèbre relationnelle et le calcul TRC. Par exemple, pour un schéma-de-bases-de-données qui contient $R[A]$ et $S[B]$, la requête TRC $\{r.1 \mid R(r) \vee S(r)\}$ donne lieu à une union en SQL. Noter aussi qu'il serait illogique de vouloir remplacer, dans cette requête TRC, l'indice 1 par un attribut (A ou B).

4.6.4 Du TRC au SQL

Prenons la requête :

Donner les couples (n_1, n_2) de noms de fournisseurs tels que les produits fournis par n_1 sont un sous-ensemble des produits fournis par n_2 .

Il est difficile d'encoder cette requête directement en SQL.

En TRC :

$$\begin{aligned} & \{s_1.2, s_2.2 \mid S(s_1) \wedge S(s_2) \wedge \forall r_1 \in \text{SP} \\ & \quad (r_1.1 = s_1.1 \rightarrow \exists r_2 \in \text{SP} (r_2.1 = s_2.1 \wedge r_2.2 = r_1.2))\} \\ & \quad \equiv \\ & \{s_1.2, s_2.2 \mid S(s_1) \wedge S(s_2) \wedge \neg \exists r_1 \in \text{SP} \\ & \quad (r_1.1 = s_1.1 \wedge \neg \exists r_2 \in \text{SP} (r_2.1 = s_2.1 \wedge r_2.2 = r_1.2))\} \end{aligned}$$

Avec les attributs :

$$\begin{aligned} & \{s_1.\text{Sname}, s_2.\text{Sname} \mid S(s_1) \wedge S(s_2) \wedge \forall r_1 \in \text{SP} \\ & \quad (r_1.S\# = s_1.S\# \rightarrow \exists r_2 \in \text{SP} (r_2.S\# = s_2.S\# \wedge r_2.P\# = r_1.P\#))\} \\ & \quad \equiv \\ & \{s_1.\text{Sname}, s_2.\text{Sname} \mid S(s_1) \wedge S(s_2) \wedge \neg \exists r_1 \in \text{SP} \\ & \quad (r_1.S\# = s_1.S\# \wedge \neg \exists r_2 \in \text{SP} (r_2.S\# = s_2.S\# \wedge r_2.P\# = r_1.P\#))\} \end{aligned}$$

La traduction en SQL :

```

SELECT  s1.Sname, s2.Sname
FROM    S AS s1, S AS s2
WHERE   NOT EXISTS (
    SELECT *
    FROM  SP AS r1
    WHERE r1.S# = s1.S#
    AND   NOT EXISTS (
        SELECT *
        FROM  SP AS r2
        WHERE r2.S# = s2.S#
        AND   r2.P# = r1.P# ) ) ;

```

Chapitre 5

SQL

Le langage SQL fera l'objet des Travaux Pratiques. Ce chapitre n'est qu'une brève introduction. La bases de données et les requêtes sont reprises de [7].

5.1 La base de données

S	S#	SNAME	STATUS	CITY	P	P#	PNAME	COLOR	WEIGHT	CITY
	S1	Smith	20	London		P1	Nut	Red	12	London
	S2	Jones	10	Paris		P2	Bolt	Green	17	Paris
	S3	Blake	30	Paris		P3	Screw	Blue	17	Rome
	S4	Clark	20	London		P4	Screw	Red	14	London
	S5	Adams	30	Athens		P5	Cam	Blue	12	Paris
						P6	Cog	Red	19	London

SP	S#	P#	QTY
	S1	P1	300
	S1	P2	200
	S1	P3	400
	S1	P4	200
	S1	P5	100
	S1	P6	100
	S2	P1	300
	S2	P2	400
	S3	P2	200
	S4	P2	200
	S4	P4	300
	S4	P5	400

5.2 Création de domaines et de tables

```
CREATE DOMAIN COLOR CHAR(6) DEFAULT '???'
CONSTRAINT VALID_COLORS
CHECK ( VALUE IN
        ( 'Red', 'Yellow', 'Blue', 'Green', '???' ) ) ;

CREATE DOMAIN S#    CHAR(5) ;
...
CREATE DOMAIN QTY   NUMERIC(9) ;
```

```

CREATE TABLE S
( S#      S#,
  SNAME    NAME,
  STATUS   STATUS,
  CITY     CITY,
  PRIMARY KEY ( S# ) ) ;

CREATE TABLE P
( P#      P#,
  PNAME    NAME,
  COLOR    COLOR,
  WEIGHT    WEIGHT,
  CITY     CITY,
  PRIMARY KEY ( P# ) ) ;

CREATE TABLE SP
( S# S# NOT NULL, P# P# NOT NULL, QTY QTY NOT NULL,
  PRIMARY KEY ( S#, P# ),
  FOREIGN KEY ( S# ) REFERENCES S
                        ON DELETE CASCADE
                        ON UPDATE CASCADE,
  FOREIGN KEY ( P# ) REFERENCES P
                        ON DELETE CASCADE
                        ON UPDATE CASCADE,
  CHECK ( QTY > 0 AND QTY < 5001 ) ) ;

```

5.3 Retrouver des informations

Get color and city for “nonParis” parts with weight greater than 10.

```

SELECT P.COLOR, P.CITY
FROM   P
WHERE  P.CITY <> 'Paris'
AND    P.WEIGHT > 10 ;

SELECT DISTINCT P.COLOR, P.CITY
FROM   P
WHERE  P.CITY <> 'Paris'
AND    P.WEIGHT > 10 ;

SELECT DISTINCT P.COLOR, P.CITY
FROM   P
WHERE  P.CITY <> 'Paris'
AND    P.WEIGHT > 10
ORDER BY CITY DESC ;

```

For all parts, get the part number and the weight of that part in grams. Supposons que le poids est donné en livres (=454 g).

```
SELECT P.P#, P.WEIGHT * 454 AS GMWT
FROM   P ;
```

Get full details of all suppliers.

```
SELECT *
FROM   S ;

SELECT S.*
FROM   S ;
```

Get all combinations of supplier number and part number such that the supplier and part in question are colocated.

```
SELECT S.S#, P.P#
FROM   S, P
WHERE  S.CITY = P.CITY ;
```

Sémantique. Premièrement, “FROM S, P” donne le produit cartésien de S et P.

S.S#	S.SNAME	S.STATUS	S.CITY	P.P#	P.PNAME	P.COLOR	P.WEIGHT	P.CITY
S1	Smith	20	London	P1	Nut	Red	12	London
S1	Smith	20	London	P2	Bolt	Green	17	Paris
S1	Smith	20	London	P3	Screw	Blue	17	Rome
S1	Smith	20	London	P4	Screw	Red	14	London
S1	Smith	20	London	P5	Cam	Blue	12	Paris
S1	Smith	20	London	P6	Cog	Red	19	London
S2	Jones	10	Paris	P1	Nut	Red	12	London
S2	Jones	10	Paris	P2	Bolt	Green	17	Paris
S2	Jones	10	Paris	P3	Screw	Blue	17	Rome
S2	Jones	10	Paris	P4	Screw	Red	14	London
S2	Jones	10	Paris	P5	Cam	Blue	12	Paris
S2	Jones	10	Paris	P6	Cog	Red	19	London
...								
S5	Adams	30	Athens	P6	Cog	Red	19	London

Deuxièmement, “WHERE S.CITY = P.CITY” sélectionne les tuples satisfaisant la condition.

S.S#	S.SNAME	S.STATUS	S.CITY	P.P#	P.PNAME	P.COLOR	P.WEIGHT	P.CITY
S1	Smith	20	London	P1	Nut	Red	12	London
S1	Smith	20	London	P4	Screw	Red	14	London
S1	Smith	20	London	P6	Cog	Red	19	London
S2	Jones	10	Paris	P2	Bolt	Green	17	Paris
S2	Jones	10	Paris	P5	Cam	Blue	12	Paris
...								

Finalement, “SELECT S.S#, P.P#” sélectionne (ou plutôt : projette) les colonnes mentionnées.

S#	P#
S1	P1
S1	P4
S1	P6
S2	P2
S2	P5
...	

Get all pairs of city names such that a supplier located in the first city supplies a part stored in the second city.

```
SELECT S.CITY, P.CITY
FROM   S, SP, P
WHERE  S.S# = SP.S#
AND    SP.P# = P.P# ;
```

Get the total number of suppliers.

```
SELECT COUNT(*) AS N
FROM   S ;
```

Get the maximum and the minimum quantity for part P2.

```
SELECT MAX ( SP.QTY ) AS MAXQ, MIN ( SP.QTY ) AS MINQ
FROM   SP
WHERE  SP.P# = 'P2' ;
```

For each part supplied, get the part number and the total shipment quantity.

```
SELECT SP.P#, SUM ( SP.QTY ) AS TOTQTY
FROM   SP
GROUP BY SP.P# ;
```

Sémantique. Premièrement, imaginons que “FROM SP GROUP BY SP.P#” donne la “table” suivante.

{S#}	P#	{QTY}
{S1, S2}	P1	{300, 300}
{S1, S2, S3, S4}	P2	{200, 400, 200, 200}
{S1}	P3	{400}
{S1, S4}	P4	{200, 300}
{S1, S4}	P5	{100, 400}
{S1}	P6	{100}

Finalement, “SELECT SP.P#, SUM (SP.QTY) AS TOTQTY” donne :

P#	TOTQTY
P1	600
P2	1000
P3	400
P4	500
P5	500
P6	100

```
SELECT P.P#, ( SELECT SUM ( SP.QTY )
                FROM   SP
                WHERE  SP.P# = P.P# ) AS TOTQTY
FROM   P ;
```

Get part number for all parts supplied by more than one supplier.


```

SELECT SP.P#
FROM    SP
GROUP   BY SP.P#
HAVING  COUNT ( SP.S# ) > 1 ;

```

Get supplier names for suppliers who supply part P2.

```

SELECT DISTINCT S.SNAME
FROM    S
WHERE   S# IN
        ( SELECT SP.S#
          FROM    SP
          WHERE   SP.P# = 'P2' ) ;

```

```

SELECT DISTINCT S.SNAME
FROM    S
WHERE   EXISTS
        ( SELECT *
          FROM    SP
          WHERE   SP.P# = 'P2'
          AND     SP.S# = S.S# ) ;

```

```

SELECT DISTINCT S.SNAME
FROM    S, SP
WHERE   S.S# = SP.S#
AND     SP.P# = 'P2' ;

```

Get supplier numbers for suppliers with status less than the current maximum status in the S table.

```

SELECT S.S#
FROM    S
WHERE   S.STATUS <
        ( SELECT MAX ( S.STATUS )
          FROM    S ) ;

```

Get supplier names for suppliers who do not supply part P2.

```
SELECT DISTINCT S.SNAME
FROM   S
WHERE  S# NOT IN
      ( SELECT SP.S#
        FROM   SP
        WHERE  SP.P# = 'P2' ) ;

SELECT DISTINCT S.SNAME
FROM   S
WHERE  NOT EXISTS
      ( SELECT *
        FROM   SP
        WHERE  SP.P# = 'P2'
        AND    SP.S# = S.S# ) ;
```

Get supplier names for suppliers who supply all red parts.

```
SELECT S.SNAME
FROM   S
WHERE  NOT EXISTS
      ( SELECT *
        FROM   P
        WHERE  P.COLOR = 'Red'
        AND    NOT EXISTS
              ( SELECT *
                FROM   SP
                WHERE  SP.S# = S.S#
                AND    SP.P# = P.P# ) ) ;
```

Get part numbers for parts that either weigh more than 16 pounds or are supplied by supplier S2, or both.

```
SELECT P.P#
FROM   P
WHERE  P.WEIGHT > 16
UNION
SELECT SP.P#
FROM   SP
WHERE  SP.S# = 'S2' ;
```

5.4 Mises à jour

Single-row INSERT.

```
INSERT
INTO   P ( P#, PNAME, COLOR, WEIGHT, CITY )
VALUES ( 'P8', 'Sprocket', 'Pink', 14, 'Nice' ) ;
```

Multi-row INSERT.

```
INSERT
INTO    TEMP ( S#, CITY )
        SELECT S.S#, S.CITY
        FROM    S
        WHERE   S.STATUS > 15 ;
```

Multi-row UPDATE.

```
UPDATE P
SET     COLOR = 'Yellow',
        WEIGHT = P.WEIGHT + 5
WHERE  P.CITY = 'Paris' ;
```

Multi-row UPDATE.

```
UPDATE P
SET     CITY = ( SELECT S.CITY
                  FROM    S
                  WHERE   S.S# = 'S5' )
WHERE  P.COLOR = 'Red' ;
```

Multi-row DELETE.

```
DELETE
FROM    SP
WHERE   'London' =
        ( SELECT S.CITY
          FROM    S
          WHERE   S.S# = SP.S# ) ;
```

5.5 Intégration de SQL à des langages de programmation

```
EXEC SQL DECLARE X CURSOR FOR
        SELECT S.S#, S.SNAME, S.STATUS
        FROM    S
        WHERE   S.CITY = :Y ;

EXEC SQL OPEN X ;                                /* execute the query */
EXEC SQL FETCH X INTO :V1, :V2, :V3 ;           /* fetch the first row (if any) */
WHILE a row is fetched LOOP
    ...                                           /* process the row */
    EXEC SQL FETCH X INTO :V1, :V2, :V3 ; /* fetch the next row (if any) */
END-LOOP
EXEC SQL CLOSE X ;                               /* deactivate cursor X */
```

5.6 Vues

5.6.1 Définition des Vues

Une vue est une table virtuelle dont le schéma et le contenu sont dérivés de la base réelle par un ensemble de requêtes.

```
CREATE VIEW REDPARTS ( P#, PNAME, WT, CITY )
  AS SELECT P.P#, P.PNAME, P.WEIGHT, P.CITY
     FROM   P
     WHERE  P.COLOR = 'Red' ;
```

```
CREATE VIEW PQ
  AS SELECT SP.P#, SUM ( SP.QTY ) AS TOTQTY
     FROM   SP
     GROUP  BY SP.P# ;
```

5.6.2 Interrogation au travers de vues

Get red parts that weigh more than 15 pounds.

```
SELECT P#
FROM   REDPARTS
WHERE  WT > 15 ;
```

⇔

```
SELECT P#
FROM   P
WHERE  WEIGHT > 15
AND    COLOR = 'Red' ;
```

5.6.3 Mise à jour au travers de vues

```
UPDATE REDPARTS
SET    WT = 454 * WT ;
```

⇔

```
UPDATE P
SET    WEIGHT = 454 * WEIGHT
WHERE  COLOR = 'Red' ;
```

```
UPDATE PQ
SET    TOTQTY = TOTQTY + 1 ;
```

⇔

```
UPDATE SP
SET    ???
```

Chapitre 6

Les Dépendances Fonctionnelles

6.1 Introduction

La table suivante enregistre l'horaire du premier quadrimestre. Le premier tuple, par exemple, indique qu'une séance du cours de Physique I aura lieu chaque lundi à 8H15 ; le titulaire de ce cours est le professeur Albert Einstein.

Prof	Cours	Heure
Albert Einstein	Physique I	lundi, 8H15
Albert Einstein	Physique I	mercredi, 10H15
Albert Einstein	Mécanique quantique	lundi, 10H15
Marie Curie	Chimie II	lundi, 8H15

Les deux contraintes suivantes doivent être respectées à chaque instant :

— Chaque cours n'a qu'un seul titulaire.

— Aucun professeur ne peut enseigner deux cours distincts à une même plage horaire.

Par exemple, on ne peut pas déplacer le cours de Physique I du “lundi, 8H15” au “lundi, 10H15”, car ce déplacement engendrerait un conflit horaire pour Albert Einstein.

Ces contraintes seront appelées des dépendances fonctionnelles, et dénotées comme suit :

Cours $\rightarrow$ Prof : À chaque cours ne peut correspondre qu'un seul professeur.

Prof, Heure $\rightarrow$ Cours : À chaque combinaison d'un professeur et une plage horaire ne peut correspondre qu'un seul cours.

Noter qu'en conséquence, à chaque combinaison d'un cours et plage horaire ne peut correspondre qu'un seul professeur : **Cours, Heure $\rightarrow$ Prof**.

6.2 Syntaxe et sémantique

Toutes les définitions dans la suite de ce chapitre sont relatives à un schéma-de-relation $\mathcal{A}$.

Syntaxe Une *dépendance fonctionnelle* (DF) sur $\mathcal{A}$ est une expression $X \rightarrow Y$ avec $X, Y \subseteq \mathcal{A}$.

Sémantique Une relation R sur $\mathcal{A}$ satisfait (ou respecte) la DF $X \rightarrow Y$, dénoté par $R \models X \rightarrow Y$, si pour tout $s, t \in R$, si $s[X] = t[X]$, alors $s[Y] = t[Y]$.¹

Soit Σ un ensemble de DF sur $\mathcal{A}$. Une relation R sur $\mathcal{A}$ satisfait Σ , dénoté par $R \models \Sigma$, si R satisfait chaque DF dans Σ . Une DF $X \rightarrow Y$ sur $\mathcal{A}$ est une conséquence logique de Σ , dénoté par $\Sigma \models X \rightarrow Y$, si pour toute relation R sur $\mathcal{A}$, si $R \models \Sigma$, alors $R \models X \rightarrow Y$.

Noter le double usage du symbole $\models$: (i) $R \models X \rightarrow Y$ signifie que la relation R satisfait $X \rightarrow Y$ (ou que $X \rightarrow Y$ est vrai dans R) ; (ii) $\Sigma \models X \rightarrow Y$ signifie que $X \rightarrow Y$ est une conséquence logique de Σ .

Dans ce syllabus, si l'on écrit $\models$ pour dénoter une conséquence logique, il est sous-entendu que l'on se restreint aux relations (et bases de données) *finies*. L'exemple 12 illustrera que l'implication "finie" peut être différente de l'implication "non restreinte".

Exemple 7. Soit $\mathcal{A} = \{A, B, C\}$. On va démontrer que $A \rightarrow C$ est une conséquence logique de $\{A \rightarrow B, B \rightarrow C\}$. Dans ce but, soit R une relation quelconque qui respecte $A \rightarrow B$ et $B \rightarrow C$. Soit s, t deux tuples quelconques de R tels que $s(A) = t(A)$. Il faut démontrer que $s(C) = t(C)$.

À partir de $s, t \in R$, $s(A) = t(A)$ et $R \models A \rightarrow B$, il est correct de conclure $s(B) = t(B)$. Ensuite, à partir de $s, t \in R$, $s(B) = t(B)$, et $R \models B \rightarrow C$, il est correct de conclure $s(C) = t(C)$. $\square$

Soit Σ_1, Σ_2 deux ensembles de DFs sur $\mathcal{A}$. On écrira $\Sigma_1 \models \Sigma_2$ si pour toute DF $X \rightarrow Y$ dans Σ_2 , on a $\Sigma_1 \models X \rightarrow Y$. On dira que Σ_1 et Σ_2 sont équivalents, dénoté par $\Sigma_1 \equiv \Sigma_2$, si $\Sigma_1 \models \Sigma_2$ et $\Sigma_2 \models \Sigma_1$.

Exemple 8. Les ensembles $\{A \rightarrow B, B \rightarrow C\}$ et $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ sont équivalents. $\square$

Proposition 3 (Theorème de Heath). *Soit R une relation sur $\mathcal{A}$ telle que $R \models X \rightarrow Y$ et $Z = \mathcal{A} \setminus XY$. Alors, $R = \pi_{XY}(R) \bowtie \pi_{XZ}(R)$.*

Démonstration. Il est facile de vérifier que l'inclusion $R \subseteq \pi_{XY}(R) \bowtie \pi_{XZ}(R)$ est vraie pour toute relation R sur U . Pour la démonstration de l'inclusion opposée, soit $t \in \pi_{XY}(R) \bowtie \pi_{XZ}(R)$. Alors $t[XY] \in \pi_{XY}(R)$ et $t[XZ] \in \pi_{XZ}(R)$. Alors, il existe des tuples $t_1, t_2 \in R$ tels que $t_1[XY] = t[XY]$ et $t_2[XZ] = t[XZ]$. Donc, $t_1[X] = t[X]$, $t_2[X] = t[X]$ et $t_1[Y] = t[Y]$. Puisque $t_1[X] = t_2[X]$ et $R \models X \rightarrow Y$, on a $t_1[Y] = t_2[Y]$, donc $t_2[Y] = t[Y]$. À partir de $t_2[Y] = t[Y]$ et $t_2[XZ] = t[XZ]$, on peut conclure $t_2 = t$, donc $t \in R$. $\square$

6.3 Fermeture d'un ensemble d'attributs

On s'intéresse à développer un algorithme qui prend en entrée un ensemble Σ de DF et une DF $X \rightarrow Y$ (les deux sur un même schéma-de-relation $\mathcal{A}$), et qui détermine si, oui ou non, $\Sigma \models X \rightarrow Y$.

La *fermeture* d'un ensemble X d'attributs par rapport à un ensemble Σ de DFs, dénotée par $X^{*,\Sigma}$, est définie comme suit :

$$X^{*,\Sigma} := \{A \in \mathcal{A} \mid \Sigma \models X \rightarrow A\}.$$

1. Noter que contrairement aux deux chapitres précédents, le symbole R dénote ici une relation (i.e., un ensemble de tuples), et non un nom de relation.

Il est facile de vérifier que $\Sigma \models X \rightarrow Y$ si et seulement si $Y \subseteq X^{*,\Sigma}$.

L'algorithme suivant prend en entrée un ensemble Σ de DFs et un ensemble X d'attributs, et retourne $X^{*,\Sigma}$.

Algorithme 1.

Entrée : un ensemble Σ de DFs, un ensemble X d'attributs

Sortie : $X^{,\Sigma}$*

NonUtilisé := Σ

Fermeture := X

répéter aussi longtemps que NonUtilisé change

si $W \rightarrow Z \in \text{NonUtilisé}$ *et* $W \subseteq \text{Fermeture}$

alors i. **NonUtilisé** := **NonUtilisé** $\setminus \{W \rightarrow Z\}$

ii. **Fermeture** := **Fermeture** $\cup Z$

retourner Fermeture

Proposition 4. Avec Σ et X en entrée, l'Algorithme 1 calcule $X^{*,\Sigma}$.

Démonstration. Il est facile de démontrer que l'algorithme se termine. Soit *résultat* le résultat retourné par une exécution de l'algorithme. Il est facile de prouver que *résultat* $\subseteq X^{*,\Sigma}$.

Pour prouver l'inclusion opposée, $X^{*,\Sigma} \subseteq \text{résultat}$, soit A un attribut quelconque tel que $A \notin \text{résultat}$. Il faut montrer que $A \notin X^{*,\Sigma}$, i.e., $\Sigma \not\models X \rightarrow A$. Construisons une relation R comme suit :

attributs de <i>résultat</i>			autres attributs		
A_1	...	A_ℓ	$A_{\ell+1}$	...	A_n
0	...	0	0	...	0
0	...	0	1	...	1

Démontrons que $R \models \Sigma$. Par contradiction, supposons que $R \not\models W \rightarrow Z$ avec $W \rightarrow Z \in \Sigma$. Par notre construction de R , on a $W \subseteq \text{résultat}$ et $Z \not\subseteq \text{résultat}$. D'autre part, puisque l'exécution de l'algorithme 1 s'est terminée, il faut que $Z \subseteq \text{résultat}$, une contradiction.

Puisque $X \subseteq X^{*,\Sigma}$ (initialisation de l'algorithme) et $A \notin \text{résultat}$, on a que $R \not\models X \rightarrow A$.

À partir de $R \models \Sigma$ et $R \not\models X \rightarrow A$, il est correct de conclure $\Sigma \not\models X \rightarrow A$. $\square$

6.4 Autres dépendances

6.4.1 Dépendances de jointure et dépendances multivaluées

Une *dépendance de jointure* (DJ) sur $\mathcal{A}$ est une expression $\bowtie [X_1, X_2, \dots, X_\ell]$ avec $\ell \geq 1$ et $\mathcal{A} = \bigcup_{i=1}^\ell X_i$. Une relation R sur $\mathcal{A}$ satisfait cette DJ si

$$R = \pi_{X_1}(R) \bowtie \pi_{X_2}(R) \bowtie \dots \bowtie \pi_{X_\ell}(R).$$

Une *dépendance multivaluée* (DMV) sur $\mathcal{A}$ est une expression $X \twoheadrightarrow Y$ avec $X, Y \subseteq \mathcal{A}$. Une relation R sur $\mathcal{A}$ satisfait cette DMV si $R \models \bowtie [XY, XZ]$ avec $Z = \mathcal{A} \setminus XY$. Une DMV est donc un cas spécial d'une DJ. Dans l'autre direction, on peut vérifier qu'une DJ $\bowtie [X_1, X_2]$ avec deux composants est équivalente à $X_1 \cap X_2 \twoheadrightarrow X_1$ (ce qui est équivalent à $X_1 \cap X_2 \twoheadrightarrow X_1 \setminus X_2$).

Exemple 9. Soit R la relation suivante.

R	A	B	C	D
1	b	c	α	
1	b	c	β	
2	b	c	α	
2	b	c	β	
3	b	d	β	
3	b	d	γ	

Les projections ci-après permettent de vérifier que $R = \pi_{ABC}(R) \bowtie \pi_{BCD}(R)$, donc $R \models \bowtie [ABC, BCD]$. Noter que la DJ $\bowtie [ABC, BCD]$ est équivalente à la DMV $BC \twoheadrightarrow A$ (ou $BC \twoheadrightarrow D$).

$\pi_{ABC}(R)$	A	B	C	$\pi_{BCD}(R)$	B	C	D
	1	b	c		b	c	α
	2	b	c		b	c	β
	3	b	d		b	d	β
					b	d	γ

Par contre, $R \not\models \bowtie [AB, BC, CD]$. En effet, $\pi_{AB}(R) \bowtie \pi_{BC}(R) \bowtie \pi_{CD}(R)$ contient le tuple $\{(A, 1), (B, b), (C, d), (D, \gamma)\}$; ce tuple n'appartient pas à R . $\square$

En utilisant la notion de DJ, le théorème de Heath (Proposition 3) peut être formulé de façon plus succincte :

$$\{X \rightarrow Y\} \models \bowtie [XY, XZ] \text{ avec } Z = \mathcal{A} \setminus XY. \quad (\text{Heath})$$

Ou de façon encore plus succincte :

$$\{X \rightarrow Y\} \models X \twoheadrightarrow Y. \quad (\text{Heath})$$

Les deux propositions suivantes expriment des interactions entre les DFs et DMVs.

Proposition 5. Soit $\mathcal{A}$ un ensemble d'attributs, et soit $\{X, Y, Z\}$ une partition de $\mathcal{A}$. Soit Σ un ensemble de DF sur $\mathcal{A}$. Alors $\Sigma \models \bowtie [XY, XZ]$ si et seulement si $\Sigma \models X \rightarrow Y$ ou $\Sigma \models X \rightarrow Z$.

Démonstration. $\boxed{\Leftarrow}$ Conséquence de la Proposition 3. $\boxed{\Rightarrow}$ Par contraposition. Supposons $\Sigma \not\models X \rightarrow Y$ et $\Sigma \not\models X \rightarrow Z$. On peut supposer que $B \in Y$ et $C \in Z$ sont des attributs distincts tels que $\Sigma \not\models X \rightarrow B$ et $\Sigma \not\models X \rightarrow C$.

Construisons une relation R avec deux tuples, t_0 et t_1 , comme suit :

attributs de $X^{*, \Sigma}$					autres attributs			
A_1	$\dots$	A_ℓ	B	C	$A_{\ell+1}$	$\dots$	A_n	
0	$\dots$	0	0	0	0	$\dots$	0	(t_0)
0	$\dots$	0	1	1	1	$\dots$	1	(t_1)

L'étudiant est invité à démontrer comme exercice que $R \models \Sigma$. Dans le paragraphe suivant, on démontre que $\pi_{XY}(R) \bowtie \pi_{XZ}(R)$ contient un tuple t tel que $t(B) \neq t(C)$, donc $R \not\models \bowtie [XY, XZ]$.

Puisque $X \subseteq X^{*, \Sigma}$, $t_0[X] = t_1[X]$. La jointure $\pi_{XY}(R) \bowtie \pi_{XZ}(R)$ contient un tuple t tel que $t[X] = t_0[X] = t_1[X]$, $t[Y] = t_0[Y]$ et $t[Z] = t_1[Z]$. Clairement, $t(B) = 0$ et $t(C) = 1$. $\square$

Proposition 6. Soit $\mathcal{A}$ un ensemble d'attributs, et soit $\{X, Y, Z\}$ une partition de $\mathcal{A}$. Soit $V \subseteq XY$ et $W \subseteq XZ$. Alors, $\{\bowtie [XY, XZ], V \rightarrow W\} \models X \rightarrow W$.

Démonstration. Soit R une relation sur $\mathcal{A}$ telle que $R \models \bowtie [XY, XZ]$ et $R \models V \rightarrow W$. Soient $s, t \in R$ tels que $s[X] = t[X]$, et soit A un attribut quelconque de W . Il suffit de démontrer $s(A) = t(A)$. Puisque $R \models \bowtie [XY, XZ]$, il existe un tuple $u \in R$ tel que

$$\begin{cases} u[X] = s[X] = t[X] \\ u[Y] = s[Y] \\ u[Z] = t[Z] \end{cases}$$

Donc, $u[XY] = s[XY]$. Puisque $V \subseteq XY$, on obtient $u[V] = s[V]$. Puisque $R \models V \rightarrow A$, $u(A) = s(A)$. Deux cas sont possibles :

Cas où $A \in X$. Puisque $s[X] = t[X]$, on a bien $s[A] = t[A]$.

Cas où $A \in Z$. Puisque $u[Z] = t[Z]$, on a bien $u[A] = t[A]$. Donc, $s(A) = t(A)$.

□

Exemple 10. En s'appuyant sur la Proposition 6, on peut affirmer que $\{\bowtie [AB, AC], B \rightarrow C\} \models A \rightarrow C$. Donc, une DF peut être une conséquence logique d'un ensemble Ξ de DFs et DMVs, sans être une conséquence logique de l'ensemble contenant seulement les DFs de Ξ .

Notez cependant qu'une DF $X \rightarrow A$ avec $A \notin X$ ne peut pas être une conséquence logique d'un ensemble Σ contenant seulement des DJs. En fait, soit $\mathcal{A} = \{A_1, \dots, A_n\}$ et soit R la relation qui contient chaque tuple sur $\mathcal{A}$ que l'on peut construire en utilisant les constantes 0 et 1. Par exemple, pour $n = 3$,

R	A_1	A_2	A_3
	0	0	0
	0	0	1
	0	1	0
	0	1	1
	1	0	0
	1	0	1
	1	1	0
	1	1	1

L'étudiant est invité à argumenter qu'une telle relation R satisfait toute DJ sur $\mathcal{A}$, mais ne satisfait pas $X \rightarrow A_i$ avec $A_i \notin X$. □

6.4.2 Dépendances d'inclusion

Soit $\mathcal{R}$ un schéma-de-base-de données. Une *dépendance d'inclusion* (DI) sur $\mathcal{R}$ est une expression

$$R[A_1, \dots, A_k] \subseteq S[B_1, \dots, B_k]$$

où

- $k \geq 1$;
- R et S sont des noms de relation dans $\mathcal{R}$ (il se peut que $R = S$) ;
- $A_1, \dots, A_k$ sont des attributs distincts dans $\text{sorte}(R)$;
- $B_1, \dots, B_k$ sont des attributs distincts dans $\text{sorte}(S)$.

Une base de données $\mathcal{I}$ sur le schéma $\mathcal{R}$ satisfait cette DI si pour tout $r \in R^{\mathcal{I}}$, il existe $s \in S^{\mathcal{I}}$ tel que pour tout $i \in \{1, \dots, k\}$, $r(A_i) = s(B_i)$.

Exemple 11. Soit $\mathcal{I}$ la base de données contenant les relations suivantes :

R	A	B	S	C	D
	1	2		2	1
	2	3		2	3
	3	2			

Cette base de données satisfait les DI suivantes :

$$\begin{aligned} R[B] &\subseteq R[A] \\ S[C, D] &\subseteq R[B, A] \end{aligned}$$

□

L'exemple suivant montre que dans certains raisonnements, on utilise le fait que les bases de données sont finies. Ces raisonnements peuvent être faux si l'on acceptait des relations infinies. Cette observation est une motivation pour la *théorie des modèles finis* [10], une branche de la logique mathématique.

Exemple 12. Soit $\mathcal{R} = \{R\}$ un schéma-de-base-de-données avec $\text{sorte}(R) = \{A, B\}$. Soit $\Sigma = \{A \rightarrow B, R[A] \subseteq R[B]\}$. Dans le paragraphe suivant, on démontre $\Sigma \models R[B] \subseteq R[A]$.

Soit $\mathcal{I}$ une base de données quelconque sur $\mathcal{R}$ qui satisfait Σ . Puisque $\mathcal{I}$ est finie, il existe un nombre n tel que $|R^{\mathcal{I}}| = n$. Puisque $\mathcal{I}$ satisfait $A \rightarrow B$, la colonne A de la relation $R^{\mathcal{I}}$ contient n valeurs distinctes. Puisque $\mathcal{I}$ satisfait $R[A] \subseteq R[B]$, ces n valeurs doivent toutes apparaître dans la colonne B de $R^{\mathcal{I}}$. Puisque le nombre de valeurs distinctes dans la colonne B ne peut pas être supérieur à n , il est correct de conclure $R[B] \subseteq R[A]$.

Noter cependant qu'il existe une relation *infinie* qui satisfait Σ mais qui ne satisfait pas $R[B] \subseteq R[A]$:

R	A	B
	1	0
	2	1
	3	2
	4	3
	$\vdots$	$\vdots$

□

Dans la section 6.3, on a vu qu'il existe un algorithme qui prend en entrée un ensemble Σ de DFs et une DF σ , et qui détermine si $\Sigma \models \sigma$ ou $\Sigma \not\models \sigma$. Il a été prouvé [1, Theorem 9.2.4] qu'un tel algorithme n'existe pas si Σ peut contenir des DI en plus des DFs.

Proposition 7. *Le problème suivant est indécidable : pour une DF σ et un ensemble Σ contenant des DFs et des DI, décider si $\Sigma \models \sigma$.*

Chapitre 7

Les Formes Normales

Critics of normalization usually miss this point ; they claim (quite rightly) that the ideas are all basically common sense, but they typically do not realize that it is a significant achievement to state what “common sense” means in a precise and formal way.

C.J. Date [6, page 309]

7.1 La redondance

Reprenons la table de la section 6.1 dans laquelle on a caché une valeur. À partir des deux premiers tuples et la DF $\text{Cours} \rightarrow \text{Prof}$, il est correct de conclure que la valeur cachée est “Albert Einstein”. La valeur cachée est “prévisible” et donc, en quelque sorte, “redondante” : la table stocke deux fois le fait qu’Albert Einstein est le titulaire du cours de Physique I.

Prof	Cours	Heure
Albert Einstein	Physique I	lundi, 8H15
	Physique I	mercredi, 10H15
Albert Einstein	Mécanique quantique	lundi, 10H15
Marie Curie	Chimie II	lundi, 8H15

Cette redondance complique la maintenance des données. Le jour où Albert Einstein serait remplacé par François Englert comme titulaire du cours de Physique I, il faut faire attention de bien encoder ce remplacement à plusieurs endroits.

Noter que cette redondance émerge dès qu’un cours à lieu à deux ou plusieurs plages horaires, ce qui est possible, car la DF $\text{Cours} \rightarrow \text{Heure}$ n’est **pas** imposée.

7.2 BCNF

Un *schéma-DF* est un couple $(\mathcal{A}, \Sigma)$ avec $\mathcal{A}$ un schéma-de-relation et Σ un ensemble de DFs sur $\mathcal{A}$.

Intuitivement, on dira qu’un schéma-DF $(\mathcal{A}, \Sigma)$ est en BCNF si dans une relation sur $\mathcal{A}$ qui respecte Σ , il ne peut pas y avoir des redondances à cause des dépendances fonctionnelles. La

définition suivante exprime cette intuition de façon formelle.

Un schéma-DF $(\mathcal{A}, \Sigma)$ est en BCNF (*Boyce-Codd Normal Form*) si pour toute DF $X \rightarrow Y \in \Sigma$ telle que $Y \not\subseteq X$, on a $\Sigma \models X \rightarrow \mathcal{A}$.

Exemple 13. Soit $\mathcal{A} = \{\text{Prof}, \text{Cours}, \text{Heure}\}$ avec $\Sigma = \{\text{Cours} \rightarrow \text{Prof}, \{\text{Prof}, \text{Heure}\} \rightarrow \text{Cours}\}$. Ce schéma-DF n'est pas en BCNF car $\text{Cours} \rightarrow \text{Prof}$ appartient à Σ , mais $\Sigma \not\models \text{Cours} \rightarrow \text{Heure}$. $\square$

Exemple 14. Soit $\mathcal{A} = \{\text{Étudiant}, \text{Dép}, \text{Fac}\}$ avec $\Sigma = \{\text{Étudiant} \rightarrow \text{Dép}, \text{Dép} \rightarrow \text{Fac}\}$. Le schéma-DF $(\mathcal{A}, \Sigma)$ n'est pas en BCNF car $\text{Dép} \rightarrow \text{Fac}$ appartient à Σ et $\Sigma \not\models \text{Dép} \rightarrow \text{Étudiant}$. Noter les redondances dans la table suivante : on stocke plusieurs fois que le département d'Info fait partie de la faculté des Sciences.

Étudiant	Dép	Fac
Anne	Info	Sciences
Bert	Info	Sciences
Carl	Info	Sciences
Doris	Math	Sciences
Els	Math	Sciences
Fred	Marketing	FWEG

La solution consiste à décomposer ce schéma-DF en deux schémas-DF, comme suit.

Étudiant	Dép	Dép	Fac
Anne	Info	Info	Sciences
Bert	Info	Math	Sciences
Carl	Info	Marketing	FWEG
Doris	Math		
Els	Math		
Fred	Marketing		

Après décomposition, les redondances ont disparu : on ne stocke qu'une seule fois que le département d'Info fait partie de la faculté des Sciences. $\square$

7.3 Décompositions

La définition suivante formalise la notion de décomposition.

Définition 1. Une *décomposition* du schéma-DF $(\mathcal{A}, \Sigma)$ est un ensemble $\{(\mathcal{A}_1, \Sigma_1), \dots, (\mathcal{A}_\ell, \Sigma_\ell)\}$ tel que

1. $\mathcal{A} = \bigcup_{i=1}^{\ell} \mathcal{A}_i$;
2. $\Sigma \models \bowtie [\mathcal{A}_1, \dots, \mathcal{A}_\ell]$;
3. pour $i \in \{1, \dots, \ell\}$, $\Sigma_i \equiv \{X \rightarrow Y \mid XY \subseteq \mathcal{A}_i, \Sigma \models X \rightarrow Y\}$.

On dira que cette décomposition *préserve les DFs* si $\Sigma \equiv \bigcup_{i=1}^{\ell} \Sigma_i$. Chaque élément $(\mathcal{A}_i, \Sigma_i)$ est appelée un *composant* de la décomposition. $\square$

La deuxième condition dans la Définition 1 est connue comme la propriété de *lossless join* dans la littérature. En français, nous parlerons d'une décomposition *sans perte* ou d'une décomposition qui *préserve le contenu*.

La troisième condition dans la Définition 1 exprime que pour chaque i , l'ensemble Σ_i encode toute DF sur $\mathcal{A}_i$ qui est une conséquence logique de Σ .

Exemple 15. Continuation de l'exemple 14. Soit $\mathcal{A} = \{\text{Étudiant}, \text{Dép}, \text{Fac}\}$ avec $\Sigma = \{\text{Étudiant} \rightarrow \text{Dép}, \text{Dép} \rightarrow \text{Fac}\}$. Sur base de la Proposition 5, on peut conclure :

$$\begin{aligned}\Sigma & \models \bowtie [\{\text{Étudiant}, \text{Dép}\}, \{\text{Étudiant}, \text{Fac}\}], \\ \Sigma & \models \bowtie [\{\text{Étudiant}, \text{Dép}\}, \{\text{Dép}, \text{Fac}\}].\end{aligned}$$

Introduisons trois schémas-DF :

$$\begin{aligned}ED & := (\{\text{Étudiant}, \text{Dép}\}, \{\text{Étudiant} \rightarrow \text{Dép}\}), \\ EF & := (\{\text{Étudiant}, \text{Fac}\}, \{\text{Étudiant} \rightarrow \text{Fac}\}), \\ DF & := (\{\text{Dép}, \text{Fac}\}, \{\text{Dép} \rightarrow \text{Fac}\}).\end{aligned}$$

Alors $\{ED, EF\}$ et $\{ED, DF\}$ sont deux décompositions. La première décomposition “perd” la DF $\text{Dép} \rightarrow \text{Fac}$. La deuxième décomposition préserve les DFs.

Il est à noter, par ailleurs, que $\{EF, DF\}$ n'est pas une décomposition, parce que sur la base de Proposition 5, on peut conclure que $\Sigma \not\models \bowtie [\{\text{Étudiant}, \text{Fac}\}, \{\text{Dép}, \text{Fac}\}]$. $\square$

7.4 Décomposition en BCNF

On dira qu'une décomposition est en BCNF si chaque composant est en BCNF.

Proposition 8. *Chaque schéma-DF a une décomposition en BCNF.*

Démonstration. Supposons que $(\mathcal{A}, \Sigma)$ n'est pas en BCNF. Alors il existe une DF $X \rightarrow A$, avec A un seul attribut, tel que $\Sigma \models X \rightarrow A$ et $\Sigma \not\models X \rightarrow \mathcal{A}$.

Sur base de la Proposition 3, $\Sigma \models \bowtie [XA, XZ]$ avec $Z = \mathcal{A} \setminus XA$. Donc, une décomposition est $\{(XA, \Sigma_1), (XZ, \Sigma_2)\}$ avec Σ_1 et Σ_2 comme spécifiés dans la Définition 1. Noter que $\Sigma_1 \models X \rightarrow A$, mais que la DF $X \rightarrow A$ ne provoque plus de violation de BCNF. Si (XA, Σ_1) et/ou (XZ, Σ_2) ne sont pas en BCNF, on décompose de même façon chaque composant qui n'est pas encore en BCNF. Il est facile de voir que cette procédure aboutira finalement à une décomposition où tous les composants sont en BCNF. $\square$

L'exemple suivant montre qu'il n'existe pas toujours une décomposition en BCNF qui préserve les DFs.

Exemple 16. Continuation de l'exemple 13. Soit $\mathcal{A} = \{\text{Prof}, \text{Cours}, \text{Heure}\}$ avec $\Sigma = \{\text{Cours} \rightarrow \text{Prof}, \{\text{Prof}, \text{Heure}\} \rightarrow \text{Cours}\}$. Ce schéma-DF n'est pas en BCNF ; on y s'attendait : la section 7.1 a illustré que ce schéma-DF n'exclut pas les redondances.

Sur la base de Proposition 5, on peut conclure :

$$\Sigma \models \bowtie [\{\text{Cours}, \text{Prof}\}, \{\text{Cours}, \text{Heure}\}].$$

Introduisons deux schémas-DF :

$$\begin{aligned}CP & := (\{\text{Cours}, \text{Prof}\}, \{\text{Cours} \rightarrow \text{Prof}\}), \\ CH & := (\{\text{Cours}, \text{Heure}\}, \emptyset).\end{aligned}$$

Alors $\{CP, CH\}$ est une décomposition qui “perd” la DF $\{\text{Prof}, \text{Heure}\} \rightarrow \text{Cours}$. Manifestement, puisque la DF $\{\text{Prof}, \text{Heure}\} \rightarrow \text{Cours}$ contient les trois attributs du schéma-DF, il n'existe pas de décomposition en BCNF qui préserve les DFs. $\square$

7.5 3NF

L'exemple 16 nous montre qu'il n'existe pas toujours une décomposition en BCNF qui préserve les DFs. Suite à cette observation, on introduit maintenant la notion de 3NF comme un affaiblissement de BCNF. On commence par la définition de la notion de clé.

Un sous-ensemble K de $\mathcal{A}$ est appelée une *clé* d'un schéma-DF $(\mathcal{A}, \Sigma)$ si K est un ensemble minimal (par rapport à $\subseteq$) tel que pour tout $A \in \mathcal{A}$, $\Sigma \models K \rightarrow A$. Un attribut A est appelé *premier* s'il appartient à une ou plusieurs clés.

Exemple 17. Soit $\mathcal{A} = ABCDE$ et $\Sigma = \{A \rightarrow BCD, BC \rightarrow AD\}$. Les clés sont AE et BCE . Les attributs premiers sont A, B, C et E . $\square$

Un schéma-DF $(\mathcal{A}, \Sigma)$ est en 3NF si pour toute DF $X \rightarrow Y \in \Sigma$ telle que $Y \not\subseteq X$, soit $\Sigma \models X \rightarrow \mathcal{A}$, soit chaque attribut de $Y \setminus X$ est premier (soit les deux).

La différence entre BCNF et 3NF est la phrase “soit chaque attribut de $Y \setminus X$ est premier”. Intuitivement, l'effet de cette phrase peut être aperçu comme suit. Soit $(\mathcal{A}, \Sigma)$ un schéma-DF tel que pour un attribut premier $A \in \mathcal{A}$, on a $\Sigma \models X \rightarrow A$ et $\Sigma \not\models X \rightarrow \mathcal{A}$. Cette situation représente une violation de BCNF, mais n'est pas une infraction contre 3NF. Le paragraphe suivant montre que dans cette situation, une décomposition en BCNF divisera une clé en deux, ce qui met en péril la préservation des DFs. Voir aussi l'exemple 16 où la décomposition $\{CP, CH\}$ divise la clé $\{\text{Prof}, \text{Heure}\}$ en deux.

Puisque A est premier, il existe une clé K telle que $A \in K$. Supposons, par contradiction, que $K \subseteq XA$. À partir de $\Sigma \models X \rightarrow A$ et $\Sigma \models XA \rightarrow \mathcal{A}$ (car XA contient une clé), on dérive $\Sigma \models X \rightarrow \mathcal{A}$, une contradiction. Il est donc correct de conclure $K \not\subseteq XA$, i.e., il existe un attribut $B \in K$ tel que $B \notin XA$. Pour enlever la violation de BCNF, une décomposition en (XA, Σ_1) et (XBZ, Σ_2) s'impose avec $Z = \mathcal{A} \setminus XAB$. Noter que les attributs A et B de la clé K sont séparés dans cette décomposition.

On dira qu'une décomposition est en 3NF si chaque composant est en 3NF. Le résultat suivant est donné sans démonstration.

Proposition 9. *Chaque schéma-DF a une décomposition en 3NF qui préserve les DFs.*

En pratique, un schéma-DF en 3NF, mais pas en BCNF, est acceptable à la condition que le schéma-DF n'ait pas de décomposition en BCNF qui préserve les DFs. C'est le cas pour le schéma-DF de l'exemple 16. L'exemple suivant montre l'existence d'un schéma-DF en 3NF, mais pas en BCNF, qui a une décomposition en BCNF qui préserve les DFs. Dans ce cas, la décomposition en BCNF est préférable.

Exemple 18. Soit $\mathcal{A} = ABCD$ et $\Sigma = \{AC \rightarrow D, CD \rightarrow A\}$. Les clés de $(\mathcal{A}, \Sigma)$ sont ABC et BCD . Puisque chaque attribut est premier, le schéma-DF $(\mathcal{A}, \Sigma)$ est en 3NF. Ce schéma-DF n'est pas en BCNF, car $\Sigma \models AC \rightarrow D$ mais $\Sigma \not\models AC \rightarrow B$.

Par la Proposition 3, $\Sigma \models \bowtie [ACD, ABC]$, ce qui donne lieu à une décomposition dont les deux composants sont comme suit :

$$\begin{aligned} & (ACD, \{AC \rightarrow D, CD \rightarrow A\}) \\ & (ABC, \emptyset) \end{aligned}$$

Cette décomposition est en BCNF (et donc en 3NF) et préserve les DFs. Il est à noter, par ailleurs, que la clé BCD est divisée en deux, mais quand-même préservée, parce que $\{AC \rightarrow D, CD \rightarrow A\} \models BCD \rightarrow ABCD$. $\square$

7.6 Dépendances fonctionnelles singulières

Disons qu'une DF $X \rightarrow Y$ est *singulière* si $|Y| = 1$ et $Y \not\subseteq X$. Il est facile de démontrer que chaque ensemble de DFs est équivalent à un ensemble de DFs qui ne contient que des DFs singulières. Les définitions de BCNF et 3NF peuvent alors être simplifiées comme suit.

Soit $(\mathcal{A}, \Sigma)$ un schéma-DF tel que chaque DF dans Σ est singulière. Alors,

- $(\mathcal{A}, \Sigma)$ est en BCNF si pour tout $X \rightarrow A$ dans Σ , on a $\Sigma \models X \rightarrow \mathcal{A}$.
- $(\mathcal{A}, \Sigma)$ est en 3NF si pour tout $X \rightarrow A$ dans Σ , soit $\Sigma \models X \rightarrow \mathcal{A}$, soit A est premier (soit les deux).

7.7 Décomposition en 3NF

Un schéma-DF $(\mathcal{A}, \Sigma)$ est *irréductible* si les trois conditions suivantes sont respectées :

1. Chaque DF dans Σ est singulière.
2. Pour chaque DF $X \rightarrow A \in \Sigma$, aucun attribut de X n'est superflu. De façon formelle, pour chaque DF $X \rightarrow A \in \Sigma$ et $B \in X$,

$$(\Sigma \setminus \{X \rightarrow A\}) \cup \{(X \setminus B) \rightarrow A\} \not\equiv \Sigma.$$

3. Aucune DF de Σ n'est redondante. De façon formelle, pour chaque DF $X \rightarrow A \in \Sigma$, $(\Sigma \setminus \{X \rightarrow A\}) \not\equiv \Sigma$.

Exemple 19. $\Sigma_1 = \{A \rightarrow C, AB \rightarrow C\}$ n'est pas irréductible, parce que $\{A \rightarrow C\} \equiv \Sigma_1$.

$\Sigma_2 = \{A \rightarrow C, ABC \rightarrow D\}$ n'est pas irréductible, parce que $\{A \rightarrow C, AB \rightarrow D\} \equiv \Sigma_2$. $\square$

Un exercice intéressant (et plutôt facile) est de développer un algorithme qui prend en entrée un schéma-DF $(\mathcal{A}, \Sigma)$, et qui retourne en sortie un schéma-DF $(\mathcal{A}, \Sigma')$ irréductible tel que $\Sigma' \equiv \Sigma$. Noter que la sortie d'un tel algorithme n'est pas déterministe, comme illustré dans l'exemple suivant.

Exemple 20. Soit $\mathcal{A} = ABC$ et $\Sigma = \{A \rightarrow B, B \rightarrow AC, A \rightarrow C\}$. Soient

$$\begin{aligned} \Sigma_1 &= \{A \rightarrow B, B \rightarrow A, A \rightarrow C\}, \\ \Sigma_2 &= \{A \rightarrow B, B \rightarrow A, B \rightarrow C\}. \end{aligned}$$

Alors, $\Sigma \equiv \Sigma_1 \equiv \Sigma_2$. Les schémas-DF (ABC, Σ_1) et (ABC, Σ_2) sont tous les deux irréductibles. $\square$

La Proposition 9 est une conséquence logique de la proposition suivante.

Proposition 10. *Soit $(\mathcal{A}, \Sigma)$ un schéma-DF irréductible avec $\Sigma = \{X_1 \rightarrow A_1, \dots, X_\ell \rightarrow A_\ell\}$. Soit K une clé de $(\mathcal{A}, \Sigma)$. L'ensemble suivant est une décomposition de $(\mathcal{A}, \Sigma)$ en 3NF qui préserve les DFs :*

$$\{(K, \Sigma_0), (X_1 A_1, \Sigma_1), \dots, (X_\ell A_\ell, \Sigma_\ell)\},$$

avec $\Sigma_0, \Sigma_1, \dots, \Sigma_\ell$ comme spécifiés dans la Définition 1.

Démonstration. Notez que chaque attribut dans $\mathcal{A} \setminus \{A_1, \dots, A_\ell\}$ appartiendra à chaque clé de $(\mathcal{A}, \Sigma)$, et appartiendra donc à K . On a donc que $K \cup \bigcup_{j=1}^\ell X_j A_j = \mathcal{A}$.

Il est trivial de voir que la décomposition de la Proposition 10 préserve les DFs.

Soit $i \in \{1, \dots, \ell\}$. Puisque $(\mathcal{A}, \Sigma)$ est irréductible, on a que X_i est une clé de $(X_i A_i, \Sigma_i)$, donc chaque attribut de X_i est premier. Supposons que Σ_i contienne $V \rightarrow W$ avec $W \not\subseteq V$. Supposons $A_i \in W \setminus V$. Puisque $(\mathcal{A}, \Sigma)$ est irréductible, $V = X_i$, donc $\Sigma_i \models V \rightarrow A_i$. Il est maintenant correct de conclure que $(X_i A_i, \Sigma_i)$ est en 3NF.

Prouvons ensuite que (K, Σ_0) est en 3NF. Supposons que Σ_0 contienne $V \rightarrow W$ avec $W \not\subseteq V$. Donc $\Sigma_0 \models V \rightarrow W \setminus V$. Alors $\Sigma \models K \setminus (W \setminus V) \rightarrow \mathcal{A}$, et donc K n'est pas une clé, une contradiction. Il est maintenant correct de conclure que $\Sigma_0 \equiv \emptyset$, donc (K, Σ_0) est en 3NF.

Démontrons finalement que $\Sigma \models \bowtie [K, X_1 A_1, \dots, X_\ell A_\ell]$. Puisque $K^{*, \Sigma} = \mathcal{A}$, on peut supposer, sans perte de généralité, que pour tout $i \in \{1, \dots, \ell\}$, $X_i \subseteq K \cup \bigcup_{j=1}^{i-1} X_j A_j$. Cette numérotation des composants $X_i A_i$ peut être obtenue à partir de l'algorithme 1. Soit R une relation telle que $R \models \Sigma$. Soit $t \in \pi_K(R) \bowtie \pi_{X_1 A_1}(R) \bowtie \dots \bowtie \pi_{X_\ell A_\ell}(R)$. Il existe un tuple $s \in R$ tel que $t[K] = s[K]$. On démontre, par induction sur i , que pour tout $i \in \{0, \dots, \ell\}$,

$$t[K \cup \bigcup_{j=1}^i X_j A_j] = s[K \cup \bigcup_{j=1}^i X_j A_j] \quad (7.1)$$

Pour $i = 0$, l'équation (7.1) est $t[K] = s[K]$. Pour l'induction, $i \rightarrow i + 1$, l'hypothèse est que l'équation (7.1) est vraie pour i . Il existe un tuple $r \in R$ tel que $t[X_{i+1} A_{i+1}] = r[X_{i+1} A_{i+1}]$. Puisque $X_{i+1} \subseteq K \cup \bigcup_{j=1}^i X_j A_j$, en utilisant l'hypothèse d'induction, $t[X_{i+1}] = s[X_{i+1}]$, donc $r[X_{i+1}] = s[X_{i+1}]$. Puisque $R \models X_{i+1} \rightarrow A_{i+1}$, on obtient $r[X_{i+1} A_{i+1}] = s[X_{i+1} A_{i+1}]$, donc $t[X_{i+1} A_{i+1}] = s[X_{i+1} A_{i+1}]$. Il est maintenant correct de conclure que l'équation (7.1) est vraie pour $i + 1$.

Pour $i = \ell$, on obtient $t[K \cup \bigcup_{j=1}^\ell X_j A_j] = s[K \cup \bigcup_{j=1}^\ell X_j A_j]$. Puisque $K \cup \bigcup_{j=1}^\ell X_j A_j = \mathcal{A}$, on a bien $t = s \in R$. $\square$

Exemple 21. Cet exemple montre l'importance du composant (K, Σ_0) dans l'énoncé de la Proposition 10. Soit $\mathcal{A} = \{\text{Étudiant}, \text{Dép}, \text{Langue}\}$. Un tuple $\{(\text{Étudiant}, \text{"Ed"}), (\text{Dép}, \text{"Info"}), (\text{Langue}, \text{"espagnol"})\}$ signifie que Ed est un étudiant au département d'Info qui sait s'exprimer en espagnol. L'ensemble des DFs est le singleton $\Sigma = \{\text{Étudiant} \rightarrow \text{Dép}\}$. En conséquence, $\{\text{Étudiant}, \text{Langue}\}$ est la seule clé. Le schéma-DF $(\mathcal{A}, \Sigma)$ n'est pas en 3NF, parce que Dép n'est pas premier et un étudiant peut maîtriser plusieurs langues, i.e., $\Sigma \not\models \text{Étudiant} \rightarrow \text{Langue}$. L'application de la Proposition 10 nous donne la décomposition suivante en 3NF :

$$\{(\{\text{Étudiant}, \text{Langue}\}, \emptyset), (\{\text{Étudiant}, \text{Dép}\}, \{\text{Étudiant} \rightarrow \text{Dép}\})\}.$$

Noter que $\Sigma \models \bowtie [\{\text{Étudiant, Langue}\}, \{\text{Étudiant, Dép}\}]$ est aussi une conséquence de la Proposition 3.

Le composant $(\{\text{Étudiant, Langue}\}, \emptyset)$ enregistre une association *multiple-sur-multiple* entre étudiants et langues : un étudiant peut maîtriser plusieurs langues, et la même langue peut être maîtrisée par plusieurs étudiants. Le composant $(\{\text{Étudiant, Dép}\}, \{\text{Étudiant} \rightarrow \text{Dép}\})$ enregistre une association *un-sur-multiple* (ou *multiple-sur-un*) entre départements et étudiants : un département peut héberger plusieurs étudiants, mais chaque étudiant n'est associé qu'à un seul département. $\square$

L'exemple suivant montre que la décomposition en 3NF de la Proposition 10 peut ne pas être pratique.

Exemple 22. Prenons les attributs de la figure 2.3 : $\mathcal{A} = \{\text{Plaque, Châssis, Type, Année, Accessoire}\}$ avec

$$\Sigma = \left\{ \begin{array}{l} \text{Plaque} \rightarrow \{\text{Châssis, Type, Année}\}, \\ \text{Châssis} \rightarrow \text{Plaque} \end{array} \right\}.$$

Voici un ensemble Σ' de DFs tel que $(\mathcal{A}, \Sigma')$ est irréductible et $\Sigma' \equiv \Sigma$:

$$\Sigma' = \left\{ \begin{array}{l} \text{Plaque} \rightarrow \text{Châssis}, \\ \text{Châssis} \rightarrow \text{Type}, \\ \text{Plaque} \rightarrow \text{Année}, \\ \text{Châssis} \rightarrow \text{Plaque} \end{array} \right\}.$$

Une clé de $(\mathcal{A}, \Sigma')$ est l'ensemble $\{\text{Plaque, Accessoire}\}$. La Proposition 10 résulte en la décomposition suivante :

$$\left\{ \begin{array}{l} (\{\text{Plaque, Accessoire}\}, \emptyset), \\ (\{\text{Plaque, Châssis}\}, \{\text{Plaque} \rightarrow \text{Châssis}, \text{Châssis} \rightarrow \text{Plaque}\}), \\ (\{\text{Châssis, Type}\}, \{\text{Châssis} \rightarrow \text{Type}\}), \\ (\{\text{Plaque, Année}\}, \{\text{Plaque} \rightarrow \text{Année}\}) \end{array} \right\}.$$

Cette décomposition théorique est clairement moins pratique que la décomposition en deux composants présentée dans l'exemple 5. $\square$

7.8 1NF, 2NF, 4NF, 5NF

7.8.1 1NF et 2NF

En ce qui concerne 1NF et 2NF, essayer de comprendre la note de bas de page [12, page 402] $\odot$:

Yes Virginia, there is a first normal form and there is a second normal form. First normal form merely states that the domain of each attribute is an elementary type, rather than a set or a record structure, [...]. Second normal form is only of historical interest and is mentioned in the exercises.

Les formes normales 4NF et 5NF concernent la situation où un schéma contient non seulement des DFs, mais aussi des DMVs et DJs.

7.8.2 4NF

Un schéma-DF-DMV est un couple $(\mathcal{A}, \Xi)$ avec $\mathcal{A}$ un ensemble d'attributs et Ξ un ensemble qui contient des DFs et DMVs sur $\mathcal{A}$.

Un schéma-DF-DMV $(\mathcal{A}, \Xi)$ est en 4NF si pour tous les ensembles $X, Y \subseteq \mathcal{A}$ tels que $Y \not\subseteq X$ et $XY \neq \mathcal{A}$, si $\Xi \models X \twoheadrightarrow Y$, alors $\Xi \models X \rightarrow \mathcal{A}$.

Notez que si $Y \subseteq X$ ou $XY = \mathcal{A}$, alors $X \twoheadrightarrow Y$ est équivalente à une DJ contenant $\mathcal{A}$ comme composant. Les DJs $\bowtie [X_1, \dots, X_\ell]$ avec $X_i = \mathcal{A}$, pour une valeur de i entre 1 et ℓ , sont exclues dans la définition de 4NF parce que ces DJs sont trivialement satisfaites.

Proposition 11. *Si un schéma-DF-DMV $(\mathcal{A}, \Xi)$ est en 4NF, alors il est en BCNF.*

Démonstration. Preuve par contraposition. Supposons $(\mathcal{A}, \Xi)$ n'est pas en 4NF. Alors il existe $X \subseteq \mathcal{A}$ et $B \in \mathcal{A} \setminus \{X\}$ tel que $\Xi \models X \rightarrow B$ et $\Xi \not\models X \rightarrow \mathcal{A}$. Il est alors clair que $XB \neq \mathcal{A}$. Par la Proposition 3, $\Xi \models X \twoheadrightarrow B$. Puisque $XB \neq \mathcal{A}$ et $B \not\subseteq X$, il est correct de conclure que $\Xi \models X \twoheadrightarrow B$ est une violation de 4NF. $\square$

La proposition suivante capture bien l'esprit de 4NF : un schéma $(\mathcal{A}, \Xi)$ est en 4NF si, à la fois, il est en BCNF et toutes les contraintes de Ξ peuvent être exprimées en n'utilisant que des DFs.

Proposition 12. *Soit $(\mathcal{A}, \Xi)$ un schéma-DF-DMV. Les expressions suivantes sont équivalentes :*

1. $(\mathcal{A}, \Xi)$ est en 4NF.
2. Il existe un schéma-DF $(\mathcal{A}, \Sigma)$ en BCNF tel que $\Sigma \equiv \Xi$.

Démonstration. $[1 \implies 2]$ Supposons que $(\mathcal{A}, \Xi)$ est en 4NF. Soit Σ_{DF} l'ensemble des DFs de Ξ , et soit $\Sigma_{DMV} := \{X \rightarrow Y \mid X \twoheadrightarrow Y \in \Xi, Y \not\subseteq X, XY \neq \mathcal{A}\}$. En s'appuyant sur la Proposition 3, il est facile de démontrer que $\Sigma_{DF} \cup \Sigma_{DMV} \models \Xi$.

Il est évident que $\Xi \models \Sigma_{DF}$. On a $\Xi \models \Sigma_{DMV}$ parce que Ξ est en 4NF.

Donc, $\Xi \equiv \Sigma_{DF} \cup \Sigma_{DMV}$. Le schéma $(\mathcal{A}, \Sigma_{DF} \cup \Sigma_{DMV})$ est en BCNF à cause de la Proposition 11.

$[2 \implies 1]$ Supposons qu'il existe un schéma-DF $(\mathcal{A}, \Sigma)$ en BCNF tel que $\Sigma \equiv \Xi$. Supposons $\Xi \models X \twoheadrightarrow Y$ avec $Y \not\subseteq X$ et $XY \neq \mathcal{A}$. Puisque $\Xi \equiv \Sigma$, on a $\Sigma \models X \twoheadrightarrow Y$. Donc, $\Sigma \models \bowtie [X(Y \setminus X), XZ]$ avec $Z = \mathcal{A} \setminus XY$. Puisque $XY \neq \mathcal{A}$, on a $Z \neq \emptyset$. En s'appuyant sur la Proposition 5, $\Sigma \models X \rightarrow Y \setminus X$ ou $\Sigma \models X \rightarrow Z$. Puisque $(\mathcal{A}, \Sigma)$ est en BCNF, $\Sigma \models X \rightarrow \mathcal{A}$. Puisque $\Xi \equiv \Sigma$, on a bien $\Xi \models X \rightarrow \mathcal{A}$. Il est correct de conclure que $(\mathcal{A}, \Xi)$ est en 4NF. $\square$

Exemple 23. Soit $\mathcal{A} = \{\text{Étudiant}, \text{Langage}, \text{Plat}\}$. Un tuple $\{(\text{Étudiant}, \text{"Ed"}), (\text{Langage}, \text{"Java"}), (\text{Plat}, \text{"spaghetti"})\}$ signifie qu'Ed est un étudiant qui sait programmer en Java et qui sait préparer un spaghetti.

Soit $\Xi = \{\bowtie [\{\text{Étudiant}, \text{Langage}\}, \{\text{Étudiant}, \text{Plat}\}]\}$. Noter que $\Xi \equiv \{\text{Étudiant} \twoheadrightarrow \text{Langage}\}$ et $\Xi \equiv \{\text{Étudiant} \twoheadrightarrow \text{Plat}\}$.

La table suivante satisfait Ξ , mais ne satisfait aucune DF singulière. En conséquence, aucun ensemble de DFs ne peut être équivalent à Ξ , et donc le schéma-DF-DMV $(\mathcal{A}, \Xi)$ n'est pas en 4NF.

Étudiant	Langage	Plat
Ed	Java	spaghetti
Ed	Java	paella
Ed	Python	spaghetti
Ed	Python	paella
Anne	Java	spaghetti
Anne	Java	tiramissu

Il est assez évident que cette relation doit être décomposée en deux relations avec comme schémas-de-relation $\{\text{Étudiant}, \text{Langage}\}$ et $\{\text{Étudiant}, \text{Plat}\}$. Intuitivement, 4NF nous dit de ne jamais stocker, dans une même relation, deux renseignements n'ayant rien à voir l'un avec l'autre. $\square$

7.8.3 5NF

Pour rappel, chaque DMV est une DJ. Cependant, il existe des DJs $\bowtie [X_1, \dots, X_\ell]$ avec $\ell \geq 3$ que l'on ne peut pas exprimer comme DMV (voir exemple 26). La notion de 5NF est une généralisation de 4NF qui prend en compte aussi les DJs qui ne sont pas des DMVs. Il existe plusieurs façons de généraliser 4NF [15]; ici, nous proposons une définition suivant l'esprit de la Proposition 12.

Un schéma-DF-DJ est un couple $(\mathcal{A}, \Xi)$ avec $\mathcal{A}$ un ensemble d'attributs et Ξ un ensemble qui contient des DFs et DJs sur $\mathcal{A}$.

Un schéma-DF-DJ $(\mathcal{A}, \Xi)$ est en 5NF s'il existe un schéma-DF $(\mathcal{A}, \Sigma)$ en BCNF tel que $\Sigma \equiv \Xi$.

Notez qu'une conséquence immédiate de cette définition est que chaque schéma-DF en BCNF est aussi en 5NF.

Exemple 24. Soient $\mathcal{A} = ABCD$ et $\Xi = \{A \rightarrow BCD, \bowtie [AB, AC, AD]\}$. Il est facile de démontrer (cf. Proposition 3) que $\{A \rightarrow BCD\} \models \bowtie [AB, AC, AD]$. Donc, $\Xi \equiv \{A \rightarrow BCD\}$. Il est correct de conclure que le schéma-DF-DJ $(\mathcal{A}, \Xi)$ est en 5NF. $\square$

Exemple 25. Soient $\mathcal{A} = ABC$ et $\Xi = \{AB \rightarrow C, AC \rightarrow B, BC \rightarrow A, \bowtie [AB, AC, BC]\}$. La relation

A	B	C
0	0	1
0	2	0
3	0	0

respecte $\{AB \rightarrow C, AC \rightarrow B, BC \rightarrow A\}$, mais ne respecte pas $\bowtie [AB, AC, BC]$, donc $\{AB \rightarrow C, AC \rightarrow B, BC \rightarrow A\} \not\models \bowtie [AB, AC, BC]$. On peut vérifier qu'aucune DF dans $\{A \rightarrow B, A \rightarrow C, B \rightarrow A, B \rightarrow C, C \rightarrow A, C \rightarrow B\}$ n'est une conséquence logique de Ξ . Par exemple, la relation

A	B	C
0	0	0
0	1	1

respecte Ξ , mais ne respecte pas $A \rightarrow B$. Il est maintenant facile de voir qu'aucun ensemble de DFs n'est équivalent à Ξ . En conclusion, $(\mathcal{A}, \Xi)$ ne peut pas être en 5NF. $\square$

Exemple 26. Exemple issu de [9]. Soit $\mathcal{A} = \{\text{Agent}, \text{Company}, \text{Product}\}$. Un tuple $\{(\text{Agent}, \text{“Smith”}), (\text{Company}, \text{“Ford”}), (\text{Product}, \text{“car”})\}$ signifie que Smith est un agent qui vend des voitures pour la marque Ford.

Soit $\Xi = \{\bowtie [\{\text{Agent}, \text{Company}\}, \{\text{Agent}, \text{Product}\}, \{\text{Company}, \text{Product}\}]\}$, ce qui veut dire, en français, que si un agent a vend un produit quelconque pour une compagnie c , et si cet agent a vend un produit p pour une compagnie quelconque, alors l’agent a vend le produit p pour la compagnie c , pourvu que la compagnie c fabrique le produit p .

La table suivante satisfait Ξ , mais ne satisfait aucune DF singulière. En conséquence, aucun ensemble de DFs ne peut être équivalent à Ξ , et donc le schéma-DF-DJ $(\mathcal{A}, \Xi)$ n’est pas en 5NF.

R	Agent	Company	Product
	Smith	Ford	car
	Smith	Ford	truck
	Smith	GM	car
	Smith	GM	truck
	Jones	Ford	car
	Jones	Ford	truck
	Brown	Ford	car
	Brown	GM	car
	Brown	Toyota	car
	Brown	Toyota	bus

Il est assez évident que cette relation doit être décomposée en trois relations avec comme schémas-de-relation $\{\text{Agent}, \text{Company}\}$, $\{\text{Agent}, \text{Product}\}$ et $\{\text{Company}, \text{Product}\}$: la DJ de Ξ exprime que cette décomposition est sans perte !

R_1	Agent	Company	R_2	Agent	Product	R_3	Company	Product
	Smith	Ford		Smith	car		Ford	car
	Smith	GM		Smith	truck		Ford	truck
	Jones	Ford		Jones	car		GM	car
	Brown	Ford		Jones	truck		GM	truck
	Brown	GM		Brown	car		Toyota	car
	Brown	Toyota		Brown	bus		Toyota	bus

Observer :

- Jones vend des voitures et GM fabrique des voitures, mais Jones ne travaille pas pour GM. Donc, $R \not\models \bowtie [\{\text{Product}, \text{Agent}\}, \{\text{Product}, \text{Company}\}]$.
- Brown travaille pour Ford et Ford fabrique des camions, mais Brown ne vend pas de camions. Donc, $R \not\models \bowtie [\{\text{Company}, \text{Agent}\}, \{\text{Company}, \text{Product}\}]$.
- Brown travaille pour Ford et Brown vend des autobus, mais Ford ne fabrique pas d’autobus. Donc, $R \not\models \bowtie [\{\text{Agent}, \text{Company}\}, \{\text{Agent}, \text{Product}\}]$.

Intuitivement, on peut décomposer R sans perte en trois composants, mais chaque décomposition en deux composants engendre une perte. Il est correct de conclure que la DJ

$$\bowtie [\{\text{Agent}, \text{Company}\}, \{\text{Agent}, \text{Product}\}, \{\text{Company}, \text{Product}\}],$$

avec trois composants, ne peut pas être exprimée comme une ou plusieurs DMVs. $\square$

Chapitre 8

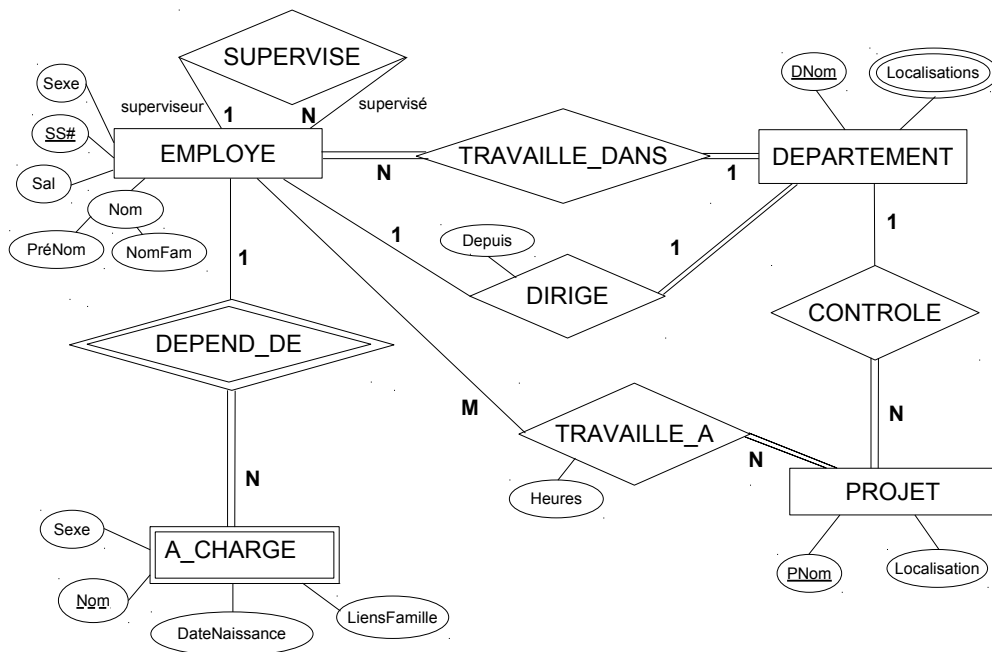
Le Modèle Entité-Association

Le modèle Entity-Relationship (modèle ER, en français : modèle entité-association) a été introduit par P.P. Chen en 1976 [4].

8.1 Énoncé

- L'entreprise est organisée en plusieurs départements. Chaque département possède un nom unique et est dirigé par un seul employé. Il faut mémoriser la date à laquelle l'employé a commencé la direction du département. Un département peut être localisé à plusieurs endroits.
- Un département contrôle un certain nombre de projets. Chaque projet a un nom unique et est localisé à un seul endroit.
- Pour chaque employé, il faut mémoriser le numéro de sécurité sociale, le nom de famille, le prénom, le sexe et le salaire. Un employé est attribué à un seul département mais peut travailler pour plusieurs projets, qui ne sont pas nécessairement contrôlés par son département. On mémorise le nombre d'heures par semaine qu'un employé doit consacrer à chaque projet. Chaque employé est supervisé par un autre.
- On doit connaître les personnes à charge de chaque employé. Pour chaque personne à charge, on stocke le nom, le sexe, la date de naissance et les liens de famille.

8.2 Le diagramme entité-association



8.3 Traduction vers le modèle relationnel

EMPLOYE(SS#, Sexe, Sal, Prénom, NomFam, Travailledans, SuperviséPar)

PK(SS#)

FK(SuperviséPar) REFS EMPLOYE

FK(Travailledans) REFS DEPARTEMENT

DEPARTEMENT(DNom, DirigéPar, Depuis)

PK(DNom)

FK(DirigéPar) REFS EMPLOYE

LOCALISATION(DNom, Loc)

PK(DNom, Loc)

FK(DNom) REFS DEPARTEMENT

PROJET(PNom, Localisation, ContrôléPar)

PK(PNom)

FK(ContrôléPar) REFS DEPARTEMENT

TRAVAILLE_A(SS#, PNom, Heures)

PK(SS#, PNom)

FK(SS#) REFS EMPLOYE

FK(PNom) REFS PROJET

A_CHARGE(Nom, SS#, Sexe, DateNaissance, LiensFamille)
FK(SS#) REFS EMPLOYE

Deuxième partie

Gestion des Transactions

Chapitre 9

Two-Phase Locking (2PL)

[...] two-phase locking (I know Jim Gray won [the Turing award] for many contributions, but this one is, I think, the centerpiece)

J.D. Ullman [14]

9.1 Cadre théorique

Une base de données est une ressource partagée parmi plusieurs transactions, où une transaction est définie comme l'exécution d'un programme informatique qui effectue une "unité logique de travail". Une transaction peut être l'ouverture d'un compte bancaire, un virement, créditer ou débiter un compte. . . L'objectif de la gestion des transactions est d'assurer que ce partage n'engendre pas de conflits.

La gestion de transactions sera étudiée dans le cadre théorique suivant.

1. On suppose que la base de données consiste en un nombre d'objets $A, B, C, \dots$. En pratique, un objet peut prendre différentes formes, telles qu'une valeur dans une table, une rangée dans une table, voire une table entière. . .
2. Les seules actions effectuées sur les objets sont "consulter (ou lire) le contenu d'un objet" et "modifier (ou écrire) le contenu d'un objet". On écrira $R(A)$ et $W(A)$ pour respectivement la lecture (Read) et l'écriture (Write) de l'objet A . Noter que $W(A)$ exprime que l'objet A "subit" une écriture, sans spécifier quelle est la valeur écrite. Notre cadre théorique n'incorpore pas l'insertion ou la suppression d'un objet A .
3. Une *transaction* est caractérisée par une suite d'actions R et W . Les actions autres que R et W sont des actions de "cuisine interne" et ne seront pas montrées. On fera correspondre, à chaque transaction, un indice unique. On écrira $R_i(A)$ et $W_i(A)$ pour respectivement une lecture et une écriture de l'objet A par la transaction [avec indice] i . Un exemple d'une transaction est $R_7(A)R_7(B)W_7(A)W_7(B)$.
4. Une *exécution* (ou un ordonnancement, en anglais : *schedule*) est un ensemble de transactions exécutées "en parallèle", ou de façon plus précise, "en entrelacement". Par exemple, les transactions $T_1 = R_1(A)R_1(B)W_1(A)W_1(B)$ et $T_2 = R_2(A)R_2(B)W_2(C)$ peuvent être entrelacées comme suit :

$$R_1(A)R_1(B)R_2(A)W_1(A)W_1(B)R_2(B)W_2(C). \quad (9.1)$$

Un entrelacement respecte l'ordre des actions à l'intérieur de chaque transaction : si l'on ne retient que les actions avec indice 1, on obtient la transaction T_1 ; si l'on ne retient que les actions avec indice 2, on obtient la transaction T_2 .

Il est important de ne pas confondre un programme informatique avec les transactions qui résultent de son exécution. Les actions R et W ne sont pas des instructions à la disposition du programmeur. Un programme informatique accède typiquement à une base de données à l'aide des instructions SQL (SELECT, UPDATE...), imbriquées dans des boucles et des conditions if-then-else. Une transaction n'est qu'une trace, au niveau interne du SGBD, des événements (lectures et écritures) effectués lors d'une exécution d'un tel programme.

9.2 Les exécutions sérielles

On va argumenter que l'exécution (9.1) risque d'introduire des erreurs dans la base de données. Hypothétiquement, supposons que les lectures et écritures des transactions T_1 et T_2 ont pour but d'effectuer les copiés-collés suivants.

$T_1 = R_1(A)R_1(B)W_1(A)W_1(B)$: T_1 écrit la même valeur pour A et B ; la nouvelle valeur est la concaténation des anciennes valeurs de A et B . Par exemple, si $A = a$ et $B = b$ avant l'exécution de T_1 , alors $A = B = ab$ après l'exécution de T_1 . Noter que la valeur écrite par une transaction dépend typiquement des valeurs lues plus tôt dans la transaction. Dans cet exemple, T_1 doit lire A et B pour connaître la valeur à écrire.

$T_2 = R_2(A)R_2(B)W_2(C)$: T_2 modifie la valeur de C ; la nouvelle valeur de C est la concaténation des anciennes valeurs de A et B .

Pour pouvoir identifier des erreurs, il faut se mettre d'accord sur ce qui est correct. Il est logique d'accepter qu'une exécution est correcte si aucune transaction n'est "interrompue" par une autre. Une telle exécution est appelée *sérielle* (ou *succession*). Par exemple, pour les transactions T_1 et T_2 de notre exemple, il existe deux exécutions sérielles :

$$\begin{aligned} T_1T_2 &= R_1(A)R_1(B)W_1(A)W_1(B)R_2(A)R_2(B)W_2(C), \\ T_2T_1 &= R_2(A)R_2(B)W_2(C)R_1(A)R_1(B)W_1(A)W_1(B). \end{aligned}$$

En général, le nombre d'exécutions sérielles de n transactions est de $n!$ (i.e., factorielle de n).

Pour notre exemple, si $A = a, B = b$ avant l'exécution de T_1 et T_2 , alors $C = abab$ après l'exécution sérielle T_1T_2 , et $C = ab$ après l'exécution sérielle T_2T_1 .

Noter que l'exécution (9.1) résulte en une base de données où $C = aab$: la lecture $R_2(A)$ lit $A = a$, la lecture $R_2(B)$ lit $B = ab$, ce qui est la valeur écrite par $W_1(B)$. L'état de $C = aab$ est considéré comme erroné, car il ne peut pas être produit par une exécution sérielle.

9.3 Les exécutions sérialisables

On définit une relation binaire $\sim$ sur l'ensemble des exécutions. Intuitivement, $\sigma_1 \sim \sigma_2$ exprime que les exécutions σ_1 et σ_2 ont les mêmes effets sur toute base de données.

Premièrement, deux lectures juxtaposées peuvent être permutées :

$$\text{si } i \neq j, \text{ alors } \alpha R_i(X)R_j(Y)\beta \sim \alpha R_j(Y)R_i(X)\beta.$$

Deuxièmement, une lecture et une écriture juxtaposées peuvent être permutées si elles portent sur des objets différents :

$$\text{si } i \neq j \text{ et } X \neq Y, \text{ alors } \alpha R_i(X)W_j(Y)\beta \sim \alpha W_j(Y)R_i(X)\beta.$$

Noter que $R_i(A)W_j(A) \not\sim W_j(A)R_i(A)$, car la valeur lue par la transaction T_i peut être différente selon que la lecture se passe avant ou après l'écriture par T_j .

Troisièmement, deux écritures juxtaposées peuvent être permutées si elles portent sur des objets différents :

si $i \neq j$ et $X \neq Y$, alors $\alpha W_i(X)W_j(Y)\beta \sim \alpha W_j(Y)W_i(X)\beta$.

Finalement, la relation $\sim$ est réflexive, symétrique et transitive, c'est-à-dire, pour toutes exécutions $\sigma, \sigma_1, \sigma_2, \sigma_3$,

$\sigma \sim \sigma$;

si $\sigma_1 \sim \sigma_2$, alors $\sigma_2 \sim \sigma_1$;

si $\sigma_1 \sim \sigma_2$ et $\sigma_2 \sim \sigma_3$, alors $\sigma_1 \sim \sigma_3$.

Une exécution σ est *sérialisable* s'il existe une exécution sérielle σ' telle que $\sigma \sim \sigma'$.

Le *graphe de précedence* d'une exécution σ est un graphe orienté G dont les sommets sont les transactions de σ . Le graphe de précedence contient une arête orientée de T_i à T_j ($i \neq j$) si l'exécution est de la forme $\alpha E_i \beta E_j \gamma$ avec $E_i E_j \not\sim E_j E_i$, où E_i et E_j dénotent des actions (Read ou Write).

La preuve de la proposition suivante est un simple exercice.

Proposition 13. *Une exécution est sérialisable si et seulement si son graphe de précedence est acyclique.*

Par ailleurs, si le graphe de précedence G d'une exécution σ est acyclique, alors pour tout tri topologique $\sigma' = T_{i_1} T_{i_2} \dots T_{i_\ell}$ de G , on a $\sigma \sim \sigma'$.

9.4 L'écriture aveugle

Une écriture $W_i(A)$ dans une transaction est *aveugle* (en anglais : *blind write*) si elle n'est pas précédée par une lecture $R_i(A)$. Par exemple, l'écriture $W_2(C)$ est aveugle dans la transaction $T_2 = R_2(A)R_2(B)W_2(C)$: puisque T_2 ne lit pas C , la nouvelle valeur de C est forcément indépendante de l'ancienne valeur de C .

Une exécution non sérialisable avec des écritures aveugles peut avoir les mêmes effets qu'une exécution sérielle. Par exemple, l'exécution suivante n'est pas sérialisable :

$$R_3(A)W_4(A)W_3(A)W_5(A). \quad (9.2)$$

Les écritures $W_4(A)$ et $W_5(A)$ sont aveugles. On peut argumenter que l'exécution (9.2) a les mêmes effets que l'exécution sérielle $R_3(A)W_3(A)W_4(A)W_5(A)$: la lecture $R_3(A)$ lit la même valeur dans les deux exécutions ; après chacune des deux exécutions, la valeur de A est la valeur écrite par $W_5(A)$, qui ne dépend pas de l'ancienne valeur de A (car l'écriture $W_5(A)$ est aveugle).

Cependant, dans la suite, l'exécution (9.2) sera interdite parce qu'elle n'est pas sérialisable.

9.5 Spécification du protocole 2PL

2PL est un protocole qui garantit la sérialisabilité des exécutions.

9.5.1 Le comportement d'une transaction

Le protocole 2PL fait correspondre, à chaque objet A , plusieurs *verrous partagés* (en anglais : *shared lock* ou *S-lock* sur A) et un seul *verrou exclusif* (en anglais : *exclusive lock* ou *X-lock* sur A). Intuitivement, un S-lock sur un objet A est la permission de lire A ; le X-lock sur A est la permission d'écrire A .

Le protocole 2PL stipule que chaque transaction T_i doit demander (et obtenir) le X-lock sur A pour pouvoir écrire A ; cette demande sera dénotée par $X_i(A)$. Puis, chaque transaction T_i doit demander (et obtenir) un S-lock sur A pour pouvoir lire A ; cette demande sera dénotée par $S_i(A)$. Par ailleurs, une transaction qui possède le X-lock sur A peut aussi lire A .

Dans une transaction T_i , la demande $S_i(A)$ peut être suivie de $X_i(A)$. Dans ce cas, la demande $X_i(A)$ est appelée un *lock upgrade*.

Les transactions doivent relâcher (en anglais : *unlock*) leurs verrous avant de se terminer. On dénote par $U_i(A)$ que la transaction T_i relâche son verrou sur A . Après $S_i(A)$, le relâchement $U_i(A)$ porte sur un S-lock ; après $X_i(A)$, le relâchement $U_i(A)$ porte sur le X-lock sur A .

La terminaison avec succès d'une transaction T_i est appelée Commit et sera dénotée par l'action C_i .

On dira qu'une transaction T_i possède un S-lock sur un objet A à partir de (l'acceptation de) la demande $S_i(A)$ jusqu'au relâchement $U_i(A)$. On dira qu'une transaction T_i possède le X-lock sur A à partir de $X_i(A)$ jusqu'au $U_i(A)$.

En plus de ce qui précède, le protocole 2PL stipule que chaque transaction T_i doit s'exécuter "en deux phases" :

2PL-transaction : Une lecture $R_i(A)$ n'est permise que lorsque la transaction T_i possède un S-lock ou le X-lock sur A . Une écriture $W_i(A)$ n'est permise que lorsque la transaction T_i possède le X-lock sur A . Tout verrou doit être relâché avant le Commit C_i . De plus, aucune demande S_i ou X_i ne peut être effectuée après un relâchement U_i , quel que soit l'objet concerné par ce relâchement.

La première occurrence de U_i dans une transaction T_i est appelé le *lock point* de cette transaction. Le lock point divise la transaction en deux phases : avant le lock point, aucune action U_i n'est possible (par la définition de lock point) ; après le lock point, aucune action S_i ou X_i n'est possible (à cause de la règle ci-dessus).

Par exemple, la transaction suivante respecte le protocole 2PL

$$S_1(A)S_1(B)R_1(A)R_1(B)X_1(B)U_1(A)W_1(B)U_1(B)C_1,$$

parce que :

- la lecture $R_1(A)$ s'effectue entre $S_1(A)$ et $U_1(A)$;
- la lecture $R_1(B)$ s'effectue entre $S_1(B)$ et $U_1(B)$;
- l'écriture $W_1(B)$ s'effectue entre $X_1(B)$ et $U_1(B)$;
- aucune demande S_1 ou X_1 n'a lieu après le lock point $U_1(A)$; et
- les verrous sur A et B sont relâchés avant le commit C_1 .

9.5.2 L'exécution globale

La règle suivante contrôle les exécutions :

2PL-exécution : Au moment où une transaction possède le X-lock sur un objet A , aucune autre transaction ne peut posséder le X-lock ou un S-lock sur ce même objet A .

Une exécution est *conforme au protocole 2PL* s'il est possible d'ajouter des demandes et relâchements de verrous en respectant les règles 2PL-transaction et 2PL-exécution.

Exemple 27. Démontrons que l'exécution $W_2(A)R_1(A)R_3(B)W_2(B)R_1(B)$ n'est pas conforme au protocole 2PL. Entre $W_2(A)$ et $R_1(A)$, on doit avoir un relâchement $U_2(A)$ suivi d'une demande $S_1(A)$. Le relâchement $U_2(A)$ doit être précédé par la demande $X_2(B)$, à cause de la règle 2PL-transaction. Mais alors T_2 possède le X-lock sur B au moment où T_3 a besoin d'un S-lock sur B (pour pouvoir effectuer $R_3(B)$), ce qui est une infraction contre la règle 2PL-exécution. $\square$

Lemme 1. Soit σ une exécution qui est conforme au protocole 2PL. Si le graphe de précédence de σ contient une arête orientée de T_i à T_j ($i \neq j$), alors le lock point de T_i précède le lock point de T_j .

Démonstration. Supposons que le graphe de précédence de σ contient une arête orientée de T_i à T_j . Alors, l'exécution a une des trois formes suivantes :

$$\begin{aligned} & \alpha R_i(A) \beta W_j(A) \gamma \\ & \alpha W_i(A) \beta R_j(A) \gamma \\ & \alpha W_i(A) \beta W_j(A) \gamma \end{aligned}$$

Dans les trois cas, un relâchement $U_i(A)$ doit être ajouté aux actions de β , et ce relâchement doit être suivi par une demande de verrou par T_j . Le lock point de T_i précède ou est égal à $U_i(A)$. A cause de la règle 2PL-transaction, le lock point de T_j se situe après la demande de verrou par T_j . Le lock point de T_i précède donc le lock point de T_j . $\square$

Proposition 14. Si une exécution σ est conforme au protocole 2PL, alors σ est sérialisable.

Démonstration. Preuve par contraposition. Soit σ une exécution qui n'est pas sérialisable. Sur base de la Proposition 13, le graphe de précédence de σ contient un cycle $\langle T_{i_1}, T_{i_2}, T_{i_3}, \dots, T_{i_\ell}, T_{i_1} \rangle$.

On démontrera que σ n'est pas conforme au protocole 2PL.

Nous procédons par contradiction : supposons que σ est conforme au protocole 2PL. Sur base du Lemme 1, il est correct de conclure que le lock point de T_{i_1} précède le lock point de T_{i_2} , que le lock point de T_{i_2} précède le lock point de T_{i_3} . . . En conséquence, le lock point de T_{i_1} précède le lock point de T_{i_1} , une contradiction. $\square$

L'exécution de l'exemple 27 est sérialisable mais n'est pas conforme au protocole 2PL. La réciproque de la Proposition 14 n'est donc pas vraie.

9.6 Implémentation du protocole 2PL

À quoi sert la restriction des exécutions sérialisables aux exécutions qui sont conformes au protocole 2PL ? La réponse est que le protocole 2PL se prête à une implémentation pratique.

Chaque programme doit être encodé de façon à ce que son exécution, en tant que transaction, respecte la règle 2PL-transaction.

Le respect de la règle 2PL-exécution est assuré par le gestionnaire de verrous, qui fait partie du SGBD. Ce gestionnaire mémorise, dans une table, quelles transactions possèdent quels verrous sur quels objets.

Si une transaction T_i effectue $S_i(A)$ au moment où une autre transaction possède le X-lock sur A , alors la transaction T_i est suspendue et mise en attente, jusqu'au moment où le S-lock sur A peut lui être accordé. Pareillement, si une transaction T_i effectue $X_i(A)$ au moment où une autre transaction possède un S-lock ou le X-lock sur A , alors la transaction T_i est suspendue et mise en attente, jusqu'au moment où le X-lock sur A peut lui être accordé. De cette façon, le gestionnaire de verrous assure que la règle 2PL-exécution soit respectée.

En plus, le gestionnaire doit tenir compte des problèmes de la famine (*starvation*) et du verrou mortel (interblocage ou *deadlock*). Le problème de la famine fait référence à la situation où une transaction suspendue reste trop longtemps (voire éternellement) en attente à cause d'autres transactions qui passent devant elle.

La *table de verrouillage* contient, pour chaque objet Y , un couple :

verrous_acquis, file_d'attente,

où

- **verrous_acquis** enregistre les verrous sur l'objet Y en possession des transactions. Cet ensemble contient soit zéro ou plusieurs S-locks, soit un seul X-lock.
- **file_d'attente** est une file qui enregistre les verrous sur Y qui ont été demandés mais qui n'ont pas pu être accordés.

Par exemple, une entrée $(A, \{S_1, S_2\}, \langle X_1, X_3 \rangle)$ enregistre la situation suivante : les transactions T_1 et T_2 possèdent un S-lock sur A ; la transaction T_1 a été suspendue suite à la demande d'un lock-upgrade $X_1(A)$; T_3 a été suspendue suite à la demande $X_3(A)$. Si la transaction T_2 effectue ensuite $U_2(A)$, le lock-upgrade peut être accordé à T_1 et la nouvelle situation devient $(A, \{X_1\}, \langle X_3 \rangle)$.

En général, les demandes S_i et X_i sont traitées comme suit.

- Une demande $S_i(A)$ est acceptée si à la fois (i) l'ensemble **verrous_acquis** pour A dans la table de verrouillage contient zéro, un ou plusieurs S-locks, et (ii) la file d'attente **file_d'attente** est vide. Dans le cas contraire, la demande est mise dans la file d'attente (et la transaction T_i est suspendue). Si l'ensemble **verrous_acquis** ne contient que des S-locks mais la file d'attente n'est pas vide, la demande $S_i(A)$ est quand-même mise en attente afin d'éviter le problème de la famine.
- Une demande $X_i(A)$ est acceptée si l'ensemble **verrous_acquis** pour A est vide $\{\}$ ou le singleton $\{S_i\}$ (lock upgrade). Dans le cas contraire, la demande est mise dans la file d'attente ; si l'ensemble **verrous_acquis** contient d'autres verrous partagés en plus de S_i , alors la demande est insérée à la tête de la file d'attente, afin d'éviter un verrou mortel.

Exemple 28. Supposons que les demandes de verrous arrivent dans l'ordre suivant :

$$S_1(A)S_2(A)X_3(A)X_1(A)S_4(A)U_2(A).$$

La table suivante montre comment l'entrée pour A évolue dans la table de verrouillage.

Après l'exécution de :	verrous_acquis	file_d'attente
$S_1(A)$	$\{S_1\}$	$\langle \rangle$
$S_1(A)S_2(A)$	$\{S_1, S_2\}$	$\langle \rangle$
$S_1(A)S_2(A)X_3(A)$	$\{S_1, S_2\}$	$\langle X_3 \rangle$
$S_1(A)S_2(A)X_3(A)X_1(A)$	$\{S_1, S_2\}$	$\langle X_1, X_3 \rangle$
$S_1(A)S_2(A)X_3(A)X_1(A)S_4(A)$	$\{S_1, S_2\}$	$\langle X_1, X_3, S_4 \rangle$
$S_1(A)S_2(A)X_3(A)X_1(A)S_4(A)U_2(A)$	$\{X_1\}$	$\langle X_3, S_4 \rangle$

Noter qu'une transaction suspendue peut posséder des verrous ; c'est le cas pour T_1 juste après $S_1(A)S_2(A)X_3(A)X_1(A)$. $\square$

L'implémentation du protocole 2PL discutée ci-dessus traite les demandes S , X et U sans s'occuper des lectures et écritures. En fait, une transaction qui possède un X-lock sur un objet peut lire et écrire l'objet de façon "illimitée" ; une transaction qui possède un S-lock sur un objet peut lire l'objet une, deux ou plusieurs fois. La Proposition 14 donne la garantie que les exécutions qui en résultent sont sérialisables.

Noter qu'une transaction ne peut apparaître qu'une seule fois dans la colonne **file_d'attente** de la table de verrouillage : une transaction suspendue ne fait "plus rien" et ne fera donc plus de demandes S ou X , ni de relâchement U .

9.7 Le verrou mortel

Dans l'implémentation du protocole 2PL, une transaction suspendue attend le relâchement d'un verrou par une autre transaction. Ceci peut engendrer une situation de verrou mortel où deux ou plusieurs transactions s'attendent mutuellement. Voici quelques tables de verrouillage qui présentent des verrous mortels.

Dans la table suivante, la transaction T_1 est suspendue et attend le relâchement $U_2(A)$ par T_2 . Ce relâchement n'aura jamais lieu, car la transaction T_2 est suspendue elle-même.

objet	verrous_acquis	file_d'attente
A	$\{S_1, S_2\}$	$\langle X_1, X_2 \rangle$

La table suivante montre un verrou mortel entre trois transactions.

objet	verrous_acquis	file_d'attente
A	$\{S_1\}$	$\langle X_2 \rangle$
B	$\{S_2\}$	$\langle X_3 \rangle$
C	$\{S_3\}$	$\langle X_1 \rangle$

Dans la table suivante, T_2 attendra vainement le relâchement $U_1(A)$ par T_1 . Comme expliqué dans la section 9.6, ce verrou mortel aurait pu être évité si le lock upgrade $X_1(A)$ avait été accordé.

objet	verrous_acquis	file_d'attente
A	$\{S_1\}$	$\langle X_2, X_1 \rangle$

Le *graphe d'attente* (en anglais : *wait-for graph*) est un graphe orienté dont les sommets sont les transactions. Le graphe contient une arête orientée de T_i à T_j si T_i attend le relâchement d'un verrou par T_j ou si T_i se trouve derrière T_j dans une file d'attente.

Le graphe d'attente peut être construit sur base de la table de verrouillage. Un verrou mortel correspond à un cycle orienté dans le graphe d'attente. Pour mettre fin au verrou mortel, le SGBD doit annuler une ou plusieurs transactions qui font partie du cycle.

Une solution alternative consiste à prévenir que les verrous mortels ne se produisent. Supposons que l'indice d'une transaction correspond à l'instant auquel la transaction a commencé. Disons qu'une transaction T_i est plus *ancienne* que toute transaction $T_{i+1}, T_{i+2}, \dots$. Si T_i est plus ancienne que T_j , alors T_j est plus *jeune* que T_i . Deux protocoles sont possibles :

Wait-Die : Une transaction plus jeune préfère toujours de “mourir” (= d'être annulée) que d'attendre [le relâchement d'un verrou par] une transaction plus ancienne. Dans ce protocole, si le graphe d'attente contient une arête de T_i vers T_j , alors $i < j$ (i.e., T_i n'est pas plus jeune que T_j). Il est facile de voir que le graphe d'attente sera toujours acyclique.

Wound-Wait : Une transaction plus ancienne n'attend pas [le relâchement d'un verrou par] une transaction plus jeune : la jeune sera “blessée” (= sera annulée) et ses verrous sont aussitôt “déverrouillés”. Dans ce protocole, si le graphe d'attente contient une arête de T_i vers T_j , alors $i > j$ (i.e., T_i n'est pas plus ancienne que T_j). De nouveau, le graphe d'attente sera toujours acyclique.

Dans ces deux protocoles, c'est toujours la transaction la plus jeune qui est annulée. Une transaction annulée est reprise avec le même indice. Elle devient ainsi plus vieille et finit toujours par passer. De cette façon, le problème de la famine est évité. Les protocoles Wait-Die et Wound-Wait risquent d'engendrer des annulations excessives. Par exemple, supposons que la table de verrouillage est comme suit.

objet	verrous_acquis	file_d'attente
A	$\{S_2\}$	$\langle \rangle$

Dans un système Wait-Die, la transaction T_3 sera annulée quand elle demande $X_3(A)$. Dans un système Wound-Wait, la transaction T_2 sera annulée quand la transaction T_1 demande $X_1(A)$.

9.8 Strict 2PL

Si un verrou mortel se présente, une ou plusieurs transactions doivent être annulées (en anglais : *transaction abort*). L'annulation de transactions peut aussi être causée par d'autres problèmes : erreur de programmation (par exemple, division par zéro), coupure de courant... Il est évident que les modifications effectuées par une transaction annulée doivent être “défaites”. Il s'agit du principe d'*atomicité* : soit toutes les modifications d'une transaction sont effectuées, soit aucune ne l'est.

Considérons, par exemple, la transaction $T_1 = R_1(A)R_1(B)W_1(A)W_1(B)$ de la section 9.2 qui modifie $A = a, B = b$ en $A = B = ab$. Supposons que cette transaction est annulée après $W_1(A)$, mais avant $W_1(B)$. À ce stade, $W_1(A)$ a déjà écrit $A = ab$, mais la valeur de B est toujours $B = b$. Évidemment, l'état $A = ab, B = b$ est incohérent. Dans le processus d'annulation, le SGBD restaurera l'état cohérent $A = a, B = b$ qui existait avant le début de T_1 . Le chapitre 10 expliquera comment cette restauration est réalisée concrètement.

Un autre principe est celui de la *durabilité* : si une transaction a effectué son Commit, ses modifications ne peuvent plus être défaites par la suite.

Atomicité et durabilité sont des principes “de bons sens”. Imaginons que vous faites un virement dans votre système de *home banking*. Il serait inacceptable que le système vous donne le message “Virement effectué avec succès!” et puis défasse le virement “derrière votre dos” (infraction contre la durabilité). Par contre, si le système se plante au milieu du processus de virement, vous vous attendez à ce que votre compte ne soit pas débité (principe d’atomicité).

Malheureusement, il y a des situations où, dans un système 2PL, les principes d’atomicité et de durabilité ne peuvent pas être assurés ensemble. Considérons l’exécution suivante qui est conforme au protocole 2PL. La transaction T_1 est celle décrite ci-dessus. La transaction T_3 change la valeur de A ; la nouvelle valeur dépend de l’ancienne valeur lue par $R_3(A)$.

$$X_1(A)X_1(B)R_1(A)R_1(B)W_1(A)U_1(A)X_3(A)R_3(A)W_3(A)U_3(A)C_3 \not\downarrow W_1(B)U_1(B)C_1$$

Supposons, de nouveau, que la transaction T_1 est annulée après $U_1(A)$ mais avant $W_1(B)$, à l’endroit du symbole $\not\downarrow$. Lors de l’annulation, le SGBD va restaurer la valeur de A qui existait avant le début de T_1 (principe d’atomicité). Cependant, cette restauration défera la modification de T_3 , ce qui est une infraction contre le principe de durabilité, car T_3 a déjà effectué son Commit. Le problème dans cette exécution est que la lecture $R_3(A)$ lit une valeur qui n’est pas “définitive”; une telle lecture est appelée “sale”.

Un problème analogue est que 2PL nécessite des annulations en cascade. Considérez :

$$X_1(A)X_1(B)R_1(A)R_1(B)W_1(A)U_1(A)S_4(A)R_4(A)X_4(B)W_4(B) \not\downarrow W_1(B)U_1(B)C_1$$

La valeur écrite par $W_4(B)$ risque de dépendre de la valeur de A lue par $R_4(A)$, qui est la valeur écrite par $W_1(A)$. Si les modifications de T_1 sont défaites par la suite, les modifications de T_4 doivent être défaites également. On parle d’une annulation en cascade (en anglais : *cascading abort*).

Une lecture $R_i(A)$ est “sale” (en anglais : *dirty read*) si elle se produit après l’action $W_j(A)$ d’une autre transaction T_j ($i \neq j$) qui n’a pas encore terminé avec C_j .

Le protocole Strict 2PL exclut les lectures sales en remplaçant la règle 2PL-transaction par une règle plus sévère :

Strict2PL-transaction En plus de 2PL-transaction, les transactions gardent tous leurs verrous jusqu’au Commit. En fait, le relâchement des verrous peut être considéré comme une partie du Commit.

Dans Strict 2PL, une transaction T_i peut être annulée “en isolation” (i.e., sans tenir compte des autres transactions) parce qu’aucune autre transaction T_j ($i \neq j$) ne peut dépendre des modifications de T_i .

Il est facile de voir que le protocole Strict 2PL n’exclut pas les verrous mortels.

Chapitre 10

Résistance aux Pannes et Reprise

Il existe trois principaux types de pannes.

- La panne de transaction (*transaction failure*). Par exemple, une transaction qui est annulée pour résoudre un verrou mortel.
- La panne du système (*system failure*). La mémoire centrale est perdue. Par exemple, une coupure de courant.
- La panne de mémoire secondaire (*media failure*).

Ce chapitre explique les techniques mises en œuvre pour pallier la panne du système. Ces techniques permettent aussi de résoudre les pannes de transaction.

10.1 Le buffer

Cette section est reprise du cours de bases de données de Philippe Rigaux, disponible à l'URL <http://sys.bdpedia.fr/>.

“La partie de la mémoire principale affectée à un SGBD est appelée mémoires tampon, ou *buffer*. Un buffer est constitué de blocs en mémoire principale, copies des blocs sur le disque. Quand un accès est demandé (limitons-nous aux lectures pour l’instant), deux cas sont possibles :

- la donnée est déjà dans un buffer, et peut donc être servie directement au processeur ;
- sinon il faut d’abord lire un bloc du disque, et le placer en mémoire, pour se ramener au cas précédent.

La demande d’accès est appelée lecture logique : elle ignore si la donnée est présente en mémoire ou non. Le composant du SGBD chargé des entrées/sorties détermine si une lecture physique sur le disque est nécessaire. La lecture physique implique le chargement d’un bloc de la mémoire secondaire vers la mémoire principale.

[...]

Quand la mémoire cache est pleine et qu’un nouveau bloc doit être lu sur le disque, un algorithme de remplacement doit être adopté pour retirer un des blocs de la mémoire et le replacer sur le disque (opérations dite de *flush*). [...] Ce bloc est alors

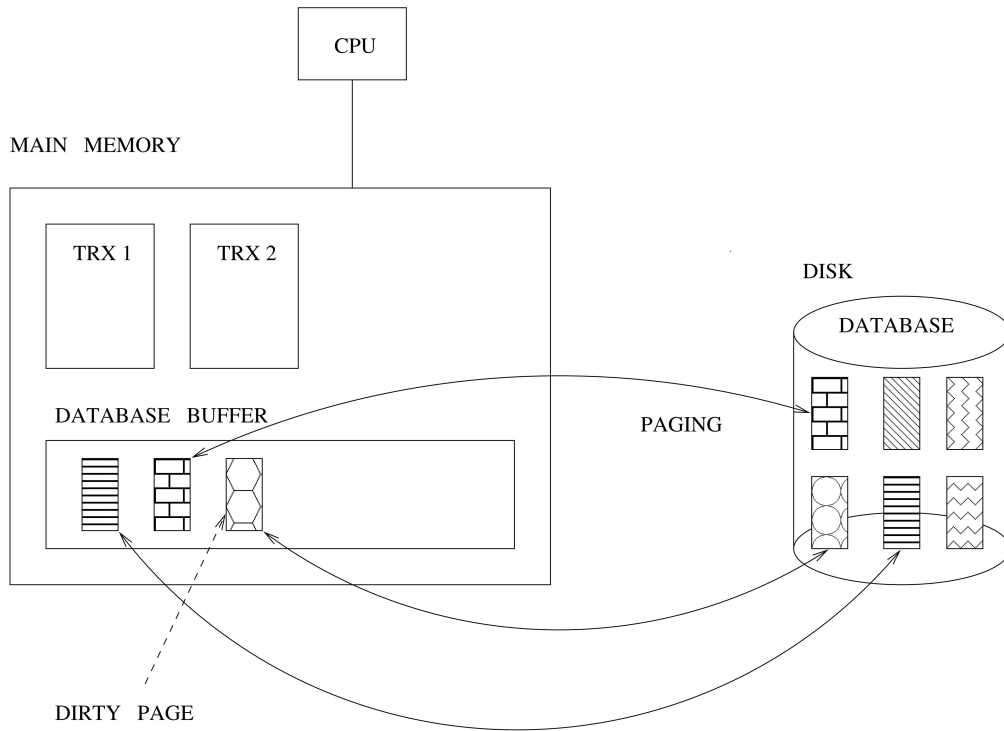


FIGURE 10.1 – Le buffer.

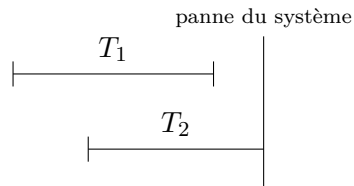


FIGURE 10.2 – La transaction T_1 est à refaire, T_2 à défaire.

soustrait de la mémoire centrale (il reste bien entendu sur le disque) et le nouveau bloc vient le remplacer.”

Les blocs sont aussi appelés *pages*. Voir figure 10.1. Le *page manager* a la responsabilité de (i) charger en mémoire centrale les pages demandées par des transactions, et (ii) réécrire sur disque les pages modifiées par ces mêmes transactions (*dirty pages*).

10.2 Undo/Redo

Supposons un protocole qui interdit à une transaction T_1 de lire un objet modifié par T_2 avant le commit de T_2 (i.e., pas de *dirty reads*). C’est-à-dire, un protocole comme strict 2PL qui ne nécessite pas de *cascading aborts*. Voir figure 10.2. Il faut être capable de :

- “Refaire” (*Redo*) les modifications effectuées par T_1 (principe de la durabilité). Il peut y avoir des pages (1) modifiées en mémoire volatile par T_1 , mais (2) pas encore stockées dans la base au moment de la panne.
- “Défaire” (*Undo*) les modifications effectuées par T_2 (principe de l’atomicité). Il peut y avoir des pages (1) modifiées en mémoire volatile par T_2 , et (2) déjà stockées dans la base

(donc sur disque) au moment de la panne.

10.3 Le journal

La méthode la plus classique pour permettre la validation atomique, l'annulation et la reprise de transactions consiste à utiliser un *journal* (ou *log*). Le journal est un fichier *append-only* gardé sur disque. On enregistre dans le journal l'occurrence des actions suivantes :

- (T, begin) , où T dénote l'identifiant d'une transaction. La transaction T a commencé.
- $(T, A, \text{image_avant}, \text{image_après})$, où A dénote un "objet" de la base de données. La transaction T a modifié A . L'ancienne valeur de A était **image_avant** ; la nouvelle valeur écrite est **image_après**. Dans ce chapitre, supposons que les objets qui constituent l'unité de la résistance aux pannes, sont des pages entières.
- (T, commit) . La transaction T a terminé.

Comme pour tout fichier, ces enregistrements sont d'abord produits en mémoire volatile, puis sauvegardés dans le journal sur disque. Les enregistrements sont ajoutés au journal dans l'ordre qu'ils sont produits. On dit qu'une transaction T est *commise* si (T, commit) se trouve dans le journal sur disque. Ci-après, la terminologie suivante est adoptée :

Sauvegarder : copier sur disque une donnée qui se trouve en mémoire volatile

Stocker, stockage : sauvegarder dans la base de données sur disque une page modifiée en mémoire volatile

Journaliser : sauvegarder dans le journal sur disque

En tout cas, il faut obéir aux règles suivantes :

Journalisation avant Stockage : Avant de stocker une page modifiée en mémoire volatile par une transaction *non-commise*, l'image-avant de cette page doit être journalisée (afin de pouvoir défaire).

Sauvegarde avant Commit : Toute page modifiée en mémoire volatile par une transaction doit être sauvegardée (stockée ou journalisée) avant le commit de la transaction (afin de pouvoir refaire).

10.4 Procédure de Reprise

La procédure de reprise se déroule en trois phases :

Analyse Déterminer quelles transactions sont à défaire/refaire (parcourir le journal en avant).

Défaire Parcourir le journal en arrière et stocker dans la base les images-avant des modifications qui sont à défaire.

Refaire Parcourir le journal en avant et stocker dans la base les images-après des modifications qui sont à refaire.

10.5 Exemple

T_1 dépose 1\$ sur le compte A . T_2 essaie de transférer 1\$ du compte B au compte A .

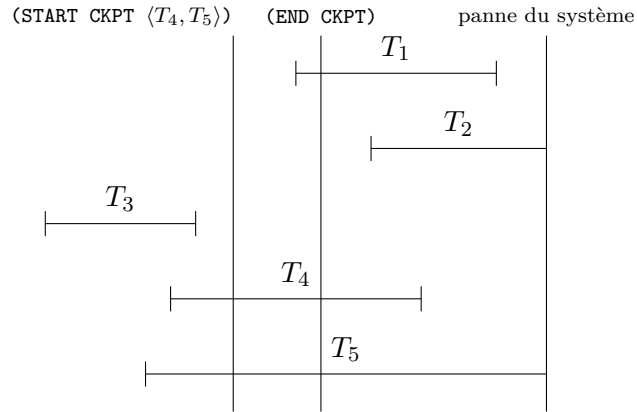


FIGURE 10.3 – Transactions T_1 et T_4 sont à refaire, T_2 et T_5 à défaire.

T_1	T_2	$A = 3, B = 7$	Enregistrements dans le journal
begin			(T_1, begin)
read $A \rightarrow x$		$x = 3$	
$x \leftarrow x + 1$		$x = 4$	
write $x \rightarrow A$		$A = 4$	$(T_1, A, 3, 4)$
commit			(T_1, commit)
	begin		(T_2, begin)
	read $B \rightarrow y$	$y = 7$	
	$y \leftarrow y - 1$	$y = 6$	
	write $y \rightarrow B$	$B = 6$	$(T_2, B, 7, 6)$
	read $A \rightarrow z$	$z = 4$	
	$z \leftarrow z + 1$	$z = 5$	
⚡ panne d'électricité ☹			

Au moment de la panne, l'état de la base est indéfini ; il y a quatre possibilités :

- $A = 3, B = 7$: aucune modification n'a été stockée ;
- $A = 3, B = 6$;
- $A = 4, B = 7$;
- $A = 4, B = 6$: toutes les modifications ont été stockées.

La procédure de reprise va restaurer l'état $A = 4, B = 7$.

10.6 Checkpointing

Comment peut-on réduire le nombre de transactions à refaire lors d'une reprise ? La réalisation d'un *checkpoint* consiste à :

1. journaliser un enregistrement (START CKPT $\langle T_1, \dots, T_k \rangle$), où $T_1, \dots, T_k$ sont les transactions actives ;
2. stocker toutes les pages modifiées dans la base de données sur disque ;
3. journaliser un enregistrement (END CKPT).

Voir figure 10.3 : la transaction T_3 n'intervient pas lors de la reprise après la panne. Une fois que le (END CKPT) apparaît dans le journal, on peut donc supprimer du journal tous les enregistrements concernant les transactions commises avant le START CKPT.

10.7 Undo/No-Redo et Redo/No-Undo

Il est généralement nécessaire de défaire les transaction non-commises et refaire les transactions commises. Cependant, différents protocoles peuvent faciliter la reprise :

Undo/No-Redo Toujours stocker dans la base de données toutes les pages modifiées par une transaction T avant le commit de T . Cette règle est plus sévère que le principe du Sauvegarde avant Commit introduit ci-dessus.

Aucune image-après ne doit être journalisée. Par contre, les images-avant doivent être journalisées (pour pouvoir défaire, principe de la Journalisation avant Stockage). Cette stratégie peut aboutir à des entrées-sorties excessives.

Redo/No-Undo Ne jamais stocker dans la base de données les pages modifiées par une transaction T avant le commit de T . Le principe de la Journalisation avant Stockage est manifestement satisfait.

Aucune image-avant ne doit être journalisée. Par contre, les images-après doivent être journalisées (pour pouvoir refaire, principe de la Sauvegarde avant Commit). Il y a un problème si le buffer n'est pas assez large pour stocker toutes les pages modifiées.

Troisième partie

Exercices

Exercices sur le Chapitre 2

Question 1. On pourrait choisir de remplacer les deux relations VINS et ABUS par une seule relation avec le schéma-de-relation {Nom, Cru, Millésime, Qualité}. Discutez ce choix.

Question 2. Voici deux tables. Ajoutez les clés primaires et étrangères. Motivez votre choix.

Cities	Name	Country	Population
	Bergen	Belgium	20.3
	Bergen	Norway	30.5
	Brussels	Belgium	370.6
	...		

Countries	Name	Capital	Population	Currency
	Belgium	Brussels	10 255.6	EUR
	Norway	Oslo	4 463.2	NOK
	Japan	Tokyo	128 888.0	YEN
	...			

Réponse. Supposons que toutes les villes d'un même pays s'appellent différemment.

```
Cities      ( PRIMARY KEY(Name, Country),
              FOREIGN KEY(Country) REFERENCES Countries );
Countries ( PRIMARY KEY(Name),
              FOREIGN KEY(Capital, Name) REFERENCES Cities,
              UNIQUE(Capital) );
```

- Pourquoi est-il erroné d'écrire FOREIGN KEY(Name, Capital) au lieu de FOREIGN KEY(Capital, Name) ?
- UNIQUE(Capital) indique que la table Countries ne peut pas contenir deux tuples différents avec la même valeur pour Capital.

□

Question 3. Voici trois tables d'une agence spécialisée dans les voyages en autobus. Déterminez les identifiants possibles et choisissez, parmi ceux-ci, les clés primaires. Ajoutez les clés étrangères. Motivez votre choix.

TRIPS	Date	Number_Plate	Driver	Destination	Departure_Time
	15/10/2001	DDT 123	John	Antwerp Zoo	09.00
	15/10/2001	LPG 234	Tim	Ostende Beach	08.00
	16/10/2001	DDT 123	Tim	Dinant Citadel	10.00
	17/10/2001	LPG 234	John	Antwerp Zoo	08.15
	17/10/2001	DDT 123	Tim	Antwerp Zoo	08.15
			...		

BUSES	Number_Plate	Chassis	Make	Mileage
	DDT 123	XGUR6775	Renault	212 342
	LPG 234	ZXRY9823	Mercedes	321 734
	RAM 221	XXZZ7345	Renault	10 000
			...	

DESTINATIONS	Name
	Antwerp Zoo
	Ostende Beach
	Dinant Citadel
	Brussels Atomium
	...

Réponse. Supposons qu'il s'agit d'excursions d'une journée. C'est-à-dire, un bus ou un chauffeur ne fait qu'une excursion par jour.

```

TRIPS      ( PRIMARY KEY(Date, Number_Plate),
             UNIQUE(Date, Driver),
             FOREIGN KEY(Number_Plate) REFERENCES BUSES,
             FOREIGN KEY(Destination) REFERENCES DESTINATIONS );
BUSES      ( PRIMARY KEY(Number_Plate),
             UNIQUE(Chassis) );
DESTINATIONS ( PRIMARY KEY(Name) );

```

□

Question 4. Voici deux tables d'une entreprise qui gère plusieurs dépôts. La première rangée de la table STOCK signifie que 200 charnières jaunes sont stockées au dépôt D1. Ce dépôt se trouve 6, Rue de l'Eglise à Mons. Les quantités sont des entiers positifs (≥ 1). Déterminez les identifiants possibles et choisissez, parmi ceux-ci, les clés primaires. Ajoutez les clés étrangères. Motivez votre choix.

WAREHOUSES	W#	Address	City
	D1	6, Rue de l'Eglise	Mons
	D2	18, Place du Parc	Mons
	D3	18, Place du Parc	Chimay
	D4	5, Avenue Louise	Enghien

STOCK	W#	Product	Color	Qty
	D1	hinge	yellow	200
	D1	hinge	blue	150
	D2	lock	blue	100
	D2	hinge	yellow	200
	D2	handle	red	100
	D4	hinge	red	150
	D4	lock	red	600

Réponse.

```

WAREHOUSES ( PRIMARY KEY(W#),
               UNIQUE(Address, City) );
STOCK      ( PRIMARY KEY(W#, Product, Color),
             FOREIGN KEY(W#) REFERENCES WAREHOUSES );

```

□

Question 5. Le championnat de la Formule 1 se déroule chaque année du mois de mars au mois d'octobre. Dans cette période se déroulent 16 courses, appelées *Grands Prix* (GP), sur 16 circuits différents. Tout pilote est membre d'une équipe, appelée "écurie", pendant toute une année. La table **AFFILIATION** enregistre l'affiliation des pilotes aux écuries par saison. La table **PARTICIPATIONS** enregistre quels pilotes ont participé à quels Grands Prix. La participation à un Grand Prix est réservée aux pilotes affiliés à une écurie. Néanmoins, il se peut qu'un pilote ne participe pas à tous les Grands Prix. Par exemple, en 2001, M. Häkkinen, membre de l'écurie McLaren, n'a pas participé au Grand Prix de Belgique. Finalement, la table **PODIUM** enregistre les trois meilleurs pilotes de chaque Grand Prix ; bien sûr, seuls les participants peuvent gagner.

PODIUM	Année	GP	Vainqueur	Deuxième	Troisième
	2001	Belgique	M. Schumacher	J. Trulli	R. Barrichello
	2003	Espagne	M. Schumacher	F. Alonso	R. Barrichello
	2003	Belgique	G. Fisichella	K. Räikkönen	F. Alonso

AFFILIATION		
Année	Pilote	Écurie
2001	M. Schumacher	Ferrari
2001	R. Barrichello	Ferrari
2001	J. Trulli	Jordan
2001	G. Fisichella	Benetton
2001	M. Häkkinen	McLaren
2003	M. Schumacher	Ferrari
2003	R. Barrichello	Ferrari
2003	J. Trulli	Renault
2003	F. Alonso	Renault
2003	G. Fisichella	Jordan
2003	K. Räikkönen	Jordan

PARTICIPATIONS		
Année	Pilote	GP
2001	M. Schumacher	Belgique
2001	R. Barrichello	Belgique
2001	J. Trulli	Belgique
2001	G. Fisichella	Belgique
2003	M. Schumacher	Espagne
2003	R. Barrichello	Espagne
2003	F. Alonso	Espagne
2003	G. Fisichella	Espagne
2003	M. Schumacher	Belgique
2003	R. Barrichello	Belgique
2003	J. Trulli	Belgique
2003	F. Alonso	Belgique
2003	G. Fisichella	Belgique
2003	K. Räikkönen	Belgique

Déterminez les identifiants possibles et choisissez, parmi ceux-ci, les clés primaires. Ajoutez les clés étrangères. Notez que la base de donnée montrée ci-dessous est cohérente.

Question 6. Considérez les tables suivantes :

EMPLOYÉ	NrEmp	Dept	Pourcent	DÉPARTEMENT	NomDept	Budget	Chef
	E1	Info	40				
	E1	Bio	60				
	E2	Eco	100		Info	5000	E1
	E3	Bio	50		Bio	3500	E1
	E3	Eco	50		Eco	4000	E5
	E4	Eco	100				
	E5	Eco	50				
	E5	Bio	25				
	E5	Info	25				

Les deux premières rangées de la table EMPLOYÉ indiquent que l'employé identifié par le numéro E1 travaille pendant 40% de son temps pour le département Info, et pendant 60% pour le département Bio. Tout employé travaillera à 100% en total. La première rangée de DÉPARTEMENT indique que E1 est le chef du département Info. Ce département dispose d'un budget annuel de 5000 Euros. Tout département aura un seul budget et un seul chef. Un chef travaillera pendant au moins 25% de son temps pour le département dont il est chef. Un employé peut être le chef de plusieurs départements.

Déterminez les identifiants possibles et choisissez, parmi ceux-ci, les clés primaires. Ajoutez les clés étrangères.

Question 7. Considérez deux relations :

EMPLOYÉS	Emp	Dept	Depuis	DÉPARTEMENTS	Dept	Chef
	Ed	Jeux	8 jan 1982			
	Ed	MIS	11 jan 1997		Jeux	An
	An	Jeux	13 sep 2001		MIS	Pierre
	Pierre	MIS	23 oct 1992			
	Pierre	Jeux	11 nov 1995			

Ces tuples expriment, entre autres, que Ed travaille pour le département Jeux depuis le 8 janvier 1982 ; ce département est actuellement dirigé par An. Un employé est associé à un ou plusieurs départements. Un département n'a qu'un seul chef et personne n'est chef de plusieurs départements. En plus, le chef d'un département doit toujours figurer parmi les employés de ce département.

Donnez les clés primaires et étrangères pour ce schéma-de-base-de-données.

Question 8. La table CDA représente un Calendrier de Dates d'Anniversaire. La table MARIAGES stocke les mariages en vigueur ; un homme ou une femme ne peut pas avoir plusieurs conjoints en même temps. On maintient l'anniversaire et l'adresse de tous les mariés. On apprend que Tante Odette est née le 27 juin 1936. Elle est mariée avec Oncle Urbain depuis le 1 mai 1950.

CDA	Nom	Anniversaire	Année	Adresse	Ville
	Tante Odette	27 juin	1936	17 Rue R. Barre	Mons
	Oncle Urbain	27 juin	1927	17 Rue R. Barre	Mons
	Mon chat	17 mars	2001	Chez moi	Enghien
	Jean Bidon	23 mai	1963	36 Rue d'Egmont	Bruxelles
	Anne Lalo	15 mars	1965	35 Rue d'Egmont	Mons
	Jean Crevette	12 janvier	1965	23 Place du Parc	Mons

MARIAGES	Femme	Mari	Jour	Année
	Tante Odette	Oncle Urbain	1 mai	1950
	Anne Lalo	Jean Crevette	14 juillet	1978

Déterminez les identifiants possibles et choisissez, parmi ceux-ci, les clés primaires. Ajoutez les clés étrangères. Motivez votre choix.

Question 9. Pour le problème de la question 85, la prochaine décomposition est proposée. Donnez pour chaque table la clé primaire, les clés étrangères et les contraintes de type UNIQUE.

DATES	Année	Nage	Distance	Sexe	Date	VILLES	Année	Ville
	2004	Papillon	200	M	17/8		2004	Athènes
	2004	Libre	200	M	16/8		2000	Sydney
	2000	Libre	50	F	23/9		:	
	2000	Papillon	200	M	23/9		:	
		:						
NAT	Année	Athlète	Pays			ATHLÈTES	Athlète	Sexe
	2004	M. Phelps	USA				M. Phelps	M
	2004	T. Yamamoto	JAP				T. Yamamoto	M
	2004	S. Parry	GBR				S. Parry	M
	2000	I. de Bruijn	NED				I. de Bruijn	F
	2000	M. Phelps	GBR				:	
	:							
MED	Année	Nage	Distance	Sexe	Athlète	Médaille	Temps	
	2004	Papillon	200	M	M. Phelps	Or	1 :54.04	
	2004	Papillon	200	M	T. Yamamoto	Argent	1 :54.56	
	2004	Papillon	200	M	S. Parry	Bronze	1 :55.52	
	2004	Libre	200	M	M. Phelps	Bronze	1 :45.32	
	2000	Libre	50	F	I. de Bruijn	Or	0 :24.32	
	2000	Papillon	200	M	M. Phelps	Argent	1 :55.19	
				:				

Question 10. La table PROCHES enregistre le sexe et la date de naissance de mes proches ; chaque personne est identifiée de manière unique par son nom. Les mariages sont enregistrés dans la table MARIAGES ; aucune personne n'est mariée plusieurs fois. La table FAVORIS stocke les bières favorites de mes proches, au maximum trois bières par personne. Notez que Tante Odette n'aime aucune bière. Le petit Nicolas est trop jeune pour boire de la bière !

PROCHES	Nom	Sexe	Anniversaire	Année
	Oncle Urbain	M	9 sep	1945
	Caroline	F	28 oct	1960
	Jim	M	24 jan	1971
	An	F	12 sep	1974
	Pierre	M	11 feb	1959
	Tante Odette	F	2 jan	1947
	Nicolas	M	30 août	2015
	Eric	M	11 sep	1981

FAVORIS	Nom	Bière	MARIAGES	Homme	Femme
	Oncle Urbain	Duvel		Oncle Urbain	Tante Odette
	Oncle Urbain	Leffe		Jim	An
	Oncle Urbain	Orval		Pierre	Caroline
	Caroline	Leffe			
	Caroline	Carlsberg			
	Jim	Grimbergen			
	An	Leffe			
	Pierre	Leffe			
	Eric	Chimay			

Déterminez les identifiants possibles et choisissez, parmi ceux-ci, les clés primaires. Ajoutez les clés étrangères.

Requêtes à formuler en algèbre, calcul et SQL

Question 11. Base de données de la question 2.

Quelle est la population de la capitale du Mali ?

Réponse algèbre. Créons d'abord, à l'aide de la requête CC définie ci-dessous, une relation avec attributs **Country** et **Capital**, telle qu'un tuple $\{(\text{Country}, x), (\text{Capital}, y)\}$ signifie que y est la capitale de x :

$$\text{CC} := \rho_{\text{Name} \rightarrow \text{Country}}(\pi_{\text{Name}, \text{Capital}}(\text{Countries})).$$

La requête demandée :

$$\pi_{\text{Population}}((\rho_{\text{Name} \rightarrow \text{Capital}}(\text{Cities})) \bowtie (\sigma_{\text{Country} = \text{"Mali"}}(\text{CC}))).$$

□

Question 12. Base de données de la question 2.

Quelle monnaie est utilisée dans plusieurs pays ?

Réponse algèbre. Introduisons d'abord une abréviation : $\sigma_{A \neq B}(E)$ est une abréviation pour $E - \sigma_{A=B}(E)$, où $A, B \in \text{sorte}(E)$. Définissons la requête **CURRENCIES** comme suit :

$$\text{CURRENCIES} := \pi_{\text{Name}, \text{Currency}}(\text{Countries}).$$

Alors la requête demandée est :

$$\pi_{\text{Currency}}(\sigma_{\text{Country} \neq \text{Name}}(\rho_{\text{Name} \rightarrow \text{Country}}(\text{CURRENCIES}) \bowtie \text{CURRENCIES})).$$

□

Question 13. Base de données de la question 3.

Quel chauffeur a été à toutes les destinations ?

Réponse algèbre. Soit

$$R := (\pi_{\text{Driver}}(\text{TRIPS}) \bowtie \text{DESTINATIONS}) - \rho_{\text{Destination} \rightarrow \text{Name}}(\pi_{\text{Driver}, \text{Destination}}(\text{TRIPS})).$$

Un tuple $\{(Driver, d), (Name, n)\}$ dans R signifie que le chauffeur d n'a jamais fait une excursion à la destination n . Donc, il faut retenir tout chauffeur qui ne figure pas dans R :

$$\pi_{Driver}(TRIPS) - \pi_{Driver}(R).$$

□

Question 14. Base de données de la question 3.

Quelles destinations n'ont jamais été visitées par John ?

Réponse algèbre.

$$DESTINATIONS - \rho_{Destination \mapsto Name}(\pi_{Destination}(\sigma_{Driver="John"}(TRIPS))).$$

□

Question 15. Base de données de la question 3.

Qui a conduit un autobus de la marque Renault pour aller à “Antwerp Zoo” ?

Réponse calcul.

$$\{x \mid \exists y \ (\ \exists u(\exists v(TRIPS(u, y, x, "AntwerpZoo", v))) \wedge \exists w(\exists z(BUSES(y, w, "Renault", z))) \) \}$$

□

Question 16. Base de données de la question 4.

Quels produits sont disponibles en plusieurs couleurs dans un même dépôt ? $\leadsto$ hinge

Réponse calcul.

$$\{y \mid \exists x(\exists z_1(\exists z_2(\exists q_1(\exists q_2(STOCK(x, y, z_1, q_1) \wedge STOCK(x, y, z_2, q_2) \wedge \neg(z_1 = z_2))))))\}$$

□

Question 17. Base de données de la question 4.

Quel entrepôt ne stocke aucun produit rouge ? $\leadsto$ D1 et D3

Réponse calcul. On utilisera $is_a_warehouse(x)$ comme abbréviation pour :

$$\exists v(\exists w(WAREHOUSES(x, v, w))).$$

La requête demandée :

$$\{x \mid \text{is_a_warehouse}(x) \wedge \neg \exists y (\exists q (\text{STOCK}(x, y, \text{"red"}, q)))\} \quad (10.1)$$

Une autre solution est :

$$\{x \mid \text{is_a_warehouse}(x) \wedge \forall y (\forall z (\forall q (\neg \text{STOCK}(x, y, z, q) \vee \neg (z = \text{"red"}))))\} \quad (10.2)$$

□

Question 18. Base de données de la question 4.

Quel entrepôt ne stocke aucun produit non-rouge ? $\leadsto$ D3 et D4

Réponse calcul. Il suffit de remplacer $\neg(z = \text{"red"})$ par $(z = \text{"red"})$ dans la solution (10.2) ci-dessus. □

Question 19. Base de données de la question 5.

Quels pilotes ont donné forfait pour quels Grands Prix ?

Pour la base de données de la question 5, la réponse est comme suit :

Pilote	GP	Année
M. Häkkinen	Belgique	2001
J. Trulli	Espagne	2003
K. Räikkönen	Espagne	2003

Le résultat contient le tuple $\{(\text{Pilote}, \text{M. Häkkinen}), (\text{GP}, \text{Belgique}), (\text{Année}, 2001)\}$ parce qu'en 2001, M. Häkkinen était inscrit au championnat mais ne figure pas parmi les participants au GP de Belgique. Notez que l'absence de M. Häkkinen dans le GP d'Espagne de 2003 ne peut pas être considéré comme un forfait, car M. Häkkinen n'était pas affilié à une écurie en 2003.

Question 20. Base de données de la question 5.

Donnez les Grands Prix (GP et Année) où le vainqueur et le deuxième faisaient partie de la même écurie ? $\leadsto$ seul le GP de Belgique en 2003

Dans l'algèbre, utilisez le moins possible la sélection "attribut égal attribut".

Question 21. Base de données de la question 5.

Donnez les écuries qui n'ont jamais gagné de Grand Prix. $\leadsto$ Benneton, McLaren et Renault ; notez que G. Fisichella n'était plus chez Benneton au moment où il a gagné le GP de Belgique.

Question 22. Base de données de la question 6.

Quel employé partage son temps moitié-moitié (c'est-à-dire 50%-50%) entre deux départements ? $\leadsto$ E3

Question 23. Base de données de la question 6.

Qui sont les chefs de E3 ? $\leadsto$ E1 et E5

Question 24. Bases de données de la question 7.

Donnez tout chef qui ne travaille que pour le département dont il est chef. $\leadsto$ An

Question 25. Base de données de la question 8.

Donnez toute femme mariée qui habite au même endroit que son mari. $\leadsto$ Tante Odette

Question 26. Base de données de la question 8.

Donnez tout individu qui n'est pas marié. $\leadsto$ Mon chat et Jean Bidon

Question 27. Base de données de la question 9.

Donnez les villes qui ont organisé les jeux olympiques plus qu'une fois.

Question 28. Base de données de la question 9.

Donnez les athlètes qui n'ont jamais changé de nationalité.

Question 29. Base de données de la question 10. On dit qu'un mariage est *superbe* si les deux conjoints ont une bière favorite commune. Dans l'exemple, seul le mariage de Pierre et Caroline est superbe.

Donnez les noms des hommes mariés dont le mariage n'est pas superbe. $\leadsto$ Oncle Urbain et Jim

Dans l'algèbre, utilisez le moins possible la sélection "attribut égal attribut".

Question 30. Base de données de la question 10. On dit qu'une personne *A surpasse-en-matière-de-bière* une personne *B* si *A* aime au moins toutes les bières que *B* aime. Par exemple, Oncle Urbain surpasse-en-matière-de-bière An et Pierre. Évidemment, puisque Tante Odette n'aime aucune bière et puisque Nicolas est trop jeune pour boire de la bière, tous mes proches surpassent-en-matière-de-bière Tante Odette et Nicolas.

Donnez toute femme qui surpasse-en-matière-de-bière au moins un homme né en 1985 ou avant. Vous pouvez utiliser l'extension de langage introduction à la question 32. $\leadsto$ Caroline et An

Question 31. Une relation binaire est symétrique si pour tout $(a, b) \in R$, on a $(b, a) \in R$. On utilise un nom de relation R avec $sorte(R) = AB$ pour stocker une relation binaire.

Est-ce que R est symétrique ?

Question 32. Cette question a besoin d'une extension de l'algèbre relationnelle et du calcul relationnel. Soit $(D, \leq)$ un domaine ordonné. Soit A, B deux attributs avec $Dom(A) = Dom(B) = D$. Soit E une expression algébrique telle que $A, B \in sorte(E)$. On peut étendre l'algèbre SPJRUD

avec des sélections qui comparent, en utilisant $<$ ou $\leq$, un attribut à une constante ou à un autre attribut. La sémantique est évidente : pour toute base de données $\mathcal{I}$,

$$\begin{aligned}\llbracket \sigma_{A \leq B}(E) \rrbracket^{\mathcal{I}} &:= \{t \in \llbracket E \rrbracket^{\mathcal{I}} \mid t(A) \leq t(B)\}; \\ \llbracket \sigma_{A \leq "a"}(E) \rrbracket^{\mathcal{I}} &:= \{t \in \llbracket E \rrbracket^{\mathcal{I}} \mid t(A) \leq a\}; \\ \llbracket \sigma_{A < B}(E) \rrbracket^{\mathcal{I}} &:= \{t \in \llbracket E \rrbracket^{\mathcal{I}} \mid t(A) < t(B)\}; \\ \llbracket \sigma_{A < "a"}(E) \rrbracket^{\mathcal{I}} &:= \{t \in \llbracket E \rrbracket^{\mathcal{I}} \mid t(A) < a\}.\end{aligned}$$

De même façon, on peut étendre le calcul relationnel avec des formules atomiques $x \leq y$, $x \leq "a"$, $x < y$, $x < "a"$.

Voici une base de données avec deux tables.

COURS	Nom	Prof	Ects	NOTES	Nom	C	Note
	Génie logiciel	Mens	7		Ed	Génie logiciel	13
	Gestion de projets	Mens	4		Ed	Bases de données	15
	Bases de données	Wijzen	10		Tim	Bases de données	13
	$\vdots$					$\vdots$	

COURS PRIMARY KEY (Nom)
NOTES PRIMARY KEY(Nom, C)
FOREIGN KEY (C) REFS COURS

1. Donnez les cours de Mens qui n'ont pas été suivis par Ed.
2. Quels étudiants ont obtenu une note supérieure à 10 dans un cours enseigné par Mens ?
3. Donnez une relation binaire qui contient une rangée $\langle n_1, n_2 \rangle$ ssi n_1 et n_2 sont deux étudiants qui ont eu le même professeur (pas forcément le même cours).
4. Quels professeurs n'enseignent qu'un seul cours ?
5. Quels professeurs n'ont jamais attribué une note inférieure à 10 ?
6. Donnez la note maximale obtenue en "Bases de Données".
7. Qui a suivi tous les cours de Mens ?
8. Un étudiant est "prodigieux" s'il a obtenu la meilleure note de la classe pour tous les cours qu'il a suivis. Donnez les noms des étudiants prodigieux (s'il y en a).
9. Donnez, pour tout étudiant, le total de ses crédits ECTS (en SQL seulement).
10. Quel professeur donne les notes, en moyenne, les plus basses (en SQL seulement) ?

Question 33. La première ligne de la table RESTAURANTS ci-dessous signifie que le Taco est un restaurant mexicain à Anvers. La première ligne de la table VISITES signifie que Jean a déjà été manger au Naxos. Les contraintes sont :

RESTAURANTS(PRIMARY KEY(Nom));
VISITES(PRIMARY KEY(Nom, Restaurant),
FOREIGN KEY(Restaurant) REFERENCES RESTAURANTS);

RESTAURANTS	Nom	Ville	Type	VISITES	Nom	Restaurant
	Taco	Anvers	mexicain		Jean	Naxos
	Naxos	Anvers	grec		Jean	Trevi
	Trevi	Anvers	italien		Jean	Pronto
	Campos	Mons	italien		Pierre	Naxos
	Pronto	Mons	italien		Pierre	Trevi
					An	Campos
					An	Pronto

Donnez les noms de tous les restaurants pour lesquels aucune visite n'a été enregistrée.
 $\leadsto$ Taco

Question 34. Base de données de la question 33.

Qui a déjà visité deux restaurants italiens dans des villes différentes ? $\leadsto$ Jean

Question 35. Base de données de la question 33.

Qui n'a jamais visité un restaurant non-italien ? $\leadsto$ An

Question 36. Soit $sorte(VINS) = \{Cru, Millésime, Qualité\}$ et $sorte(ABUS) = \{Buveur, Cru, Mill\}$.

Quels sont les vins (cru et millésime) de qualité A qui ont été bus par Jean mais pas par Pierre ?

Réponse algèbre. Soit

$$J := \pi_{Cru, Millésime}(\sigma_{Qualité="A"}(VINS) \bowtie \rho_{Mill \rightarrow Millésime}(\sigma_{Buveur="Jean"}(ABUS))),$$

les vins de qualité A bus par Jean.

Soit

$$P := \pi_{Cru, Millésime}(VINS \bowtie \rho_{Mill \rightarrow Millésime}(\sigma_{Buveur="Pierre"}(ABUS))),$$

les vins bus par Pierre.

La requête est :

$$J - P.$$

□

Question 37. Soit R un nome de relation avec $sorte(R) = \{A, B\}$.

Donnez tout tuple $\{(A, a), (B, b)\}$ de R pour lequel $\{(A, b), (B, a)\}$ fait aussi partie de R .

Par exemple, pour $R \mid \begin{array}{c|cc} & A & B \\ \hline & 1 & 1 \\ & 1 & 2 \\ & 2 & 1 \\ & 1 & 3 \end{array}$, le résultat est $\mid \begin{array}{c|cc} & A & B \\ \hline & 1 & 1 \\ & 1 & 2 \\ & 2 & 1 \end{array}$.

Réponse algèbre. $R \bowtie \rho_{AB \rightarrow BA}(R)$.

□

Question 38. Considérez les tables :

PERSONNE(P#, Nom, Prenom, Adresse)
 PRIMARY KEY(P#)

VEHICULE(V#, Marque, Type)
 PRIMARY KEY(V#)

```

CONDUCTEUR(P#, V#, NInfractions)
    PRIMARY KEY(P#, V#)
    FOREIGN KEY(P#) REFERENCES PERSONNE
    FOREIGN KEY(V#) REFERENCES VEHICULE

```

Un tuple $\langle p, v, n \rangle$ dans la relation **CONDUCTEUR** signifie que n est le nombre d'infractions commises par la personne p au volant de la voiture v . La valeur pour **NInfractions** est un entier positif (≥ 1).

Donnez le nom et le prénom de chaque personne qui a commis des infractions au volant de deux ou plusieurs véhicules *différents*.

Par exemple, pour la bases de données suivante, le résultat se compose du nom et prénom de P1.

CONDUCTEUR	P#	V#	NInfractions
	P1	V1	1
	P1	V2	3
	P1	V3	1
	P2	V1	2

Question 39. Considérez les tables suivantes avec les significations évidentes.

MANGE	Personne	Aliment	NOURRITURE	Aliment	Catégorie
	Anne	pomme		pomme	fruit
	Anne	steak		steak	viande
	Anne	tomate		tomate	légume
	Bill	pomme		orange	fruit
	Bill	tomate		salade	légume
	Ed	pomme		poulet	viande
	Ed	orange		poire	fruit
	Ed	salade			
	Ed	poulet			

Qui est végétarien (c'est-à-dire, qui ne mange pas de viande) ? $\leadsto$ Bill

Question 40. Base de données de la question 39.

Qui mange au moins deux fruits différents ? $\leadsto$ Ed

Question 41. Considérez la base de données avec les tables :

```

FOURNISSEURS(F#, FNom, Ville)
    PRIMARY KEY(F#)

PRODUITS(P#, PNom, Couleur)
    PRIMARY KEY(P#)

STOCK(F#, P#)
    PRIMARY KEY(F#, P#)
    FOREIGN KEY(F#) REFERENCES FOURNISSEURS
    FOREIGN KEY(P#) REFERENCES PRODUITS

```

Les relations FOURNISSEURS et PRODUITS ont les significations évidentes. La relation STOCK contient un tuple $\{(F\#, f), (P\#, p)\}$ si le fournisseur f garde le produit p en stock.

Donnez les numéros des fournisseurs qui gardent à la fois un produit rouge et un produit bleu en stock.

Question 42. Base de données de la question 41.

Donnez les numéros des fournisseurs qui ne gardent aucun produit rouge en stock.

Question 43. Considérez les tables :

S(S#, SNAME, STATUS, CITY)
P(P#, PNAME, COLOR, WEIGHT, CITY)
J(J#, JNAME, CITY)
SPJ(S#, P#, J#, QTY)

1. Donnez les noms des fournisseurs qui fournissent à la fois un produit rouge et un produit vert au même projet à Londres.
2. Donnez les noms des produits qui sont fournis à un ou plusieurs projets en dehors de Londres.

Question 44. Soit R un ensemble fini d'entiers positifs. Pour tout $i, j \in R$, on dira que j est le *successeur* de i si $i < j$ et il n'existe pas de $k \in R$ tel que $i < k < j$.

Donnez tout couple (i, j) tel que $i, j \in R$ et j est le successeur de i . Utilisez les extensions introduites dans la question 32.

Par exemple,

R	A		$q(R)$	A	B
	7			7	8
	8			8	10
	10	$\leadsto$		10	17
	17			17	23
	23				

Question 45. Soit R un nom de relation avec $sorte(R) = ABC$. Écrivez une requête qui résulte en une réponse vide si et seulement si la relation R satisfait la DF $A \rightarrow B$. Le nombre d'attributs du résultat est sans importance.

Réponse calcul.

$$\{x \mid \exists y_1 (\exists y_2 (\exists z_1 (\exists z_2 (R(x, y_1, z_1) \wedge R(x, y_2, z_2) \wedge \neg(y_1 = y_2))))))\}$$

□

Question 46. La table Prix permet de comparer les prix des albums entre différents magasins.

Prix	Album	Artiste	Magasin	Prix
	What's Inside	Joan Armatrading	Proxis.be	18.99
	What's Inside	Joan Armatrading	Free Record Shop	14.99
	What's Inside	Joan Armatrading	Carrefour	10.55
	à Tatons	Axelle Red	Proxis.be	9.99
	à Tatons	Axelle Red	Free Record Shop	8.99

Affichez, pour chaque album, l'endroit [les endroits dans le cas d'un ex æquo] où on peut acheter cet album au meilleur prix. Utilisez les extensions de la questions 32. ∼

Prix	Album	Artiste	Magasin
	What's Inside	Joan Armatrading	Carrefour
	à Tatons	Axelle Red	Free Record Shop

Exercices sur le Chapitre 3

Question 47. On utilise un nom de relation R avec $sorte(R) = AB$ pour stocker un graphe orienté : un tuple $\{(A, a), (B, b)\}$ dénote une arête orientée (ou arc) du sommet a vers le sommet b . Traduisez la requête $\pi_{AB}(\rho_{B \rightarrow C}(R) \bowtie \rho_{A \rightarrow C}(R)) - R$ en français et en calcul relationnel.

Question 48. Soit R, S deux noms de relation avec $sorte(R) = sorte(S)$. Soit $X \subseteq sorte(R)$. Prouvez ou réfutez :

1. $\pi_X(R \cup S) \subseteq \pi_X(R) \cup \pi_X(S)$
2. $\pi_X(R) \cup \pi_X(S) \subseteq \pi_X(R \cup S)$
3. $\pi_X(R \bowtie S) \subseteq \pi_X(R) \bowtie \pi_X(S)$
4. $\pi_X(R) \bowtie \pi_X(S) \subseteq \pi_X(R \bowtie S)$

Question 49. Soit R et S des noms de relation tels que $sorte(R) = X$ et $sorte(S) = Y$. Prouvez ou réfutez :

1. $R \equiv \pi_X(R \bowtie S)$
2. $\pi_X(R \bowtie S) \equiv R \bowtie \pi_{X \cap Y}(S)$

Réponse (partim). Soit $\mathcal{I}$ une base de données telle que $R^{\mathcal{I}} \neq \emptyset$ et $S^{\mathcal{I}} = \emptyset$. Sur la base de $\llbracket \pi_X(R \bowtie S) \rrbracket^{\mathcal{I}} = \emptyset$ et $\llbracket R \rrbracket^{\mathcal{I}} \neq \emptyset$, il est correct de conclure $R \not\equiv \pi_X(R \bowtie S)$. $\square$

Question 50. Soit R un nom de relation. Soient X, Y des ensembles tels que $sorte(R) = X \cup Y$. Prouvez ou réfutez :

1. $\pi_X(R) \bowtie \pi_Y(R) \subseteq R$
2. $R \subseteq \pi_X(R) \bowtie \pi_Y(R)$

Question 51. Pour la base de données

R	A	B
	1	1
	1	2
	2	1
	1	3

, donnez le résultat de :

$$\rho_{C \rightarrow B}(\pi_{AC}(\rho_{A \rightarrow C}(R) \bowtie R)).$$

Réponse.	A	B
	1	1
	1	2
	2	1
	2	2

□

Question 52. Soient R et S des noms de relation avec $\text{sorte}(R) = \{A, B\}$ et $\text{sorte}(S) = \{B, C\}$. Prouvez :

$$\pi_A(R \bowtie S) \subseteq \pi_A(\sigma_{B=C}(R \bowtie (\rho_{B \rightarrow C}(\pi_B(S)))).$$

Réponse. Soit $\mathcal{I}$ une base de données dont le schéma contient R et S . Soit $\{(A, a)\}$ n'importe quel tuple de $\llbracket \pi_A(R \bowtie S) \rrbracket^{\mathcal{I}}$.

Donc, il existe b, c tels que $\{(A, a), (B, b), (C, c)\} \in \llbracket R \bowtie S \rrbracket^{\mathcal{I}}$.

Donc, il existe b, c tels que $\{(A, a), (B, b)\} \in R^{\mathcal{I}}$ et $\{(B, b), (C, c)\} \in S^{\mathcal{I}}$.

Donc, il existe b tel que $\{(A, a), (B, b)\} \in R^{\mathcal{I}}$ et $\{(B, b)\} \in \llbracket \pi_B(S) \rrbracket^{\mathcal{I}}$.

Donc, il existe b tel que $\{(A, a), (B, b)\} \in R^{\mathcal{I}}$ et $\{(C, b)\} \in \llbracket \rho_{B \rightarrow C}(\pi_B(S)) \rrbracket^{\mathcal{I}}$.

Donc, il existe b tel que $\{(A, a), (B, b), (C, b)\} \in \llbracket R \bowtie \rho_{B \rightarrow C}(\pi_B(S)) \rrbracket^{\mathcal{I}}$.

Donc, il existe b tel que $\{(A, a), (B, b), (C, b)\} \in \llbracket \sigma_{B=C}(R \bowtie \rho_{B \rightarrow C}(\pi_B(S))) \rrbracket^{\mathcal{I}}$.

Donc, $\{(A, a)\} \in \llbracket \pi_A(\sigma_{B=C}(R \bowtie (\rho_{B \rightarrow C}(\pi_B(S)))) \rrbracket^{\mathcal{I}}$.

□

Question 53. Soient

$$\begin{aligned} E_1 &:= \pi_{\text{Cru, Qualité}}(\text{VINS}) \bowtie \rho_{\text{Qualité} \rightarrow \text{Qual}}(\pi_{\text{Cru, Qualité}}(\text{VINS})) \\ E_2 &:= \pi_{\text{Cru}}(\text{VINS}) - \pi_{\text{Cru}}(E_1 - \sigma_{\text{Qualité}=\text{Qual}}(E_1)) \end{aligned}$$

Exprimez la requête E_2 en français.

Réponse. Donnez tous les crus qui ont toujours été de la même qualité.

□

Question 54. Soient R et S des noms de relation tels que $\text{sorte}(R) = \text{sorte}(S)$. Démontrez que $R \cup S$ ne peut pas être exprimée en algèbre SPJRD (i.e., l'algèbre sans Union).

Réponse. Pour deux expressions algébriques E_1 et E_2 , on écrira $|E_1| \leq |E_2|$ si pour toute base de données $\mathcal{I}$, $|\llbracket E_1 \rrbracket^{\mathcal{I}}| \leq |\llbracket E_2 \rrbracket^{\mathcal{I}}|$.

Il est facile de prouver que pour toutes expressions E, E_1, E_2 ,

$$|\sigma_{A=B}(E)| \leq |E| \tag{10.3}$$

$$|\sigma_{A=\text{"a"}}(E)| \leq |E| \tag{10.4}$$

$$|\pi_X(E)| \leq |E| \tag{10.5}$$

$$|E_1 \bowtie E_2| \leq |E_1| \times |E_2| \tag{10.6}$$

$$|\rho_{A \rightarrow B}(E)| \leq |E| \tag{10.7}$$

$$|E_1 - E_2| \leq |E_1| \tag{10.8}$$

Soient R et S deux noms de relation tels que $\text{sorte}(R) = \text{sorte}(S) = \{A\}$. Soit $\mathcal{I}$ la base de données suivante :

R	$\left \begin{array}{c} A \\ 0 \end{array} \right.$	S	$\left \begin{array}{c} A \\ 1 \end{array} \right.$
-----	--	-----	--

Évidemment, $\llbracket R \cup S \rrbracket^{\mathcal{I}} = \{\{(A, 0)\}, \{(A, 1)\}\}$, un ensemble avec deux tuples. Soit E une expression quelconque en algèbre SPJRD. On démontre que $|\llbracket E \rrbracket^{\mathcal{I}}| \leq 1$.

Preuve par induction sur la structure de E (i.e., la manière dont l'expression E est composée). La base d'induction est simple : si $E \equiv R$ ou $E \equiv S$, alors $|\llbracket E \rrbracket^{\mathcal{I}}| = 1$. Ensuite, l'étape d'induction :

CAS $E \equiv \sigma_{A=B}(E_1)$. Par hypothèse d'induction, $|\llbracket E_1 \rrbracket^{\mathcal{I}}| \leq 1$. Par (10.3), $|\llbracket E \rrbracket^{\mathcal{I}}| \leq 1$.

CAS $E \equiv \sigma_{A=\text{"b"}}(E_1)$. Analogue.

CAS $E \equiv \pi_X(E_1)$. Analogue.

CAS $E \equiv \rho_{A \rightarrow B}(E_1)$. Analogue.

CAS $E \equiv E_1 \bowtie E_2$. Par hypothèse d'induction, $|\llbracket E_1 \rrbracket^{\mathcal{I}}| \leq 1$ et $|\llbracket E_2 \rrbracket^{\mathcal{I}}| \leq 1$. Par (10.6), $|\llbracket E \rrbracket^{\mathcal{I}}| \leq 1$.

CAS $E \equiv E_1 - E_2$. Par hypothèse d'induction, $|\llbracket E_1 \rrbracket^{\mathcal{I}}| \leq 1$. Par (10.8), $|\llbracket E \rrbracket^{\mathcal{I}}| \leq 1$.

Donc $|\llbracket E \rrbracket^{\mathcal{I}}| \leq 1$. Par conséquent, $E \neq R \cup S$. $\square$

Question 55. Il se fait que l'on peut souvent éviter l'usage de la sélection de la forme "attribut égal attribut". Par exemple, si R, S sont des noms de relation avec $\text{sorte}(R) = \{A\}$ et $\text{sorte}(S) = \{B\}$, alors la requête $\pi_A(\sigma_{A=B}(R \bowtie S))$ donne les tuples de S qui se trouvent aussi dans R . On peut écrire cette même requête sans utiliser $\sigma_{A=B}(\cdot)$, comme suit : $R \bowtie \rho_{B \rightarrow A}(S)$.

Prouvez que l'algèbre SPJRUD perd en expressivité si l'on interdit l'usage de la sélection "attribut égal attribut", tout en permettant la sélection de type "attribut égal constante".

Question 56. Comment peut-on exprimer la requête $\{x \mid \exists y(R(x, \text{"a"}, y) \wedge \neg \exists z(R(x, \text{"b"}, z)))\}$ en algèbre relationnelle ?

Réponse. En supposant $\text{sorte}(R) = ABC$:

$$\pi_A(\sigma_{B=\text{"a"}}R) - \pi_A(\sigma_{B=\text{"b"}}R).$$

$\square$

Exercices sur le Chapitre 4

Question 57. Soit R un nom de relation avec $\text{sorte}(R) = \{A, B\}$. Traduisez la requête

$$\{x \mid \exists w(R(x, w)) \wedge \exists w(\exists v(R(v, w)) \wedge \neg R(x, w))\}$$

en algèbre relationnelle.

Question 58. Traduisez la requête E_2 de la question 53 en calcul relationnel.

Réponse.

$$\{x \mid \exists y(\exists z(\text{VINS}(x, y, z) \wedge \neg(\exists v(\exists w(\text{VINS}(x, v, w) \wedge \neg(z = w))))))\}$$

Si on prend la liberté d'alléger la syntaxe :

$$\{x \mid \exists y, z(\text{VINS}(x, y, z) \wedge \neg \exists v, w(\text{VINS}(x, v, w) \wedge z \neq w))\}$$

En mots simples : donnez tout x tel qu'il existe un tuple $\langle x, y, z \rangle$ en **VINS** pourvu qu'il n'existe aucun autre tuple $\langle x, v, w \rangle$ qui témoigne que le cru x a été d'une qualité w autre que z (donc $w \neq z$). $\square$

Question 59. Soient R et S deux noms de relation d'arité 1. Laquelle des deux requêtes suivantes est erronée, et pourquoi ?

$$\begin{aligned} q_1 &= \{x \mid R(x) \vee \neg S(x)\} \\ q_2 &= \{x \mid \neg(R(x) \vee \neg S(x))\} \end{aligned}$$

Réponse. Si $\varphi(x)$ dénote la formule $R(x) \vee \neg S(x)$, alors $\varphi(a)$ est vraie pour toute constante a telle que $S(a)$ est fausse. Il est évident que la requête q_1 n'est pas indépendante du domaine d'interprétation.

Par contre, si l'on écrit la requête q_2 comme

$$\{x \mid \neg R(x) \wedge S(x)\},$$

il est clair que cette requête est équivalente à la requête $S - R$ de l'algèbre relationnelle. $\square$

Question 60. Traduisez la requête suivante en français :

$$\{x \mid \text{DESTINATIONS}(x) \wedge \neg(\exists u(\exists v(\exists w(\text{TRIPS}(u, v, \text{"John"}, x, w))))))\}$$

La base de données est celle de la question 3.

Question 61. Comment peut-on exprimer la requête $\{x \mid R(x, "a") \wedge \exists y(R("b", y))\}$ en algèbre relationnelle ? Et en SQL ?

Exercices sur le Chapitre 5

Question 62. Pour la base de données de la question 5, donnez une requête en SQL pour la question suivante :

Quel pilote a gagné le plus de Grands Prix? $\leadsto$ M. Schumacher

Question 63. Pour la base de données de la question 8, écrivez une requête en SQL pour répondre à la question :

Quel est le nombre de femmes mariées qui habitent Mons? $\leadsto$ 2

Question 64. Pour la base de données de la question 8, écrivez une requête en SQL pour répondre à la question :

Quel est l'homme marié le plus âgé? $\leadsto$ Oncle Urbain

Question 65. Consider a suppliers-parts-projects database. Suppliers (S) and parts (P) are as in chapter 5. Projects (J) are uniquely identified by a project number (J#). Other attributes of projects are project name (JNAME) and city. The significance of an SPJ (shipment) tuple is that the specified supplier supplies the specified part to the specified project in the specified quantity (and the combination S#-P#-J# uniquely identifies such a tuple). Write a suitable data definition for this database.

Réponse.

```
CREATE TABLE S
( ... ) ;
```

```
CREATE TABLE P
( ... ) ;
```

```
CREATE TABLE J
( J#          J#,
  JNAME       NAME,
  CITY        CITY,
  PRIMARY KEY ( J# ) ) ;
```

```
CREATE TABLE SPJ
( S#          S#,
  P#          P#,
```

```

J#      J#,
QTY     QTY,
PRIMARY KEY ( S#, P#, J# ),
FOREIGN KEY ( S# ) REFERENCES S,
FOREIGN KEY ( P# ) REFERENCES P,
FOREIGN KEY ( J# ) REFERENCES J ) ;

```

□

Question 66. Get part number for parts supplied by a supplier in London.

Réponse.

```

SELECT DISTINCT SPJ.P#
FROM   SPJ, S
WHERE  SPJ.S# = S.S#
AND    S.CITY = 'London' ;

```

□

Question 67. Get project numbers for projects supplied by at least one supplier not in the same city.

Réponse.

```

SELECT DISTINCT SPJ.J#
FROM   SPJ, S, J
WHERE  SPJ.S# = S.S#
AND    SPJ.J# = J.J#
AND    S.CITY <> J.CITY ;

```

□

Question 68. Get part numbers of parts supplied to some project in an average quantity of more than 320.

Réponse.

```

SELECT DISTINCT SPJ.P#
FROM   SPJ
GROUP  BY SPJ.P#, SPJ.J#
HAVING AVG ( SPJ.QTY ) > 320 ;

```

□

Question 69. Get project numbers for projects supplied entirely by S1.

Réponse.

```

SELECT SPJX.J#
FROM   SPJ AS SPJX
WHERE  SPJX.S# = 'S1'
AND    NOT EXISTS
      ( SELECT *
        FROM   SPJ AS SPJY
        WHERE  SPJY.J# = SPJX.J#
        AND    SPJY.S# <> 'S1' ) ;

```

□

Question 70. Get all cities in which at least one supplier, part, or project is located.

Réponse.

```

SELECT CITY
FROM   S
UNION
SELECT CITY
FROM   P
UNION
SELECT CITY
FROM   J ;

```

□

Question 71. Get all pairs of supplier numbers, S_x and S_y say, such that S_x and S_y supply exactly the same set of parts each.

Réponse.

```

SELECT SX.S#, SY.S#
FROM   S AS SX, S AS SY
WHERE  NOT EXISTS
      ( SELECT *
        FROM   SPJ AS SPJX
        WHERE  SPJX.S# = SX.S#
        AND    NOT EXISTS
              ( SELECT *
                FROM   SPJ AS SPJY
                WHERE  SPJY.S# = SY.S#
                AND    SPJY.P# = SPJX.P# ) )
AND    NOT EXISTS
      ( SELECT *
        FROM   SPJ AS SPJY
        WHERE  SPJY.S# = SY.S#
        AND    NOT EXISTS
              ( SELECT *
                FROM   SPJ AS SPJX

```

```
WHERE  SPJX.S# = SX.S#
AND    SPJX.P# = SPJY.P# ) ) ;
```

□

Question 72. Get supplier-number/part-number pairs such that the indicated supplier does not supply the indicated part.

Réponse.

```
SELECT S.S#, P.P#
FROM   S, P
WHERE  NOT EXISTS
      ( SELECT *
        FROM   SPJ
        WHERE  SPJ.S# = S.S#
        AND    SPJ.P# = P.P# ) ;
```

□

Question 73. Get supplier numbers for suppliers supplying some project with part P1 in a quantity greater than the average shipment quantity of part P1 for that project.

Réponse.

```
SELECT SPJX.S#
FROM   SPJ AS SPJX
WHERE  SPJX.P# = 'P1'
AND    SPJX.QTY >
      ( SELECT AVG ( SPJY.QTY )
        FROM   SPJ AS SPJY
        WHERE  SPJY.P# = 'P1'
        AND    SPJY.J# = SPJX.J# ) ;
```

□

Question 74. Get project numbers for projects supplied with part P1 in an average quantity greater than the greatest quantity in which any part is supplied to project J1.

Réponse.

```
SELECT SPJX.J#
FROM   SPJ AS SPJX
WHERE  SPJX.P# = 'P1'
GROUP BY SPJX.J#
HAVING AVG ( SPJX.QTY ) >
      ( SELECT MAX ( SPJY.QTY )
        FROM   SPJ AS SPJY
        WHERE  SPJY.J# = 'J1' ) ;
```

□

Question 75. Pour la base de données de la question 4, écrivez une requête en SQL pour répondre à la question :

Combien de produits rouges sont stockés en dehors de Mons ? $\leadsto$ 750

Quel est le résultat de votre requête si tous les produits en dehors de Mons sont non-rouges ?

Question 76. Traduisez la requête suivante en français :

```
SELECT  W1.CITY
FROM    WAREHOUSES W1, STOCK S1
WHERE   W1.W# = S1.W#
AND     S1.COLOR <> 'red'
GROUP BY W1.CITY
HAVING  SUM(S1.QTY) > ( SELECT SUM(S2.QTY)
                        FROM    WAREHOUSE W2, STOCK S2
                        WHERE   W2.W# = S2.W#
                        AND     W2.CITY = W1.CITY
                        AND     S2.COLOR = 'red' );
```

La base de données est celle de la question 4.

Question 77. Pour la base de données de la question 9, donnez une requête SQL qui donne le nombre total de médailles par année et par pays, pourvu que ce nombre soit supérieur à zéro. La réponse est comme suit :

Année	Pays	Nombre
2004	USA	2
2004	JAP	1
2004	GBR	1
2000	NED	1
2000	GBR	1

Question 78. Pour la base de données de la question 10, donnez une requête SQL pour répondre à la question :

Quelle bière apparaît le plus grand nombre de fois comme bière favorite ? $\leadsto$ Leffe

Exercices sur les Chapitres 6 et 7

Question 79. Voici une table qui sert à stocker des informations sur les cours à l’UMH. Tout cours a un code et un titre unique, et est enseigné par un seul enseignant. Les leçons commencent à 8h15, 10h30, 13h30 ou 15h30, et durent 2 heures. On ne tient pas compte des jours fériés. Un cours peut avoir comme prérequis un certain nombre d’autres cours. La première rangée signifie que le cours S/3I/2, intitulé “Systèmes d’information”, est enseigné par J. Wijzen ; les leçons de ce cours ont lieu à la salle 3E11/P, chaque jeudi du deuxième semestre, à partir de 13h15. Le cours S/2I/17 est un prérequis pour ce cours. La deuxième rangée stipule que le cours S/3I/5 est un autre prérequis pour le cours S/3I/2.

Code	Titre	Enseignant	Sem	Jour	Heure	Local	Prérequis
S/3I/2	Systèmes d’information	J. Wijzen	2	Jeudi	13h30	3E11/P	S/2I/17
S/3I/2	Systèmes d’information	J. Wijzen	2	Jeudi	13h30	3E11/P	S/3I/5
S/2I/17	Fichiers et Bases de Données	J. Wijzen	1	Vendredi	10h30	3E11/P	S/1I/3
S/3I/5	Structure de l’information	V. Bruyère	1	Vendredi	10h30	3E10/P	S/1I/3
S/1I/3	PrInformatique I	P. Dufour	1	Vendredi	10h30	211/VI	—
S/1I/3	Informatique I	P. Dufour	1	Mardi	15h30	209/VI	—

Quelles sont les DFs valables pour ces données ? Remarquons que ces DF doivent exprimer, entre autres, que deux cours différents qui sont enseignés au même moment, doivent avoir des enseignants et des locaux différents.

Question 80. Considérons le schéma-DF {Code, Titre, Enseignant, Sem, Jour, Heure, Local, Prérequis} avec les DFs de la question 79. Quelles sont les clés de ce schéma-DF ? Est-ce que ce schéma-DF est en 3NF ? Expliquez.

Question 81. Supposons une table FILMS qui sert à enregistrer des informations sur des films. Les attributs ont les significations évidentes. Deux films peuvent porter le même Titre, mais la combinaison Titre plus Directeur constitue une identification unique d’un film. Supposons que la production de chaque film est assurée par une seule société. L’attribut Minutes donne la durée d’un film en minutes. L’attribut Première est la date de la première. Plusieurs premières peuvent avoir lieu à la même date pourvu que les films concernés n’aient pas de réalisateurs ou acteurs en commun, afin de permettre aux directeurs et acteurs d’assister aux premières de leurs films. Voici un exemple de cette table :

FILMS

Titre	Directeur	Acteur	Société	Première	Minutes
The Birds	A. Hitchcock	T. Hedren	Universal Pictures	28/03/1963	113
The Birds	A. Hitchcock	R. Taylor	Universal Pictures	28/03/1963	113
Titanic	J. Cameron	K. Winslet	Twentieth Century Fox	19/12/1997	195
Titanic	J. Cameron	L. DiCaprio	Twentieth Century Fox	19/12/1997	195
The Birds	J. Cameron	K. Winslet	Paramount Pictures	28/01/2001	182
The Birds	J. Cameron	R. Taylor	Paramount Pictures	28/01/2001	182

1. Pour simplifier la notation, désignons chaque attribut par sa première lettre (donc T pour Titre, D pour Directeur...). Donnez l'ensemble Σ de dépendances fonctionnelles qui doivent être satisfaites par toute relation "valide" sur $\mathcal{A} := \{T, D, A, S, P, M\}$. Donnez les clés du schéma-DF $(\mathcal{A}, \Sigma)$ obtenu.
2. Expliquez pourquoi le schéma-DF $(\mathcal{A}, \Sigma)$ qui résulte du point (1) n'est pas en 3NF. Donnez une décomposition en 3NF qui préserve les DFs.
3. Si la décomposition qui résulte du point (2) n'est pas en BCNF, donnez une décomposition en BCNF.

Réponse.

$$\Sigma = \{ \begin{array}{l} TD \rightarrow SPM, \\ DP \rightarrow T, \\ AP \rightarrow TD \end{array} \}$$

Les clés sont TDA et AP . Cette relation n'est pas en 3NF parce que $\Sigma \models TD \rightarrow S$, $\Sigma \not\models TD \rightarrow A$ et S n'est pas premier.

Le bon sens évoque une décomposition en deux composants : un premier composant pour stocker la société, la date de la première et la durée de tout film, un deuxième composant pour stocker les acteurs de tout film. Noter $\Sigma \models [TDSPM, TDA]$.

Composant	Schéma	DF	Clé(s)	BCNF ?
1	$TDSPM$	$\{TD \rightarrow SPM, DP \rightarrow T\}$	TD, DP	oui
2	TDA	$\{\}$	TDA	oui

Il est clair que cette décomposition ne préserve pas les DFs $AP \rightarrow T$ et $AP \rightarrow D$.

Essayons ensuite d'ajouter au deuxième composant l'attribut P afin de préserver $AP \rightarrow TD$:

Composant	Schéma	DF	Clé(s)	BCNF ?	3NF ?
1	$TDSPM$	$\{TD \rightarrow SPM, DP \rightarrow T\}$	TD, DP	oui	oui
2	$TDAP$	$\{AP \rightarrow TD, TD \rightarrow P, DP \rightarrow T\}$	AP, TDA	non	oui

Notez que $\Sigma \models TDP \rightarrow SM$, et donc $\Sigma \models [TDSPM, TDAP]$. Ceci est une décomposition en 3NF qui préserve les DFs.

Remarquons que l'on peut supprimer l'attribut P du premier composant :

Composant	Schéma	DF	Clé(s)	BCNF ?	3NF ?
1	$TDSPM$	$\{TD \rightarrow SM\}$	TD	oui	oui
2	$TDAP$	$\{AP \rightarrow TD, TD \rightarrow P, DP \rightarrow T\}$	AP, TDA	non	oui

Notez que $\Sigma \models TD \rightarrow SM$, et donc $\Sigma \models \bowtie [TDSM, TDAP]$. Ceci est une autre décomposition en 3NF qui préserve les DFs.

Pour l'exemple, la dernière décomposition donne les tables suivantes :

Titre	Directeur	Société	Minutes
The Birds	A. Hitchcock	Universal Pictures	113
Titanic	J. Cameron	Twentieth Century Fox	195
The Birds	J. Cameron	Paramount Pictures	182

Titre	Directeur	Acteur	Première
The Birds	A. Hitchcock	T. Hedren	28/03/1963
The Birds	A. Hitchcock	R. Taylor	28/03/1963
Titanic	J. Cameron	K. Winslet	19/12/1997
Titanic	J. Cameron	L. DiCaprio	19/12/1997
The Birds	J. Cameron	K. Winslet	28/01/2001
The Birds	J. Cameron	R. Taylor	28/01/2001

□

Question 82. La Société Nationale des Cinémas Belges (SNCB) stocke des informations sur les cinémas et leur programmation actuelle dans la table montrée ci-dessous. Les premières deux lignes signifient que le film “Felice” de P. Delpeut est programmé à 19 :15 et à 21 :15 au cinéma Utopia à Alost. Ce film, avec une durée de 1 heure et 39 minutes, est aussi programmé au cinéma Rex à Alost, à 18 :00 (cinquième ligne). Il ne faut pas confondre ce film avec celui de S. Spielberg montré à l’Utopia à Namur (dernière ligne).

Bien sûr, il peut y avoir plusieurs cinémas dans la même ville, mais il est impossible d’avoir deux cinémas différents à la même adresse postale. De plus, tous les cinémas de la même ville auront des noms différents. Chaque cinéma ne dispose que d’une seule salle de projection. Dès lors, aucun cinéma ne peut programmer deux films différents au même moment. Un film est identifié de manière unique par son titre plus son régisseur. La durée d’un film est une donnée fixe. Notons que deux films différents peuvent avoir le même titre. Chaque cinéma possède un ou plusieurs numéros de téléphone, mais aucun numéro n’est partagé parmi plusieurs cinémas.

CinémaNom	Rue	Ville	Téléphone	Titre	Régisseur	Durée	Heure
Utopia	6 Place du Parc	Alost	053 66 33 33	Felice	P. Delpeut	1 :39	19 :15
Utopia	6 Place du Parc	Alost	053 66 33 33	Felice	P. Delpeut	1 :39	21 :15
Utopia	6 Place du Parc	Alost	053 88 44 44	Felice	P. Delpeut	1 :39	19 :15
Utopia	6 Place du Parc	Alost	053 88 44 44	Felice	P. Delpeut	1 :39	21 :15
Rex	8 Rue du Marché	Alost	053 44 22 22	Felice	P. Delpeut	1 :39	18 :00
Utopia	5 Avenue Codd	Namur	081 33 66 99	Little Sister	T. Ravolta	1 :30	18 :15
Utopia	5 Avenue Codd	Namur	081 33 66 99	Felice	S. Spielberg	2 :05	19 :00

Quelles sont les DFs pour ce schéma-de-relation ?

Réponse.

- CinémaNom, Ville $\rightarrow$ Rue
- Rue, Ville $\rightarrow$ CinémaNom
- Téléphone $\rightarrow$ CinémaNom, Ville
- CinémaNom, Ville, Heure $\rightarrow$ Titre, Régisseur

— Titre, Régisseur → Durée

□

Question 83. Démontrez de manière précise que le schéma-DF présenté à la question 82 n'est pas en BCNF.

Question 84. Vous êtes engagé par la SNCB comme expert en BD. En effet, la société a remarqué que la table de la question 82 est difficile à maintenir dû à des informations dupliquées. Ceci ne vous étonne pas, car la violation de BCNF est évidente (voir question 83). Donnez, en vous basant sur votre “bon sens”, une décomposition de cette table qui permet de stocker les mêmes informations et qui ne souffre pas des informations redondantes. Ajoutez les clés primaires et étrangères.

On peut se baser sur le “bon sens” pour arriver à la décomposition.

Réponse. En s'appuyant sur le “bon sens”, on trouve le schéma-de-base-de-données :

CINEMAS (CinémaNom, Ville, Rue)
 TELEPHONES (Téléphone, CinémaNom, Ville)
 FILMS (Titre, Régisseur, Durée)
 PROGRAMME (CinémaNom, Ville, Titre, Régisseur, Heure)

Cependant, ceci n'est pas une décomposition correcte, parce que la deuxième condition de la Définition 1 (i.e., la condition de *lossless join*) n'est pas vérifiée. En fait, considérez la relation R suivante qui satisfait toute DF :

R	CinémaNom	Rue	Ville	Téléphone	Titre	Régisseur	Durée	Heure
	a_1	a_2	a_3	a_4	b_5	b_6	b_7	b_8
	a_1	a_2	a_3	b_4	a_5	a_6	a_7	a_8

La décomposition donne :

CINEMAS	CinémaNom	Rue	Ville		
	a_1	a_2	a_3		
TELEPHONES	CinémaNom	Ville	Téléphone		
	a_1	a_3	a_4		
	a_1	a_3	b_4		
FILMS	Titre	Régisseur	Durée		
	b_5	b_6	b_7		
	a_5	a_6	a_7		
PROGRAMME	CinémaNom	Ville	Titre	Régisseur	Heure
	a_1	a_3	b_5	b_6	b_8
	a_1	a_3	a_5	a_6	a_8

La jointure des composants donne un tuple $\langle a_1, a_2, a_3, \dots, a_8 \rangle$ qui ne fait pas partie de R .

Pour arriver à une décomposition en BCNF qui est *lossless join*, il faut ajouter un composant TH(Téléphone, Heure). En quelque sorte, ce composant pourrait encoder quel téléphone est opérationnel à quelle heure. Cependant, il semble logique de supposer la DMV

CinémaNom, Rue, Ville $\rightarrow$ Téléphone,

qui est équivalente à

CinémaNom, Rue, Ville $\rightarrow$ Titre, Régisseur, Durée, Heure.

Intuitivement, pour chaque cinéma, les numéros de téléphone et la programmation sont indépendants. Ces DMVs ne peuvent pas être exprimées en tant que DF. Avec ces DMVs, notre solution “bon sens” est bien *lossless join*. $\square$

Question 85. La table suivante sert à stocker les podiums (médaille d’or, d’argent et de bronze) de la natation aux jeux Olympiques. On ne prend en compte que des courses individuelles, pas les courses de relais.

- Supposons que tout athlète est identifié par un nom unique et invariable. Par exemple, l’athlète nommé “M. Phelps” de 2004 est le même que celui de 2000.
- Un athlète ne peut pas changer de sexe. Néanmoins, il peut changer de nationalité. Dans l’exemple, M. Phelps est passé de la Grande-Bretagne (GBR) aux états-Unis (USA) entre 2000 et 2004. Évidemment, il est interdit de nager pour deux pays différents pendant les mêmes jeux Olympiques.
- La finale d’une discipline a lieu à un jour déterminé. Par exemple, en 2004, la finale du 200 mètres papillon messieurs a eu lieu le 17/8. Dans une course, le temps du vainqueur (médaille d’or) est inférieur au temps du deuxième (médaille d’argent); le temps du deuxième est inférieur à celui du troisième (médaille de bronze). Pour une course, il y a exactement une médaille de chaque couleur. évidemment, un même nageur ne peut pas gagner deux médailles différentes dans une même course. Il est possible de gagner plusieurs médailles à une même date dans des disciplines différentes.
- Les jeux Olympiques sont attribués à une seule ville : Athènes en 2004, Sydney en 2000... Certaines villes, telles que Paris et Athènes, ont déjà organisé les jeux plus d’une fois.

NATATION

Année	Ville	Nage	Distance	Sexe	Date	Athlète	Pays	Médaille	Temps
2004	Athènes	Papillon	200	M	17/8	M. Phelps	USA	Or	1 :54.04
2004	Athènes	Papillon	200	M	17/8	T. Yamamoto	JAP	Argent	1 :54.56
2004	Athènes	Papillon	200	M	17/8	S. Parry	GBR	Bronze	1 :55.52
2004	Athènes	Libre	200	M	16/8	M. Phelps	USA	Bronze	1 :45.32
2000	Sydney	Libre	50	F	23/9	I. de Bruijn	NED	Or	0 :24.32
2000	Sydney	Papillon	200	M	23/9	M. Phelps	GBR	Argent	1 :55.19
									$\vdots$

- Quelles sont les DFs pour ce schéma-de-relation ? Écrivez les DFs en format $X \rightarrow A$ avec A un seul attribut. Assurez vous que X est minimal. Évitez les DFs redondantes, i.e., les DFs qui sont une conséquence logique des autres.
- Donnez une clé pour ce schéma-DF.

Question 86. Un club de tennis gère les réservations de ses terrains de tennis. Les dix terrains sont numérotés I, II, III, IV, ..., X. L’attribut Site indique si un terrain se trouve en salle (salle)

ou en plein air (dehors) ; cette caractéristique d'un terrain ne change jamais. Les terrains peuvent être réservés en tranches d'une heure : la première tranche commence à 9h, la dernière à 19h. On stockera la personne (**Nom**) qui fait la réservation, le terrain (**Terrain**) et la tranche (**Jdate** et **Tranche**). L'attribut **Nom** détermine une personne de manière unique. On enregistre aussi la date où la réservation a été introduite, c'est l'attribut **Rdate**. La date où on introduit la réservation (**Rdate**) est bien sûr antérieure à la date où on joue (**Jdate**). L'attribut **Membre** indique si la personne était membre du club au moment où la réservation était introduite. L'affiliation au club se fait sur base quotidienne ; c'est-à-dire, une personne ne peut avoir des affiliations différentes à une même date. Le prix d'une réservation ne dépend que de deux facteurs : le site (les terrains en salle sont plus chers) et l'affiliation (les membres payent moins). Il faut veiller à ce qu'un même terrain ne soit réservé qu'une seule fois pour une même tranche. Notons qu'une même personne peut réserver plusieurs terrains au même moment.

Par exemple, les deux premières rangées indiquent qu'à la date du 8 janvier, J. Henin a réservé le terrain IV pour le 22 janvier pendant deux tranches successives, à partir de 9h. Après vient K. Clijsters. Notons que le prix d'un terrain en salle est de 100 euros pour les membres et de 150 euros pour les non-membres. Apparemment, K. Clijsters est devenue membre du club à partir du 10 janvier ; à cette date, elle introduit une réservation pour les terrains IV et II à l'heure du midi.

	Rdate	Nom	Membre	Terrain	Site	Jdate	Tranche	Prix
	8 jan 2004	J. Henin	oui	IV	salle	22 jan 2004	9	100
	8 jan 2004	J. Henin	oui	IV	salle	22 jan 2004	10	100
	9 jan 2004	K. Clijsters	non	IV	salle	22 jan 2004	11	150
	10 jan 2004	K. Clijsters	oui	IV	salle	22 jan 2004	12	100
	10 jan 2004	K. Clijsters	oui	II	dehors	22 jan 2004	12	50

— Quelles sont les DFs pour ce schéma-de-relation ?

— Donnez une clé pour ce schéma-DF.

Question 87. La table suivante sert à stocker les réservations des chambres dans un hôtel.

RESERVATIONS

#Client	NomClient	Domicile	NrChambre	FaiteLe	Séjour	CartePaiement
111	Jean Dufour	Belgique	10	10/08/2003	14/10/2003	1111-1111
111	Jean Dufour	Belgique	11	10/08/2003	14/10/2003	4444-4444
222	Pierre Dupont	France	10	12/06/2003	15/10/2003	2222-2222
222	Pierre Dupont	Espagne	10	21/07/2003	16/10/2003	3333-3333
333	Jean Dufour	Belgique	22	11/08/2003	15/10/2003	1111-1111

Les deux premières rangées indiquent qu'à la date du 10/08/2003, Jean Dufour a contacté l'hôtel pour réserver les chambres 10 et 11 pour la date du 14/10/2003. Le 10/08/2003, Jean Dufour était domicilié en Belgique. Evidemment, une chambre ne peut être réservée qu'une seule fois pour une date donnée.

Une carte de paiement est indispensable pour garantir une réservation ; toute carte de paiement est personnalisée et portera le nom de la personne qui a demandé la réservation. On n'enregistre jamais plus d'une carte de paiement par réservation. Bien sûr, une personne peut avoir plusieurs cartes de paiement. Tout numéro de carte est unique au cours du temps.

La même personne peut être connue sous différents identifiants (**#Client**). Par exemple, les identifiants 111 et 333 identifient tous les deux le porteur de la carte 1111-1111. Néanmoins,

les identifiants des clients, comme les cartes de paiements, sont personnalisés et ne seront jamais ré-utilisés pour d'autres clients. Donc, bien que deux personnes différentes puissent occasionnellement avoir le même nom et domicile, elles ne peuvent jamais partager la même carte de paiement ou le même #Client.

Une personne ne peut être domiciliée dans différents pays en même temps. Néanmoins, le domicile d'une personne peut varier dans le temps. Par exemple, Pierre Dupont habitait d'abord en France, puis en Espagne.

- Quelles sont les DFs pour ce schéma-de-relation ?
- Donnez une clé pour ce schéma-DF.

Question 88. *Monsieur Bricolage* est une entreprise qui loue des outils de bricolage (perceuses, bétonnières. . .) au grand public sur base journalière.

NrOutil	Type	Prix	NrNat	Nom	Tel	Date
111	perceuse	34	60 10 07 123 12	Jean Leduc	053 56 65 34	7 dec 2002
111	perceuse	37	59 07 06 233 57	Anne Leblanc	053 56 65 34	8 dec 2002
232	bétonnière	233	60 10 07 123 12	Jean Leduc	054 26 23 11	9 jan 2002
333	perceuse	37	59 12 07 223 90	Anne Leblanc	016 99 33 34	8 dec 2002
999	marteau	3	59 12 07 223 90	Anne Leblanc	016 99 33 34	8 dec 2002

NrOutil dénote le numéro unique d'un outil. NrNat est le numéro national de l'emprunteur. Les autres attributs sont le type d'outil (Type), le prix journalier de location (Prix), le nom et le téléphone de l'emprunteur (Nom et Tel), la date de location (Date). Il n'y a qu'un seul prix pour un type d'outil pour une journée donnée. Par exemple, le 8 décembre 2002, le prix d'une perceuse était fixé à 37 Euros. Bien sûr, le prix d'un type d'outil peut augmenter au cours du temps. Évidemment, il ne peut y avoir qu'un seul emprunteur par jour et par outil. Le type d'un outil ne change pas. Deux personnes peuvent avoir le même nom. Le nom et le numéro national d'une personne ne changent jamais. Le numéro de téléphone d'une personne peut évoluer. Néanmoins, si une personne loue deux outils au même jour, ces deux locations seront accompagnées du même numéro d'appel.

- Quelles sont les DFs pour ce schéma-de-relation ?
- Donnez toutes les clés pour ce schéma-DF.

Question 89. La Société du Tour de France stocke dans une table pour chaque étape :

- la date (jour + année) de l'étape ;
- la ville de départ et d'arrivée ;
- le vainqueur avec son équipe (VEquipe) et sa nationalité (VNat) ;
- le coureur qui a remporté le maillot jaune à la fin de l'étape (Jaune), avec son équipe (JEquipe) et sa nationalité (JNat).

L'attribut Nr donne le numéro du tour ; par exemple, la 61ème édition du Tour de France avait lieu en 1974. Tout coureur est identifié de manière unique par son nom ; la nationalité d'un coureur ne change jamais. Un coureur peut changer d'équipe entre deux éditions, mais il est

clair que les équipes ne changent pas pendant les trois semaines de course. Le Tour de France étant toujours organisé au début de l'été, aucune édition n'est à cheval sur deux années civiles.

Le deuxième tuple signifie que dans la 61ème édition, qui avait lieu en 1974, l'étape du 22/7 entre Bazas et Pau a été remportée par le belge Rik Looy, qui en plus a pris le maillot jaune de l'espagnol Juan Epo. En 1974, Rik Looy était payé par Molteni. Notez que l'année suivante, en 1975, Rik Looy avait un autre sponsor (Dotcom).

ETAPES

Nr	Année	Jour	Départ	Arrivée	Vainqueur	VEquipe	VNat	Jaune	JEquipe	JNat
61	1974	21/7	Bazas	Bazas	Jean Vite	Dotcom	F	Juan Epo	Cofidis	E
61	1974	22/7	Bazas	Pau	Rik Looy	Molteni	B	Rik Looy	Molteni	B
61	1974	23/7	Pau	La Mongie	Ed Vlug	Molteni	NL	Rik Looy	Molteni	B
61	1974	24/7	Aime	Cluses	Ed Vlug	Molteni	NL	Juan Epo	Cofidis	E
62	1975	22/7	Metz	Nancy	Jim Fast	Molteni	UK	Rik Looy	Dotcom	B
...										

Quelles sont les DFs pour ce schéma-de-relation ?

Question 90. Vous êtes engagé par la Société du Tour de France comme expert en BD. En effet, la Société a remarqué que la table de la question 89 est difficile à maintenir dû à des informations dupliquées. Ceci ne vous étonne pas, car la violation de BCNF est évidente. Donnez une décomposition de cette table qui permet de stocker les mêmes informations et qui ne souffre pas des informations redondantes. Ajoutez les clés primaires et étrangères.

Question 91. Chaque année, l'UMH organise la Journée Math-Sciences. Les scientifiques de l'université donnent alors des exposés destinés aux élèves du secondaire. Chaque exposé est donné par un seul conférencier et appartient à exactement une discipline parmi *bio*, *chimie*, *info*, *math* et *physique*. Les exposés ont lieu en trois sessions parallèles ; chaque session est attribuée à un amphithéâtre. Les exposés durent une heure et commencent à 9h00, 10h30, 13h00, 14h30. Ceci pourrait être le programme pour l'année 2005 :

	Van Gogh	Plisnier	Curie
9h00	3 est premier ! Ch. Michaux <i>math</i>	P=NP ? R. Astier <i>info</i>	Les abeilles P. Falmagne <i>bio</i>
10h30	L'élément H M. Hecq <i>chimie</i>	La résonance P. Gillis <i>physique</i>	Fermat R. Astier <i>math</i>
13h00	La vie sur Mars M. Wautelet <i>physique</i>	Devenir centenaire P. Rasmont <i>bio</i>	Le recyclage Ph. Dubois <i>chimie</i>
14h30	Les bourdons P. Rasmont <i>bio</i>	Vivre en 2100 M. Wautelet <i>physique</i>	La récursivité V. Bruyère <i>info</i>

Un conférencier est identifié de manière unique par son nom. Chaque exposé a un intitulé unique

et n'est donné qu'une seule fois pendant la journée. Les organisateurs assurent qu'il n'ait jamais deux exposés de la même discipline au même moment, afin de permettre aux participants de suivre tous les exposés d'une même discipline. Le même conférencier peut donner deux exposés sur des disciplines différentes (par exemple, R. Astier donne un exposé d'*info* et de *math*). Bien sûr, aucun conférencier ne peut assurer deux exposés au même moment. Un amphithéâtre ne peut accueillir qu'un seul exposé à la fois.

Les écoles qui souhaitent participer à cette journée fournissent une liste des élèves avec les exposés qu'ils souhaitent suivre. Chaque élève inscrit doit assister à quatre exposés successifs. Supposez que tout élève est identifié de manière unique par la combinaison de son nom plus son école. Deux élèves dans deux écoles différentes peuvent avoir le même nom. Pour chaque école, on enregistre aussi les noms d'un ou plusieurs professeurs qui accompagnent les élèves de leur école. Supposez que tout professeur est identifié de manière unique par son nom. Tout professeur est associé à une seule école.

Toutes les données sont stockées dans une seule table. L'exemple ci-après nous montre que J. Petit, un élève du Collège St-Luc II, suivra d'abord l'exposé intitulé "3 est premier!", puis l'exposé "La résonance". Les professeurs V. Delue et E. Depré accompagnent les élèves de ce collège.

Élève	École	Responsable	Intitulé	Conférencier	Discipline	Heure	Amphi
J. Petit	Collège St-Luc II	V. Delue	3 est premier !	Ch. Michaux	math	9h00	Van Gogh
J. Petit	Collège St-Luc II	E. Depré	3 est premier !	Ch. Michaux	math	9h00	Van Gogh
J. Petit	Collège St-Luc II	V. Delue	La résonance	P. Gillis	physique	10h30	Plisnier
J. Petit	Collège St-Luc II	E. Depré	La résonance	P. Gillis	physique	10h30	Plisnier
⋮							

- Quelles sont les DFs pour ce schéma-de-relation ? Écrivez les DFs en format $X \rightarrow A$ avec A un seul attribut ; assurez que X est minimal. Évitez les DFs redondantes, i.e., les DFs qui sont une conséquence logique des autres.
- Donnez une clé pour ce schéma-DF.

Question 92. Le contrôle technique automobile stocke dans une base de données les informations sur les voitures contrôlées :

- la marque et le type de voiture ;
- le carburant : diesel ou essence ;
- l'année de construction et le numéro de châssis ;
- la plaque d'immatriculation ;
- les dates où la voiture a été contrôlée, avec le kilométrage (Km) et le numéro national (NN) du propriétaire au moment du contrôle.

Les propriétés suivantes d'une voiture ne changent jamais : Marque, Type, Carburant, Année, Châssis#. Par contre, la plaque d'immatriculation et le propriétaire changent quand la voiture est vendue. Les plaques d'immatriculation sont personnalisées : une même plaque restera toujours attribuée au même numéro national. Une personne peut utiliser une plaque qui lui est attribuée pour plusieurs voitures au cours du temps, mais pas pour deux voitures différentes en même

temps. Une voiture n'appartient qu'à une seule personne à la fois. Les voitures ne sont contrôlées qu'une fois par an. Il est donc impossible d'avoir deux enregistrements pour la même voiture à la même journée. Le kilométrage d'une voiture ne diminuera jamais au cours du temps.

Marque	Type	Carburant	Année	Châssis#	Plaque	NN	Km	Date
Renault	Clio	benzine	1989	123456789	CGD689	1956 06 02 148 45	30765	17/06/2003
Renault	Clio	benzine	1989	123456789	CGD689	1956 06 02 148 45	39345	13/06/2004
Renault	Clio	benzine	1989	123456789	HHH111	1972 04 02 999 99	44005	17/06/2005
Peugeot	106	diesel	2003	000111222	CGD689	1956 06 02 148 45	1390	13/08/2005
⋮								

- Quelles sont les DFs pour ce schéma-de-relation ? Écrivez les DFs en format $X \rightarrow A$ avec A un seul attribut ; assurez que X est minimal. Évitez les DFs redondantes, i.e., les DFs qui sont une conséquence logique des autres.
- Donnez une clé pour ce schéma-DF.

Question 93. La relation R sert à stocker le contenu de mes CDs audio. L'exemple montre deux CDs : l'album "What's Inside" de Joan Armatrading et "À Tatons" d'Axelle Red. Chaque album est réalisé par un seul artiste et une seule maison de disques. L'ordre des chansons sur le CD est indiqué par des chiffres 1,2,3,... Supposons que l'attribut Artiste identifie les artistes de manière unique. Deux albums [chansons] peuvent porter le même titre, mais un même artiste ne sortira jamais deux albums [chansons] avec le même titre. Un album ne contient jamais deux fois la même chanson.

R	Album	Artiste	Année	MaisonDisques	Ordre	Chanson	Durée
	What's Inside	Joan Armatrading	1995	BMG Music	1	In Your Eyes	2 :59
	What's Inside	Joan Armatrading	1995	BMG Music	2	Everyday Boy	4 :34
				⋮			
	What's Inside	Joan Armatrading	1995	BMG Music	13	Trouble	4 :04
	À Tatons	Axelle Red	1996	Virgin Belgium	1	À Tatons	1 :28
	À Tatons	Axelle Red	1996	Virgin Belgium	2	Mon Café	4 :23
				⋮			
	À Tatons	Axelle Red	1996	Virgin Belgium	14	À Tatons reprise	1 :47

- Quelles sont les DFs pour ce schéma-de-relation ? Écrivez les DFs en format $X \rightarrow A$ avec A un seul attribut ; assurez vous que X est minimal. Évitez les DFs redondantes, i.e., les DFs qui sont une conséquence logique des autres.
- Donnez une clé pour ce schéma-DF.

Question 94. Une DF ou DJ sur un ensemble $\mathcal{A}$ d'attributs est *triviale* si $X \rightarrow Y$ est satisfaite par toute relation sur $\mathcal{A}$. Prouvez :

1. Une DF $X \rightarrow Y$ est triviale si et seulement si $Y \subseteq X$.
2. Une DJ $X \bowtie [X_1, \dots, X_\ell]$ est triviale si et seulement s'il existe $i \in \{1, \dots, \ell\}$ tel que $X_i = \mathcal{A}$.

Réponse. [1] L'implication $\Leftarrow$ est triviale. On démontrera $\Rightarrow$ par contraposition. Soit $X \rightarrow Y$ une DF telle que $Y \not\subseteq X$. Soit $A \in Y \setminus X$. Soit $R = \{t_1, t_2\}$ une relation telle que $t_1[X] = t_2[X]$ et $t_1(A) \neq t_2(A)$. Il est clair que R ne satisfait pas $X \rightarrow Y$. Donc, si $X \rightarrow Y$ n'est pas triviale, on peut toujours construire une relation R qui ne satisfait pas $X \rightarrow Y$.

[2] L'implication $\Leftarrow$ est facile à prouver. On démontrera $\Rightarrow$ par contraposition. Soit $\bowtie [X_1, \dots, X_\ell]$ une DJ telle que pour tout $i \in \{1, \dots, \ell\}$, $X_i \neq \mathcal{A}$. Soit $R = \{t_1, \dots, t_\ell\}$ une relation telle que pour tout $i \in \{1, \dots, \ell\}$, pour tout $A \in \mathcal{A}$,

$$t_i(A) = \begin{cases} 0 & \text{si } A \in X_i; \\ i & \text{sinon.} \end{cases}$$

Par exemple, pour $\bowtie [AB, BC, CD]$, la relation R est

R	A	B	C	D	
	0	0	1	1	(t_1)
	2	0	0	2	(t_2)
	3	3	0	0	(t_3)

Soit s le tuple sur $\mathcal{A}$ tel que $s(A) = 0$ pour tout $A \in \mathcal{A}$. Il est clair que $R \not\models \bowtie [X_1, \dots, X_\ell]$, car $s \in \pi_{X_1}(R) \bowtie \dots \bowtie \pi_{X_\ell}(R)$, mais $s \notin R$. $\square$

Question 95. Soit Ξ un ensemble de DJs sur un ensemble $\mathcal{A}$ d'attributs. Prouvez que pour toute DF $X \rightarrow Y$ sur $\mathcal{A}$ avec $Y \not\subseteq X$, on a $\Xi \not\models X \rightarrow Y$.

Question 96. Développez un algorithme pour décider si deux ensembles Σ_1 et Σ_2 de DFs sont équivalents.

Question 97. Soit $\mathcal{A}$ un ensemble d'attributs et $X, Y \subseteq \mathcal{A}$. Prouvez $\{X \rightarrow Y\} \models \bowtie [XY, X(\mathcal{A} \setminus Y)]$.

Question 98. Considérez le schéma-de-relation SPJ . Prouvez $\bowtie [SP, PJ] \models \bowtie [SP, PJ, SJ]$ et $\bowtie [SP, PJ, SJ] \not\models \bowtie [SP, PJ]$.

Question 99. Soit $\mathcal{A} = ABCD$ un ensemble d'attributs. Donnez un ensemble Σ de DFs sur $\mathcal{A}$ tel que le schéma-DF $(\mathcal{A}, \Sigma)$ possède au moins six clés différentes.

Réponse. Remarquez : si X est une clé, alors tout ensemble Y tel que $Y \subsetneq X$ n'est pas une clé.

1. Si $\mathcal{A} = AB$, l'ensemble $\{A \rightarrow \mathcal{A}, B \rightarrow \mathcal{A}\}$ résulte en deux clés A et B . Il est clair que deux est le nombre maximal de clés pour deux attributs.
2. Prenons ensuite $\mathcal{A} = ABC$.
 - Si A est une clé, toute autre clé ne peut pas inclure A et est donc un sous-ensemble de BC . Puisque BC a au maximum deux clés (voir (1)), on obtient trois clés au maximum : A , B et C .
 - Si AB est une clé, ni A ni B n'est une clé. En plus, toute autre clé ne peut pas inclure AB . Les possibilités pour les autres clés sont donc : C , AC et BC . Si AB et C sont des clés, ni AC ni BC n'est une clé. Par contre, on obtient trois clés en choisissant AB , AC et BC .
 - Dès lors il est clair que trois est le nombre maximal de clés pour trois attributs.
3. Prenons ensuite $\mathcal{A} = ABCD$.

- Si A est une clé, toute autre clé ne peut pas inclure A et est donc un sous-ensemble de BCD . Puisque BCD résulte en trois clés au maximum (voir (2)), on n'obtient que quatre clés. Aucune clé ne peut donc être un singleton si l'on souhaite arriver à six clés.
- Si ABC est une clé, tout autre clé doit inclure D . Les possibilités sont donc AD , BD , CD , ABD , ACD , BCD . On arrive à quatre clés au maximum. Par exemple,
 - ABC , AD , BD et CD ; ou
 - ABC , ABD , ACD et BCD .

Aucune clé ne peut donc se composer de trois attributs si l'on souhaite arriver à six clés.

- Et si l'on prenait des clés de deux éléments ? L'ensemble :

$$\{AB \rightarrow \mathcal{A}, AC \rightarrow \mathcal{A}, AD \rightarrow \mathcal{A}, BC \rightarrow \mathcal{A}, BD \rightarrow \mathcal{A}, CD \rightarrow \mathcal{A}\}$$

résulte en six clés :

$$AB, AC, AD, BC, BD, CD.$$

□

Question 100. Soit $(\mathcal{A}, \Sigma)$ un schéma-DF avec $\mathcal{A} = ABCDEF$ et

$$\Sigma = \{AB \rightarrow CD, BC \rightarrow DE, CD \rightarrow EF, DE \rightarrow FA, EF \rightarrow B\}.$$

1. Quelles sont les clés de ce schéma-DF ?
2. Est-ce que ce schéma-DF est en BCNF ? Et en 3NF ? Expliquez.

Réponse. Cherchons d'abord les clés.

- Il est clair qu'aucun singleton n'est une clé (pourquoi?).
- AB détermine AB (bien sûr) ainsi que CD à cause de $AB \rightarrow CD$. AB détermine donc $ABCD$. Dès lors, AB détermine DE et EF à cause de $BC \rightarrow DE$ et $CD \rightarrow EF$. AB détermine donc $ABCDEF$. AB détermine donc tout autre attribut. Puisque ni A ni B n'est une clé, AB est une clé.
- De même façon, on trouve les clés BC , CD , DE .
- EF ne détermine que BEF et n'est donc pas une clé.
- On peut vérifier que toute autre combinaison de deux attributs n'est pas une clé : AC , AD , AE , AF , BD , BE , BF , CE , CF , DF .
- Par contre, AEF détermine tout autre attribut et ne contient aucune clé avec moins de trois éléments. AEF est donc une clé. On peut vérifier que AEF et CEF sont les seules clés avec trois éléments et qu'il n'y a pas de clé avec plus de trois attributs.
- Les clés sont donc : AB , BC , CD , DE , AEF , CEF .

Ce schéma-DF n'est pas en BCNF parce que $\Sigma \models EF \rightarrow B$ et $\Sigma \not\models EF \rightarrow A$. Ce schéma-DF est en 3NF puisque tout attribut fait partie d'une clé. $\square$

Question 101. Trouver un ensemble irréductible équivalent à $\{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C, AC \rightarrow D\}$.

Question 102. Considérez le schéma-DF $(\mathcal{A}, \Sigma)$ où

$$\mathcal{A} = \{W\#, \text{Address}, \text{City}, \text{Product}, \text{Color}, \text{Qty}, \text{Weight}\}$$

et

$$\Sigma = \{ \begin{array}{l} W\# \rightarrow \text{Address}, \text{City}; \\ \text{Address}, \text{City} \rightarrow W\#; \\ W\#, \text{Product}, \text{Color} \rightarrow \text{Qty}; \\ \text{Product} \rightarrow \text{Weight} \end{array} \}.$$

- Quelles sont les clés de ce schéma-DF ?
- Expliquez pourquoi ce schéma-DF n'est pas en 3NF.

Question 103. Pour le schéma-DF proposé en question 102, donnez une décomposition en 3NF qui préserve les DFs. Est-ce que cette décomposition est en BCNF ?

Question 104. Soit $(\mathcal{A}, \Sigma)$ un schéma-DF avec $\mathcal{A} = \{A, B, C\}$ et $\Sigma = \{A \rightarrow BC, B \rightarrow C\}$. Soit $\{(\mathcal{A}_1, \Sigma_1), (\mathcal{A}_2, \Sigma_2)\}$ une décomposition de ce schéma-DF avec :

- $\mathcal{A}_1 = \{A, B\}$,
- $\Sigma_1 = \{A \rightarrow B\}$,
- $\mathcal{A}_2 = \{B, C\}$,
- $\Sigma_2 = \{B \rightarrow C\}$.

Est-ce que cette décomposition préserve toutes les DFs du schéma-DF original ? Expliquez.

Question 105. Soit donné le schéma-DF $(ABCDEF, \Sigma)$ avec $\Sigma = \{ABC \rightarrow E, BCD \rightarrow F, EF \rightarrow ABC, CEF \rightarrow D\}$. Quelles sont les clés de ce schéma-DF ? Est-ce que ce schéma-DF est en 3NF ? Et en BCNF ?

Réponse. Les clés sont : $ABCD$, $ABCF$, $BCDE$ et EF . Le schéma-DF est en 3NF parce que tous les attributs font partie d'une clé. Le schéma-DF n'est pas en BCNF parce qu'il contient $ABC \rightarrow E$ et $\Sigma \not\models ABC \rightarrow F$. $\square$

Question 106. Soit donné le schéma-DF $(ABCDE, \{AB \rightarrow CD, C \rightarrow A, D \rightarrow CB\})$.

- Donnez toutes les clés de ce schéma-DF.

- En quelle forme normale (BCNF, 3NF, ni BCNF ni 3NF) se trouve ce schéma-DF ?
- Si le schéma-DF n'est pas en 3NF, proposez une décomposition en 3NF qui préserve les DFs.
- Est-ce que la décomposition en 3NF est en BCNF ?

Mêmes questions pour les schémas-DF suivants :

$(ABCD, \{AB \rightarrow C, B \rightarrow D\})$
 $(ABCDEF, \{ABC \rightarrow D, D \rightarrow E, E \rightarrow AD\})$
 $(ABCDE, \{AB \rightarrow C, C \rightarrow BDE, D \rightarrow C\})$
 $(ABCD, \{A \rightarrow B, B \rightarrow D, C \rightarrow AD, D \rightarrow C\})$
 $(ABCDE, \{AB \rightarrow C, CD \rightarrow E, E \rightarrow AC\})$
 $(ABCDE, \{AB \rightarrow CD, C \rightarrow A, D \rightarrow B, C \rightarrow E\})$

Question 107. Voici trois schémas-DF :

$(\mathcal{A}_1, \Sigma_1) = (ABCDE, \{A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E\})$
 $(\mathcal{A}_2, \Sigma_2) = (ABCDE, \{A \rightarrow BC, D \rightarrow AE\})$
 $(\mathcal{A}_3, \Sigma_3) = (ABCDEF, \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, BE \rightarrow C, CE \rightarrow FA, CF \rightarrow BD, D \rightarrow EF\})$

1. Est-ce que Σ_1 et Σ_2 sont équivalents ?
2. En quelle forme normale (BCNF, 3NF) est chacun de ces schémas-DF ?
3. Pour les schémas-DF qui ne sont pas en BCNF, trouver une décomposition en BCNF.
4. Pour les schémas-DF qui ne sont pas en 3NF, trouver une décomposition en 3NF qui préserve les DFs.

Question 108. Pour le schéma-DF $(\mathcal{A}, \Sigma)$ où $\mathcal{A} = \{A, B, C, D\}$ et $\Sigma = \{A \rightarrow B, B \rightarrow A, C \rightarrow D, D \rightarrow C\}$, répondez aux questions suivantes et expliquez vos réponses.

1. Est-ce que ce schéma-DF est en 3NF ? Si non, donnez une décomposition en 3NF qui préserve les DFs.
2. Est-ce que ce schéma-DF est en BCNF ? Si non, donnez une décomposition en BCNF qui préserve, si possible, les DFs.

Question 109. Pour le schéma-DF $(\mathcal{A}, \Sigma)$ où $\mathcal{A} = \{A, B, C, D, E\}$ et $\Sigma = \{A \rightarrow C, B \rightarrow D, CD \rightarrow E\}$, répondez aux questions suivantes et expliquez vos réponses.

1. Est-ce que le schéma-DF $(\mathcal{A}, \Sigma)$ est en 3NF ? Si non, donnez une décomposition en 3NF qui préserve les DFs.
2. Votre décomposition en 3NF est-elle aussi en BCNF ?

Question 110. Soit $\mathcal{R} = \{R\}$ un schéma-de-base-de-données avec $sorte(R) = \{A, B\}$. Soit $\Sigma = \{A \rightarrow B, R[A] \subseteq R[B]\}$. Soit $\sigma = B \rightarrow A$.

- Démontrer $\Sigma \models \sigma$.
- Démontrer qu'il existe une relation *infinie* qui satisfait Σ mais qui ne satisfait pas σ .

Exercices sur le Chapitre 8

Développer des schémas Entité-Association (Entity-Relationship) pour les énoncés ci-après. Traduire les schémas en modèle relationnel.

Question 111. La fanfare Saint-Martin possède différents instruments : flûtes traversières, clarinettes, saxophones (sopranos, altos, ténors, barytons), trompettes, trombones... Un numéro unique est gravé sur chaque instrument. La fanfare met ces instruments à la disposition de ses membres. Au moment où un membre loue un instrument, on enregistre la date et les défauts éventuels de l'instrument (par exemple, “embouchure tordue” ou “deux petites bosses”). Un membre peut louer plusieurs instruments de différents types (par exemple, trompette et saxophone). Cependant, plusieurs membres possèdent leur propre instrument et ne font pas appel au service de location.

Pour chaque membre, on enregistre le(s) instrument(s) qu'il sait jouer, avec un niveau d'excellence (débutant, avancé, virtuose). Une personne peut, par exemple, être un trompettiste débutant et un saxophoniste virtuose. Il faut aussi connaître le nom, prénom, âge et adresse de chaque membre.

La fanfare s'engage dans un nombre de concerts. Chaque concert a lieu à une date et endroit précis (par exemple, concert du 24 décembre 2007 à l'église St-Martin à Ghlin). Tous les membres sont censés participer à chaque concert. Néanmoins, suite à certains problèmes d'absentéisme dans le passé, le secrétaire voudrait désormais enregistrer la présence des membres. Évidemment, un membre peut toujours se faire excuser ; on enregistre alors la raison de son absence (par exemple, fièvre, examen à l'université...).

La base de données doit pouvoir répondre, entre autres, aux questions suivantes :

- Qui était absent, sans demande d'être excusé, au concert du 24 décembre 2007 à l'église St-Martin à Ghlin ?
- Qui sont les membres qui louent une trompette ?
- Quels trompettistes étaient présents au concert du 24 décembre 2007 à l'église St-Martin à Ghlin ?
- Qui atteint le niveau “virtuose” pour deux instruments différents ?

Question 112. Dix magasins de location de cassettes vidéo se sont regroupés pour mettre en commun les cassettes dont ils disposent et ont fondé un club de location. A la suite d'une rencontre avec les représentants de ce club, il ressort que chaque point de vente disposera d'un terminal clavier-écran relié à un site central et qu'il faudra pouvoir prendre en compte les

éléments suivants :

- un client qui s'inscrit au club verse une caution. Suivant le montant de cette caution il aura le droit d'emprunter en même temps de 1 à 6 cassettes ;
- les cassettes empruntées doivent être retournées dans un délai de 3 jours dans n'importe quelle boutique du club ;
- plusieurs cassettes peuvent contenir le même film ;
- un film est rattaché à un genre cinématographique (nom et type de public) et est caractérisé par sa durée, son réalisateur et la liste des acteurs principaux ;
- une location n'est permise que si le client est en règle (pas de dépassement du nombre d'emprunts maximum, pas de cassette en retard) ;
- la consultation d'un client permettra d'obtenir son nom, son adresse, son nombre d'emprunts en cours, la liste des numéros de cassettes et des titres qu'il a actuellement empruntés ;
- la consultation d'un genre permettra d'obtenir la liste des films de ce genre disponibles dans le magasin ;
- périodiquement, on veut obtenir la liste des retardataires ; on veut pour chaque cassette non retournée à temps les informations suivantes : nom et adresse du client, date de l'emprunt, numéro(s) de cassette et titre du (des) film(s) concerné(s) ;
- on veut pouvoir connaître pour chaque cassette (identifiée par une numérotation commune aux dix magasins) où elle est, quand elle a été mise en service, quel film y est enregistré, combien de fois elle a déjà été louée, et quel est son état (de très bon à mauvais).

Question 113. VOITURE5 est une chaîne de service automobile avec plusieurs centres dans lesquels les conducteurs peuvent se rendre pour des interventions techniques. Chaque intervention fait l'objet d'une garantie technique de trois mois (limitée à 20.000 kilomètres). Pour gérer ces garanties, la chaîne souhaite développer une base de données. Les interventions techniques sont de plusieurs types, chacun caractérisée par un code de deux lettres :

PE Petit entretien.

GR Grand entretien.

PN Montage pneu.

RE Montage rétroviseur.

AT Montage attelage.

...

Un prix forfaitaire est établi pour chaque type d'intervention et type de voiture :

Prix en EUR	PE	GR	PN	RE	AT	...
Peugeot Partner	40	80	100	10	135	...
Peugeot 307	35	75	100	10	—	...
BMW X3	60	120	—	20	320	...
⋮				⋮		

Notez que certaines interventions ne sont pas offertes pour certains types de voiture (indiqué par —), par exemple, l'intervention AT n'est pas offerte pour la Peugeot 307.

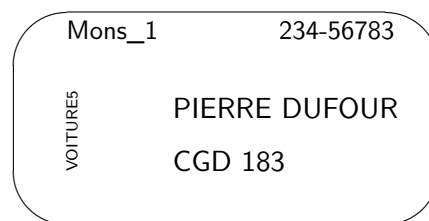
Les centres de service sont bien répandus en Belgique. Chaque centre est identifié par le nom de la ville suivi d'un numéro : Bruxelles_1, Bruxelles_2, Mons_1, Ninove_1, ... Pour chaque centre, il faut stocker l'adresse et un ou plusieurs numéros de téléphone. Chaque centre embauche un exploitant en chef et environs dix mécaniciens. Pour chaque personne embauchée, on enregistre le numéro de sécurité sociale, le nom, le prénom et l'adresse.

Seules les personnes possédant une carte de client **VOITURE5** peuvent profiter de la garantie sur les interventions techniques. Une telle carte peut être obtenue auprès de chaque centre de service ; les informations suivantes sont enregistrées dans la base de données :

Informations sur la personne : Nom, prénom et adresse du client.

Informations sur la voiture : Numéro de châssis, plaque d'immatriculation, type de voiture (par exemple, Peugeot Partner), année de construction.

Le nom, le prénom et l'immatriculation sont imprimés sur la carte, ainsi qu'un numéro d'identification unique et l'identifiant du centre qui a émis la carte :



Une telle carte de client donne accès à tous les autres centres de la chaîne. Donc, la carte montrée plus haut n'est pas limitée au centre Mons_1.

Pour chaque intervention, il faut mémoriser :

- le centre de service où l'intervention a eu lieu (Bruxelles_2, Mons_1,...),
- le type d'intervention (PE, GR, PN...),
- le mécanicien qui a effectué l'intervention,
- le numéro de la carte de client (qui identifie le client et sa voiture),
- la date,
- le kilométrage au moment de l'intervention.

Cette base de données devrait permettre de répondre à des questions telles que :

- Qui est le chef du centre Mons_1 ?
- Est-ce que le centre Bruxelles_2 accueille parfois des clients avec une carte émise par le centre Mons_2 ?
- Monsieur Dufour se plaint que sa voiture est tombée en panne deux jours après le grand entretien du 22 août 2007. Quel mécanicien a fait cet entretien ?
- Est-ce que l'intervention "montage attelage" (AT) est prévue pour la voiture associée à la carte 234-56783 et, si oui, à quel prix ?

Question 114. L'agence immobilière *Immons* sert d'intermédiaire entre deux parties :

- les propriétaires souhaitant vendre une ou plusieurs de leurs maisons, et
- les candidats acheteurs.

Pour chaque maison à vendre, l'agence enregistre l'adresse (rue, numéro, code postal, ville), la surface du terrain et le nombre de chambres à coucher. L'agence dispose également d'une brève description et d'une image de chaque maison. Chaque maison appartient à un seul propriétaire. Pour chaque propriétaire, l'agence enregistre ses nom et prénom, adresse et numéro d'appel.

Les maisons à vendre sont présentées sur le site web www.immons.com. Sur ce site, un candidat acheteur peut introduire une demande pour visiter une maison. Il doit fournir les données suivantes : nom, prénom, adresse postale, numéro d'appel et adresse email. L'agence prend ensuite contact avec le candidat. Si une visite est fixée, les données fournies par le candidat sont enregistrées dans une base de données (si elles n'y sont pas encore) ; l'adresse email sert comme clé. Il faut aussi stocker la date et l'heure de la visite et le "guide" qui sera responsable de la visite.

Les guides sont des personnes employées par l'agence pour assister aux visites (ouvrir et fermer la maison à clé, montrer les chambres...). Il s'agit souvent des étudiants salariés avec des contrats de courte durée. L'agence enregistre les données personnelles (nom, prénom, adresse, email, numéro d'appel) de chaque guide.

Après chaque visite, le guide doit remplir un questionnaire web avec deux questions :

1. Est-ce que le candidat acheteur s'est présenté ? Malheureusement, il arrive régulièrement que les candidats ne se présentent pas à une visite.
2. Est-ce que le candidat se montrait encore intéressé dans l'achat de la maison à la fin de la visite ?

Les réponses à ces deux questions sont ajoutées aux autres données sur la visite ; elle permettent de distinguer les "faux" candidats des vrais intéressés.

Les informations stockées dans la base de données sont utilisées pour répondre à des questions telles que :

- Combien de visites ont eu lieu pour la maison située Rue Saint-Thomas 5 à Mons ?
- Un candidat acheteur se plaint qu'il s'est présenté à une visite, mais qu'il n'y avait personne pour ouvrir la porte. Quel guide avait été désigné pour cette visite ? Est-ce que ce guide a déjà répondu aux deux questions ?
- Est-ce qu'il y a un candidat acheteur qui a déjà été absent à deux visites différentes ?
- Est-ce qu'il y a un propriétaire avec les mêmes nom et prénom qu'un candidat acheteur ? Peut-être qu'il s'agit d'un propriétaire qui est en même temps candidat acheteur pour une autre maison.

Question 115. Nailliers est une petite commune en France qui souhaite automatiser la gestion de sa bibliothèque communale. Le catalogue des livres doit stocker, pour chaque livre, le titre, le(s) auteur(s), l'année de publication et l'éditeur. La bibliothèque peut posséder plusieurs copies d'un même livre. Chaque copie est identifiée par un numéro unique. Chaque citoyen du village peut s'inscrire en tant que membre de la bibliothèque. Les noms, prénoms et adresses postales des membres doivent être enregistrés ; chaque membre sera identifié par un code unique. Les membres peuvent emprunter des livres à la bibliothèque. À chaque emprunt, on enregistre la

date, le code de l'emprunteur et le numéro de la copie empruntée. Les livres doivent être rendus dans une période de 15 jours ouvrables ; après cette période, le bibliothécaire peut (mais n'est pas obligé) envoyer un rappel. Le système doit garder une trace (la date et le membre destinataire) de chaque rappel. Le destinataire devra payer le coût de l'envoi au moment de son prochain emprunt. Un membre peut réserver un livre dont toutes les copies sont empruntées. Quand une copie d'un livre réservé est rendue, la copie n'est pas remise dans son rayon ; le membre qui a fait la réservation est averti qu'il peut venir chercher le livre.

La base de données doit permettre de répondre à plusieurs questions, par exemple,

- Où sont les deux copies du livre *L'amour fou* d'André Breton ?
- Donnez les codes des membres qui devraient recevoir un rappel.
- Qui a réservé *Tintin en Afrique* ?
- Combien de rappels ont été envoyés dans les six derniers mois ?

Question 116. La *Belgian Ocean Race* (BOR) est une course nautique autour du monde en équipage et par étapes. Les organisateurs de cette célèbre course voudraient créer une base de données permettant de retrouver toutes les informations relatives à l'organisation de la course et à sa sécurité.

La course se déroule en plusieurs étapes. Chaque étape débute et se termine dans un port, le port d'arrivée pouvant être le même que le port de départ. Il n'y a jamais plus d'une étape par jour. à l'arrivée de chaque étape, le chronométrateur détermine le temps mis par chaque bateau pour boucler l'étape. Les bateaux sont classés en deux catégories : les bateaux de séries et les prototypes. Après chaque étape, le classement d'étape et le classement général par catégorie de bateau sont publiés sur le site Web www.bor.be.

Les bateaux portent un nom unique et battent pavillon d'un seul pays. Chaque bateau est financé par un ou plusieurs sponsors et est armé d'un équipage composé d'un skipper et d'équipiers. Il faut connaître le prénom, nom et nationalité de chaque personne, ainsi qu'un numéro de téléphone à contacter en cas d'urgence. Pour chaque étape, il faut enregistrer la date, un bulletin météorologique (un seul champ texte) et les ports d'arrivée et de départ.

La base de données doit pouvoir répondre, parmi d'autres, aux questions suivantes :

- Quel était le port d'arrivée de l'étape du 21 juillet 2006 ?
- Quels sont les équipiers du bateau qui a gagné la première étape ?
- Quelle est la nationalité du skipper du bateau qui a gagné le classement général dans la catégorie "prototypes" ?

Question 117. Pour les besoins de la gestion d'un aéroport on souhaite mémoriser dans une base de données les informations nécessaires à la description des faits suivants :

- Chaque avion géré est identifié par un numéro d'immatriculation. Il est la propriété soit d'une société, soit d'un particulier : dans les deux cas on doit connaître le nom, l'adresse et le numéro de téléphone du propriétaire, ainsi que la date d'achat de l'avion.
- Chaque avion est d'un certain type, celui-ci étant caractérisé par son nom, le nom du constructeur, la puissance du moteur, le nombre de places.

- La maintenance des avions est assurée par les mécaniciens de l’aéroport. Par sécurité, les interventions sont toujours effectuées par deux mécaniciens (l’un répare, l’autre vérifie). Pour toute intervention effectuée, on conserve l’objet de l’intervention, la date et la durée.
- Pour chaque mécanicien on connaît son nom, son adresse, son numéro de téléphone et les types d’avion sur lesquels il est habilité à intervenir.
- Un certain nombre de pilotes sont enregistrés auprès de l’aéroport. Pour chaque pilote on connaît son nom, son adresse, son numéro de téléphone, son numéro de brevet de pilote et les types d’avion qu’il est habilité à piloter avec le nombre total de vols qu’il a effectué sur chacun de ces types.

Des questions types auxquelles l’application doit pouvoir répondre sont les suivantes :

- liste des avions de la société “Voltige” ;
- liste des avions propriété de particuliers ;
- durée totale des interventions faites par le mécanicien Durand au mois d’août ;
- liste des avions de plus de 4 places, avec le nom du propriétaire ;
- liste des interventions (objet, date) faites sur l’avion numéro 3242XZY78K3.

Exercices sur le Chapitre 9

Question 118. Supposez qu'une transaction T relâche un verrou sur un objet A dans un système 2PL. Plus tard, la transaction T est annulée afin de prévenir ou de guérir l'occurrence d'un verrou mortel. Est-ce possible dans un système qui s'appuie sur :

1. détection des verrous mortels ?
2. WAIT-DIE ?
3. WOUND-WAIT ?

Expliquez votre réponse pour chacune des trois possibilités.

Question 119. Considérez la séquence

$$S_1(A)S_2(A)X_1(A)X_2(A).$$

Comment cette séquence est-elle traitée par un gestionnaire de transactions qui exécute selon un protocole Strict 2PL + WAIT-DIE ?

Question 120. Considérez la séquence

$$S_3(B)S_1(A)S_2(A)X_3(A)X_2(A)C_1X_2(B).$$

Expliquez en détail comment cette séquence est-elle traitée par un gestionnaire de transactions qui exécute selon un protocole Strict 2PL + WOUND-WAIT ?

Réponse. Les trois premières demandes $S_3(B)S_1(A)S_2(A)$ peuvent être acceptées. Puis, la demande $X_3(A)$ est mise en attente, ce qui donne la table de verrouillage suivante :

Objet	verrous_acquis	file_d'attente
A	$\{S_1, S_2\}$	$\langle X_3 \rangle$
B	$\{S_3\}$	$\langle \rangle$

On suppose que les transactions sont estampillées selon leur temps de commencement. T_1 est donc la transaction la plus ancienne. La demande $X_2(A)$ ne peut pas être acceptée ; elle est insérée à la tête de la file d'attente, puisque T_2 possède déjà un verrou sur A .

Objet	verrous_acquis	file_d'attente
A	$\{S_1, S_2\}$	$\langle X_2, X_3 \rangle$
B	$\{S_3\}$	$\langle \rangle$

Les transactions T_2 et T_3 sont donc suspendues ; seule la transaction T_1 est encore “active” et termine (C_1) ; les verrous de T_1 sont relâchés. À ce moment, T_2 est reprise et reçoit un verrou exclusif sur A (*lock upgrade*). On obtient :

Objet	verrous_acquis	file_d'attente
A	$\{X_2\}$	$\langle X_3 \rangle$
B	$\{S_3\}$	$\langle \rangle$

T_2 continue avec la demande $X_2(B)$. T_2 n'attend pas jusqu'à T_3 relâche son verrou sur A : T_3 est plus jeune que T_2 et est donc annulée. On obtient :

Objet	verrous_acquis	file_d'attente
A	$\{X_2\}$	$\langle \rangle$
B	$\{X_2\}$	$\langle \rangle$

□

Question 121. Si, pour un objet donné, les **verrous_acquis** sont $\{S_2, S_3\}$, et la **file_d'attente** est $\langle X_1, X_2 \rangle$, alors un verrou mortel s'est produit. Dessinez le graphe d'attente pour cette situation et montrez l'existence du verrou mortel.

Question 122. Considérez l'exécution :

$$W_3(A)R_2(A)W_3(B)W_1(A)W_2(B).$$

Est-ce que cette exécution peut se produire, *dans exactement cet ordre*, dans un système 2PL ? En d'autres termes, imaginez-vous un observateur externe qui enregistre toutes les écritures et lectures d'un système 2PL en exécution. Est-ce qu'il est possible que cet observateur enregistre l'exécution donnée ? Si oui, complétez cette exécution en indiquant les positions dans l'exécution où les verrous nécessaires peuvent être acquis et relâchés. Si non, expliquez pourquoi cette exécution ne peut pas se produire. Même question pour les exécutions :

$W_1(A)R_2(A)R_3(B)W_1(B)$
 $R_1(A)W_2(B)W_2(A)W_1(A)$
 $R_1(A)W_2(B)W_2(C)R_2(A)W_1(C)$
 $W_3(A)R_2(A)W_3(B)W_1(A)W_2(B)$
 $R_1(A)R_2(A)W_2(B)W_1(B)W_3(A)R_2(A)$
 $R_1(A)R_2(A)R_3(B)W_1(B)W_2(B)R_3(C)$
 $R_1(A)R_2(A)R_3(B)W_2(B)R_3(A)W_1(B)$
 $R_1(C)R_1(A)W_2(B)R_2(A)W_1(D)W_2(C)W_1(A)$
 $R_1(A)R_2(A)R_2(B)W_3(C)W_1(B)W_1(A)W_2(C)W_3(D)$
 $R_1(A)R_1(B)R_2(A)R_3(B)W_3(B)W_1(A)W_2(C)$
 $R_1(A)W_2(B)W_2(A)W_1(C)W_2(C)$

Réponse. Cette exécution est possible en 2PL :

T_1	T_2	T_3
		$X_3(A)$
		$W_3(A)$
		$X_3(B)$
		$U_3(A)$
	$S_2(A)$	
	$R_2(A)$	
		$W_3(B)$
		$U_3(B)$
	$X_2(B)$	
	$U_2(A)$	
$X_1(A)$		
$W_1(A)$		
$U_1(A)$		
	$W_2(B)$	
	$U_2(B)$	

□

Question 123. Comme la question 122, en remplaçant 2PL par Strict 2PL.

Question 124. Simulez successivement l'exécution d'un gestionnaire de verrous

1. Strict 2PL + détection de verrou mortel,
2. Strict 2PL + WAIT-DIE et
3. Strict 2PL + WOUND-WAIT

pour les trois transactions T_1, T_2, T_3 suivantes :

$$\begin{aligned}
 T_1 &: S_1(A)X_1(B)C_1 \\
 T_2 &: S_2(A)S_2(B)C_2 \\
 T_3 &: X_3(A)C_3
 \end{aligned}$$

Supposons que l'ordre des demandes est déterminé par le *scheduler* des tâches¹ le suivant :

boucler sans arrêt

boucler pour toute transaction T_i non bloquée, en ordre d'ancienneté
traitez la prochaine demande de T_i

fin-boucler

redémarrez les transactions qui ont été annulées

fin-boucler

Réponse. Prenons d'abord le protocole Strict 2PL + WAIT-DIE. Les deux premières demandes à traiter sont $S_1(A)$ et $S_2(A)$, ce qui donne la table de verrouillage suivante :

Objet	verrous_acquis	file_d'attente
A	$\{S_1, S_2\}$	$\langle \rangle$
B	$\{\}$	$\langle \rangle$

1. Noter que ce scheduler n'intervient nulle part dans la théorie exposée dans le chapitre 9 ; il s'agit d'une tâche du système d'exploitation. Le scheduler présenté manque sans doute d'intelligence. Néanmoins, pour rendre cet exercice intéressant, il faut que l'on impose un certain entrelacement aux transactions.

La demande suivante est $X_3(A)$. La transaction T_3 “meurt” et est ensuite redémarrée. La prochaine demande $X_1(B)$ est acceptée :

Objet	verrous_acquis	file_d'attente
A	$\{S_1, S_2\}$	$\langle \rangle$
B	$\{X_1\}$	$\langle \rangle$

Puis, la transaction T_2 meurt quand elle demande $S_2(B)$:

Objet	verrous_acquis	file_d'attente
A	$\{S_1\}$	$\langle \rangle$
B	$\{X_1\}$	$\langle \rangle$

La transaction T_3 reprend la demande $X_3(A)$ et meurt de nouveau. Les transactions T_2 et T_3 sont ensuite redémarrées. La transaction T_1 atteint C_1 :

Objet	verrous_acquis	file_d'attente
A	$\{\}$	$\langle \rangle$
B	$\{\}$	$\langle \rangle$

La transaction T_2 reprend dès le début et effectue la demande $S_2(A)$:

Objet	verrous_acquis	file_d'attente
A	$\{S_2\}$	$\langle \rangle$
B	$\{\}$	$\langle \rangle$

La transaction T_3 meurt pour la troisième fois en effectuant $X_3(A)$. La transaction T_3 est redémarrée. La transaction T_2 continue avec $S_2(B)$:

Objet	verrous_acquis	file_d'attente
A	$\{S_2\}$	$\langle \rangle$
B	$\{S_2\}$	$\langle \rangle$

Ensuite la transaction T_3 meurt une fois de plus. La transaction T_2 se termine en effectuant C_2 . La transaction T_3 est maintenant la seule transaction dans le système et peut effectuer $X_3(A)C_3$. $\square$

Réponse. Prenons ensuite le protocole Strict 2PL + WOUND-WAIT. Les deux premières demandes à traiter sont $S_1(A)$ et $S_2(A)$, ce qui donne la table de verrouillage suivante :

Objet	verrous_acquis	file_d'attente
A	$\{S_1, S_2\}$	$\langle \rangle$
B	$\{\}$	$\langle \rangle$

Puis la transaction T_3 est suspendue quand elle effectue $X_3(A)$:

Objet	verrous_acquis	file_d'attente
A	$\{S_1, S_2\}$	$\langle X_3 \rangle$
B	$\{\}$	$\langle \rangle$

Puis la demande $X_1(B)$ peut être acceptée :

Objet	verrous_acquis	file_d'attente
A	$\{S_1, S_2\}$	$\langle X_3 \rangle$
B	$\{X_1\}$	$\langle \rangle$

La prochaine demande $S_2(B)$ est mise en attente et T_2 est suspendue :

Objet	verrous_acquis	file_d'attente
A	$\{S_1, S_2\}$	$\langle X_3 \rangle$
B	$\{X_1\}$	$\langle S_2 \rangle$

La demande $X_3(A)$ ne peut pas encore être acceptée ; T_3 reste donc en suspens. La transaction T_1 relâche ensuite ses verrous en effectuant C_1 :

Objet	verrous_acquis	file_d'attente
A	$\{S_2\}$	$\langle X_3 \rangle$
B	$\{\}$	$\langle S_2 \rangle$

À ce moment, la demande $S_2(B)$ peut être acceptée et la transaction T_2 reprend donc son exécution :

Objet	verrous_acquis	file_d'attente
A	$\{S_2\}$	$\langle X_3 \rangle$
B	$\{S_2\}$	$\langle \rangle$

La demande $X_3(A)$ de T_3 ne peut pas encore être acceptée ; T_3 reste en suspens. La transaction T_2 prend fin en effectuant C_2 :

Objet	verrous_acquis	file_d'attente
A	$\{\}$	$\langle X_3 \rangle$
B	$\{\}$	$\langle \rangle$

Enfin la demande $X_3(A)$ peut être acceptée :

Objet	verrous_acquis	file_d'attente
A	$\{X_3\}$	$\langle \rangle$
B	$\{\}$	$\langle \rangle$

Puis la transaction T_3 se termine. □

Question 125. Démontrez qu'il existe une exécution sérialisable qui n'est pas conforme au protocole 2PL.

Question 126. Une transaction est *read-only* si elle n'effectue aucune écriture. Supposez qu'une transaction T_1 *read-only* lit un objet A . Plus tard, T_1 lit le même objet une deuxième fois et constate que la valeur de A a changé. Qu'est-ce que ça nous apprend concernant le protocole de concurrence qui est utilisé dans ce système ? Expliquez.

Question 127. Supposez qu'une transaction T_1 relâche un verrou sur un objet A dans un système 2PL. Plus tard, la transaction T_1 est annulée pour des raisons de *deadlock*. Qu'est-ce que ça nous apprend concernant la manière dont le gestionnaire de transactions s'occupe du problème de *deadlock* ?

Exercices sur le Chapitre 10

Question 128. Voici le contenu d'un journal de type Undo/Redo après une panne de système :

```
[START T1 ]
[START T2 ]
[START T3 ]
[START T4 ]
[T1, A, 0, 1 ]
[T2, B, 0, 1 ]
[T3, C, 0, 1 ]
[T4, D, 0, 1 ]
[COMMIT T1 ]
[START CKPT ⟨T2, T3, T4⟩]
[START T5 ]
[START T6 ]
[START T7 ]
[T5, E, 0, 1 ]
[T6, F, 0, 1 ]
[T7, G, 0, 1 ]
[COMMIT T2 ]
[COMMIT T5 ]
[END CKPT ]
[COMMIT T3 ]
[COMMIT T6 ]
```

1. Quelles sont les valeurs possibles pour A, B, C, D, E, F, G dans la base de données sur disque au moment de la panne ?
2. Expliquez en détail comment le SGBD arrivera à un état cohérent lors de la procédure de reprise.

Question 129. Après une panne du système, le contenu du log est comme suit :

```
(T1, begin)
(T1, A, 4, 5)
(T2, begin)
(T1, commit)
(T2, B, 9, 10)
(T2, A, 5, 6)
(T3, begin)
(T3, C, 19, 20)
(T3, commit)
```

Quelles sont les valeurs pour A, B et C à restaurer dans la base de données pendant la reprise ?

Question 130. Comment la procédure de reprise après panne doit-elle être adaptée afin de tenir compte des enregistrements `START CKPT` et `END CKPT` ?

Question 131. Voici le contenu d'un journal Undo/Redo après une panne du système. Quelles sont les actions effectuées par le *recovery manager* pendant la reprise ?

(T1,begin), (T1,A,4,5), (T2,start), (T1,commit), (T2,B,9,10), (START CKPT(T2)),
(T2,C,14,15), (T3,begin), (T3,D,19,20), (END CKPT), (T2,commit)

Question 132. Comment la procédure de *checkpointing* doit-elle être adaptée au protocole Undo/No-Redo ? Et au Redo/No-Undo ?

Question 133. Comment la procédure de reprise après panne doit-elle être adaptée au protocole Undo/No-Redo ? Et au Redo/No-Undo ? D'abord sans *checkpointing*, puis avec la procédure de *checkpointing* élaborée ci-dessus (question 132).

Question 134. Dans une stratégie Undo/Redo, un checkpoint stocke dans la base de données *toute* page modifiée se trouvant dans le *buffer* au moment où le checkpoint démarre. Cette manière de réaliser un checkpoint n'est pas adaptée à une stratégie Redo/No-Undo, parce qu'elle stocke dans la base de données des modifications qui ne sont pas encore commises (voir aussi la section 10.7). Discutez ce problème et proposez une solution.

Bibliographie

- [1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [2] C. W. Bachman. The programmer as navigator. *Communications of the ACM*, 16(11) :635–658, 1973.
- [3] D. D. Chamberlin, M. M. Astrahan, M. W. Blasgen, J. Gray, W. F. K. III, B. G. Lindsay, R. A. Lorie, J. W. Mehl, T. G. Price, G. R. Putzolu, P. G. Selinger, M. Schkolnick, D. R. Slutz, I. L. Traiger, B. W. Wade, and R. A. Yost. A history and evaluation of system R. *Commun. ACM*, 24(10) :632–646, 1981.
- [4] P. P. Chen. The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, 1(1) :9–36, 1976.
- [5] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6) :377–387, 1970.
- [6] C. J. Date. *An Introduction to Database Systems, 6th Edition*. Addison-Wesley, 1995.
- [7] C. J. Date. *An introduction to database systems (7. ed.)*. Addison-Wesley-Longman, 2000.
- [8] G. Gardarin. *Bases de données : objet & relationnel*. Eyrolles, 1999.
- [9] W. Kent. A simple guide to five normal forms in relational database theory. *Commun. ACM*, 26(2) :120–125, 1983.
- [10] L. Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.
- [11] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.
- [12] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, 1988.
- [13] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume II*. Computer Science Press, 1989.
- [14] J. D. Ullman. Experiments as research validation : have we gone too far ? *Commun. ACM*, 58(9) :37–39, 2015.
- [15] M. W. Vincent. A corrected 5NF definition for relational database design. *Theor. Comput. Sci.*, 185(2) :379–391, 1997.

Annexe A

Les Grandes Découvertes en Bases de Données

Exposé à l'occasion de la Journée de Mathématique et de Sciences, 29 mars 2001.

A.1 Introduction

Dans ma jeunesse, j'étais passionné par les livres racontant les grandes découvertes scientifiques. Les découvertes dans le domaine des bases de données (BD) n'étaient pas parmi eux... Dans cet exposé, j'essaie néanmoins de montrer que les BD sont devenues un domaine de recherche intéressant et important en informatique depuis les années 60. J'explique quelles étaient les étapes principales de ces recherches et quels sont les problèmes restant à résoudre. Deux questions fondamentales posées par cette discipline sont :

1. Comment les données peuvent-elles être structurées ?
2. Comment les données peuvent-elles être interrogées ?

Le terme “requête” est utilisé pour une question posée à une BD en un langage interprété par l'ordinateur.

A.2 Les BD Hiérarchiques

Le premier système de BD a été conçu pour la gestion des données du projet Apollo de la NASA. Les données étaient structurées dans des hiérarchies, comparables à l'organisation des répertoires sur un PC. La figure A.1 donne un exemple d'une telle hiérarchie ; elle montre des animaux (Lion, Loup, Tigre,...) groupés dans des ordres (Carnivores, Artiodactyles, Serpents) qui eux-mêmes sont groupés dans des classes (Mammifères, Reptiles). Une telle structuration des données permet de répondre facilement aux questions de type :

Quels animaux sont carnivores ?

Supposons maintenant qu'on veuille ajouter des informations sur la répartition géographique des animaux. Au moins deux possibilités se présentent. La figure A.2 (*gauche*) ajoute les continents au plus bas niveau de la hiérarchie. Grâce à cette organisation, il est très facile de répondre à la question :

Où peut-on trouver des lions ?

Par contre, pour répondre à la question :



FIGURE A.1 – Classification hiérarchique des animaux.

Quels sont les carnivores d’Afrique ?

la hiérarchie montrée par la figure A.2 (*droite*) convient mieux, parce que les carnivores africains (Lion et Hyène) se retrouvent groupés.

Bien que personne ne mette en doute la hiérarchie représentée par la figure A.1, on observe qu’ajouter les continents peut se faire de plusieurs manières et qu’il n’y a pas d’organisation idéale pour toutes les requêtes. Il est facile de comprendre pourquoi : la relation entre les continents et les espèces n’est pas de nature hiérarchique, dans le sens où une espèce n’est pas limitée à un seul continent (et inversement, bien sûr, un continent contient plusieurs espèces). Il n’est donc pas naturel de vouloir stocker une telle relation dans une hiérarchie. On est enclin à croire qu’une structuration des données en réseau est plus naturelle qu’une organisation hiérarchique. Ce sont sans doute de telles considérations qui ont mené aux BD de type réseau.

A.3 Les BD de Type Réseau

Ce modèle de données sera toujours associé au nom de C.W. Bachman. La figure A.3 montre les mêmes données zoo-géographiques structurées en réseau. Les rectangles contiennent les données et les “circuits” représentent les relations entre les données. Par exemple, on reconnaît facilement le circuit qui relie les carnivores ; celui-ci permet de répondre à la question :

Quels animaux sont carnivores ?

Pour répondre à la question :

Quels sont les animaux d’Asie ?

il faut parcourir un chemin qui contient plusieurs circuits (lesquels ?). Finalement, pour répondre à la question :

Quels sont les carnivores d’Asie ?

plusieurs parcours sont possibles. Tout d’abord, on peut traverser le circuit qui relie les carnivores et sélectionner ceux qui sont liés à l’Asie. Alternativement, on peut partir du nœud “Asie”, parcourir les animaux asiatiques et sélectionner ceux qui se trouvent dans le circuit qui relie les carnivores.

Il faut comprendre qu’il n’est pas évident d’exprimer ces parcours en un langage de programmation. A titre d’exemple, le programme montré par la figure A.4 décrit le parcours qui trouve les

Les continents (Afrique, Asie, Europe) sont ajoutés à la base de la hiérarchie :

Classe	Ordre	Espèce	Continent
Mammifères			
	Carnivores		
		Lion	
			Afrique
			Asie
		Loup	
			Europe
			Asie
		Tigre	
			Asie
		Hyène	
			Afrique
		Artiodactyles	
			Girafe
			Afrique
		Hippopotame	
			Afrique
Reptiles			
	Serpents		
		Cobra	
			Asie

Les animaux de même ordre sont groupés par continent :

Classe	Ordre	Continent	Espèce
Mammifères			
	Carnivores		
		Afrique	
			Lion
			Hyène
		Asie	
			Loup
			Lion
			Tigre
		Europe	
			Loup
	Artiodactyles		
		Afrique	
			Girafe
			Hippopotame
Reptiles			
	Serpents		
		Asie	
			Cobra

FIGURE A.2 – Deux façons d’ajouter la répartition géographique à la classification hiérarchique des animaux.

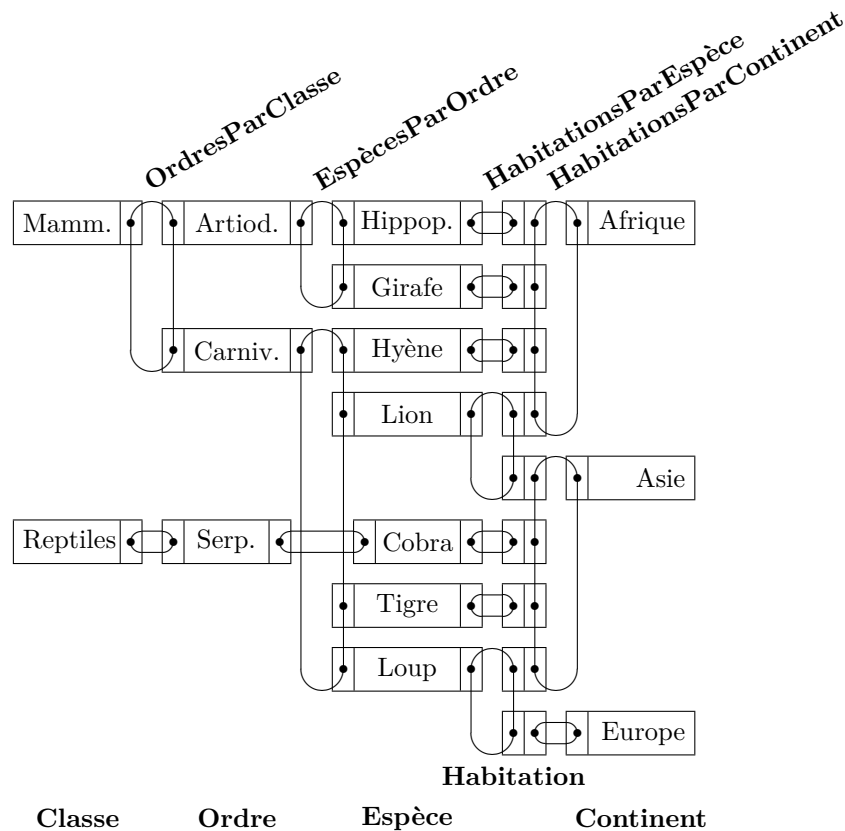


FIGURE A.3 – Classification et répartition des animaux dans une BD de type réseau.

```

FIND Continent WITHIN IndexSurContinents USING 'Asie';
FIND FIRST Habitation WITHIN HabitationsParContinent;
WHILE db-rec-found LOOP
    OBTAIN OWNER WITHIN HabitationsParEspèce;
    OBTAIN OWNER WITHIN EspècesParOrdre;
    IF Ordre = 'Carnivores' THEN print Espèce END-IF;
    FIND NEXT Habitation WITHIN HabitationsParContinent;
END-LOOP;

```

FIGURE A.4 – Programme navigationnel pour trouver les carnivores asiatiques.

carnivores asiatiques à partir du nœud “Asie”. Un tel programme est appelé “navigationnel” : le programmeur doit diriger de manière détaillée le parcours à travers les données en indiquant pas à pas les opérations à réaliser [2].

En 1973, C.W. Bachman a reçu le Prix Turing pour sa contribution à l’informatique. Ce prix signifie pour un informaticien ce qui signifie le Prix Nobel pour un physicien ou un chimiste.

A.4 Les BD Relationnelles

En 1970, au moment où les systèmes basés sur le modèle hiérarchique ou le modèle en réseau étaient en plein développement, E.F. Codd publiait un article [5] où il proposait de stocker des données dans des tables. A l’heure actuelle, cette solution peut nous sembler assez évidente ; pensons aux tables utilisées pour afficher les scores des matchs de football ou les listes de prix... Néanmoins, en 1970 cette idée était considérée comme une curiosité intellectuelle. On doutait

EO	Espèce	Ordre	OC	Ordre	Classe
	Lion	Carnivores		Carnivores	Mammifères
	Loup	Carnivores		Artiodactyles	Mammifères
	Tigre	Carnivores		Serpents	Reptiles
	Hyène	Carnivores			
	Girafe	Artiodactyles			
	Hippopotame	Artiodactyles			
	Cobra	Serpents			

VIT	Espèce	Continent
	Lion	Afrique
	Lion	Asie
	Loup	Europe
	Loup	Asie
	Tigre	Asie
	Hyène	Afrique
	Girafe	Afrique
	Hippopotame	Afrique
	Cobra	Asie

FIGURE A.5 – Classification et répartition des animaux dans une BD relationnelle avec trois tables.

que les tables puissent jamais être gérées de manière efficace par un ordinateur...

Une table se compose de plusieurs colonnes et rangées. Pour notre exemple, les tables sont celles de la figure A.5. En général, les tables et leurs colonnes sont fixées au moment de la conception de la BD. Après, on peut à tout moment changer le contenu des tables en insérant, en modifiant et en supprimant des rangées.

Dans le même article, E.F. Codd proposait d'utiliser une algèbre pour interroger les tables. L'algèbre proposée se compose de cinq opérateurs, parmi lesquels :

JOIN La jointure sert à joindre les rangées de deux tables. Les rangées à joindre sont celles qui ont la même valeur pour toute colonne commune aux deux tables.

SELECT La sélection sert à retenir les rangées qui vérifient une certaine condition, en supprimant les autres rangées.

PROJECT La projection sert à retenir certaines colonnes, en supprimant les autres.

Notons que le résultat de ces opérations est toujours une nouvelle table. La figure A.6 montre que la question :

Quels animaux sont des mammifères ?

peut être exprimée en algèbre par la requête suivante :

```
((EO JOIN OC )
  SELECT Classe='Mammifères')
  PROJECT Espèce
```

D'abord l'expression **(EO JOIN OC)** résulte en une table qui donne l'ordre et la classe de toute espèce dans la BD. Puis la sélection ne retient que les rangées qui portent sur les mammifères. Finalement, la projection ne retient que la colonne **Espèce**.

On peut maintenant vérifier que la requête :

```
((EO JOIN VIT )
  SELECT Ordre='Carnivores' &
    Continent='Asie')
```

EO JOIN OC	Espèce	Ordre	Classe
	Lion	Carnivores	Mammifères
	Loup	Carnivores	Mammifères
	Tigre	Carnivores	Mammifères
	Hyène	Carnivores	Mammifères
	Girafe	Artiodactyles	Mammifères
	Hippopotame	Artiodactyles	Mammifères
	Cobra	Serpents	Reptiles

(EO JOIN OC) SELECT Classe=Mammifères	Espèce	Ordre	Classe
	Lion	Carnivores	Mammifères
	Loup	Carnivores	Mammifères
	Tigre	Carnivores	Mammifères
	Hyène	Carnivores	Mammifères
	Girafe	Artiodactyles	Mammifères
	Hippopotame	Artiodactyles	Mammifères

((EO JOIN OC) SELECT Classe=Mammifères) PROJECT Espèce	Espèce
	Lion
	Loup
	Tigre
	Hyène
	Girafe
	Hippopotame

FIGURE A.6 – Pour répondre à la question “*Quels animaux sont des mammifères ?*” on joint (JOIN) d’abord les tables **EO** et **OC** pour ensuite en retenir (SELECT) les mammifères. Finalement, on ne retient (PROJECT) que la colonne **Espèce**.

PROJECT Espèce

donne tous les carnivores asiatiques. Cette requête est nettement plus simple que le programme équivalent pour la BD de type réseau montré par la figure A.4. Notons que toute requête en algèbre relationnelle sera traduite en un programme efficace qui peut être exécuté par l’ordinateur. Mais contrairement aux BD de type réseau, ce programme reste caché aux utilisateurs ; la traduction est effectuée automatiquement par le Système de Gestion de Bases de Données (SGBD). Pour cette raison, on dit que les requêtes en modèle relationnel sont “assertionnelles” : l’utilisateur définit les caractéristiques qui s’imposent au résultat. Le SGBD doit alors construire la stratégie de recherche.

Comme mentionné ci-dessus, au début des années 70, on considérait comme une curiosité intellectuelle l’idée de stocker les données dans des tables et d’interroger les tables de manière non-navigationnelle. Il faut comprendre que cette idée était révolutionnaire dans un temps où on

<pre> <H1> &Eacute;nigmes </H1> Comment faire entrer quatre &eacute;l&eacute;phants dans une fiat panda? Comment un &eacute;l&eacute;phant se mouche-t-il? </pre>	Énigmes 1. Comment faire entrer quatre éléphants dans une fiat panda ? 2. Comment un éléphant se mouche-t-il ?
--	--

FIGURE A.7 – Les balises dans une page HTML (gauche) sont interprétées par le navigateur qui affiche la page (droite).

était loin des interfaces conviviales pour interagir avec l'ordinateur. Ce scepticisme n'a cependant pas empêché E.F. Codd de poursuivre ses idées. Un premier prototype de Système de Gestion de Bases de Données Relationnelles (SGBDR) a été construit dans les laboratoires d'IBM. Depuis les années 80, cette technologie a mûri et a été adoptée par l'industrie. En 1987, le langage SQL, qui étend l'algèbre relationnelle, a été standardisé. A l'heure actuelle, les SGBDR sont présents dans toutes les compagnies et représentent une industrie de plusieurs milliards de dollars.

E.F. Codd a reçu le Prix Turing en 1981.

A.5 Le Web, une BD ?

A.5.1 Un Manque de Structure

Aujourd'hui, une immense quantité de données se trouve sur le Web. Néanmoins, il n'est guère possible de parler d'une vraie BD parce que, d'une part, ces données sont peu ou pas structurées et, d'autre part, il n'existe pas de langage pour interroger le Web.

Le Web manque de structure. Il est construit à partir de "pages" écrites en langage HTML (*HyperText Markup Language*). En gros, ce langage permet (i) de spécifier à l'aide de balises comment une page doit être présentée sur l'écran de l'ordinateur et (ii) d'ajouter des liens vers d'autres pages. La figure A.7 donne un exemple : le titre se trouve entre les balises <H1> et </H1> (*Header*) ; les balises et (*Ordered List*) délimitent le début et la fin d'une liste ordonnée ; chaque article de la liste se trouve entre les balises et (*List Item*).

Il y a deux manières de chercher des informations sur le Web :

- Utiliser des moteurs de recherche tels que Google, Hotbot et Alta Vista. Ces moteurs sont comme l'index d'un livre : on saisit un mot clé et le moteur retourne toutes les pages contenant ce mot. Malheureusement, cette méthode de recherche manque de précision. Par exemple, un biologiste qui s'intéresse à la symbiose entre les pandas et les mouches peut demander toutes les pages contenant à la fois les mots "panda" et "mouche". Il ne lui sera pas possible d'éviter des pages non pertinentes telles que celle montrée par la figure A.7.

Une petite expérience : le lundi 5 mars 2001, le moteur de recherche Google (<http://www.google.com/>) trouvait 168 pages rédigées en français contenant les mots "panda" et "mouche". La page classée en tête parle des "Gîtes Panda au Parc naturel régional Normandie-Maine" où on sait "pêcher la truite fario à la mouche"...

- Naviguer de site en site, ce qui fait penser à la navigation dans les BD de type réseau. Il y a pourtant une différence importante : contrairement au Web, les BD de type réseau se conforment à une structure précise.

A.5.2 Traiter le Futur Web comme BD

Le défi est de mieux structurer et décrire le contenu des pages Web. Supposons que tous les biologistes du monde se mettent d'accord pour utiliser des balises standardisées de manière à décrire les animaux dans leurs pages Web. Voici un exemple :

```
<ANIMAL>
  <ESPECE> Panda </ESPECE>
  <CLASS> Mammifères </CLASS>
  <NOURRITURE> bambou </NOURRITURE>
  <CONTINENT> Asie </CONTINENT>
```

</ANIMAL>

Par contre, le secteur automobile peut utiliser d'autres balises pour décrire les voitures :

```
<VOITURE>
  <MARQUE> Fiat </MARQUE>
  <MODELE> Panda </MODELE>
  <VITESSE en 'KMPH'> 140 </VITESSE>
</VOITURE>
```

Ces balises standardisées permettraient aux moteurs de recherche de faire la distinction entre un panda et une Fiat Panda. Un tel standard permettrait la construction de requêtes plus précises que celles s'appuyant uniquement sur des mots clés. Par exemple, la question :

Que mange le panda ?

pourrait se traduire en une requête à un moteur de recherche :

```
<ANIMAL>
  <ESPECE> Panda </ESPECE>
  <NOURRITURE> ? </NOURRITURE>
</ANIMAL>
```

Le moteur de recherche enverrait une liste contenant "bambou".

Est-ce que cette technologie est de la science-fiction ? Pas du tout : ce nouveau langage pour structurer le Web existe déjà ; il s'appelle XML (*eXtensible Markup Language*) ! Quant aux recherches, plusieurs prototypes de langages pour interroger des pages XML ont déjà été proposés. Cette technologie permettra dans un futur proche d'interroger le Web de manière précise et non-navigational, un peu comme les BD relationnelles. Il est fort possible que les recherches dans ce domaine aboutissent à un Prix Turing d'ici 10 ans. A suivre donc. . .