# Bases de Données I

Université de Mons

Jef Wijsen

2017–2018

# 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.

# Table des matières

# 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 :

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

Une *fonction totale* $f$ de $A$ vers $B$, dénotée $f : A \to B$, 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 $f : A \to B$ une fonction totale et $A' \subseteq A$. La *restriction* de $f$ à l'ensemble $A'$, dénoté par $f[A']$, est la fonction totale $f' : A' \to B$ 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).
>
> J.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* $A, B, C, A_1, B_1, C_1, \ldots$ ; 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 $t : \mathcal{A} \to \mathbf{dom}$ telle que pour tout attribut $A \in \mathcal{A}$, $t(A) \in Dom(A)$. 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 $Dom(\mathsf{Prénom}) = N$ et $Dom(\mathsf{Genre}) = \{♂, ♀\}$. Alors {Prénom, Genre} est un schéma-de-relation. Deux tuples sur ce schéma-de-relation sont $\{(\mathsf{Prénom}, \text{Jean}), (\mathsf{Genre}, ♂)\}$ et $\{(\mathsf{Prénom}, \text{Anne}), (\mathsf{Genre}, ♀)\}$. L'ensemble qui contient ces deux tuples est une relation. □

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 $\{(\mathsf{Prénom}, \text{Jean}), (\mathsf{Genre}, ♂)\}$, 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 $A$, $Dom(A) = \mathbf{dom}$.

Si $A_1, A_2, \ldots, A_\ell$ sont des attributs, on écrira souvent $A_1 A_2 \cdots A_\ell$ pour dénoter l'ensemble $\{A_1, A_2, \ldots, A_\ell\}$. 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 abrévié comme $XAB$.

## 2.2 Bases de données

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

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 $sorte(R)$. 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 {VINS, ABUS} avec $sorte(\mathsf{VINS}) = \{\mathsf{Cru}, \mathsf{Millésime}, \mathsf{Qualité}\}$ et $sorte(\mathsf{ABUS}) = \{\mathsf{Buveur}, \mathsf{Cru}, \mathsf{Millésime}\}$.

| VINS | Cru | Millésime | Qualité |
|---|---|---|---|
| | Volnay | 1983 | A |
| | Volnay | 1979 | B |
| | Chablis | 1983 | A |
| | Julienas | 1986 | C |

| ABUS | Buveur | Cru | Millésime |
|---|---|---|---|
| | Jean | Volnay | 1983 |
| | Jean | Volnay | 1979 |
| | Pierre | Volnay | 1979 |
| | 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 :

$$R \texttt{ PRIMARY KEY}(A_1, A_2, \ldots, A_k), \tag{2.1}$$

avec $A_1, A_2, \ldots, A_k$ des attributs distincts appartenant à $sorte(R)$. 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 $t_1[A_1A_2\cdots A_k] = t_2[A_1A_2\cdots A_k]$, 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 :

$$S \texttt{ FOREIGN KEY}(B_1, B_2, \ldots, B_k) \texttt{ REFERENCES } R, \tag{2.2}$$

avec $S$ un nom de relation et $B_1, B_2, \ldots, B_k$ des attributs appartenant à $sorte(S)$. 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 $i \in \{1, \ldots, k\}$, $s(B_i) = t(A_i)$, où les attributs $A_1, \ldots A_k$ 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 $i \in \{1, \ldots, k\}$). 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. □

| 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). □

## 2.4 La contrainte `UNIQUE`

La syntaxe de la contrainte `UNIQUE` est comme suit :

$R$ `UNIQUE`$(A_1, A_2, \ldots, A_k)$,

avec $R$ un nom de relation et $A_1, A_2, \ldots, A_k$ des attributs distincts appartenant à $sorte(R)$. 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 $t_1[A_1 A_2 \cdots A_k] = t_2[A_1 A_2 \cdots A_k]$, 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) `REFERENCESS` VOITURES

□

## 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

| 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.

généralement de remplacer la valeur absente par le symbole NULL. Dans la base de données de la 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., $\texttt{NULL} \notin \mathbf{dom}$), 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 $t_1(\text{Châssis}) = \texttt{NULL}$ et $t_2(\text{Châssis}) = \texttt{NULL}$, 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 $1, 2, 3, \ldots$

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 $\mathit{arité}(R) = |\mathit{sorte}(R)|$.

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 $\mathit{arité}(R)$.

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 $R[A_1, \ldots, A_n]$. Alors, un tuple "Named" $\{(A_1, a_1), \ldots, (A_n, a_n)\}$ correspond, un-à-un, à sa version "Unnamed" $\langle a_1, \ldots, a_n \rangle$.

# 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, $A \in \mathit{sorte}(E)$, et $a \in \mathbf{dom}$, alors $\sigma_{A=\text{“}a\text{”}}(E)$ est une expression algébrique avec $\mathit{sorte}(\sigma_{A=\text{“}a\text{”}}(E)) = \mathit{sorte}(E)$.

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

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

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

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

**Union** Si $E_1$ et $E_2$ sont des expressions algébriques avec $\mathit{sorte}(E_1) = \mathit{sorte}(E_2)$, alors $E_1 \uplus E_2$ est une expression algébrique avec $\mathit{sorte}(E_1 \uplus E_2) = \mathit{sorte}(E_1)$. Noter que le symbole $\uplus$ 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 $E_1 \uplus E_2$.

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

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 $[\![E]\!]^{\mathcal{I}}$ la relation qui est le résultat de l'évaluation de $E$ sur $\mathcal{I}$. La définition de $[\![E]\!]^{\mathcal{I}}$ est comme suit :

- Si $R$ est un nom de relation, alors $[\![R]\!]^{\mathcal{I}} := R^{\mathcal{I}}$. 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 $[\![R]\!]^{\mathcal{I}} \neq [\![R]\!]^{\mathcal{J}}$.
- $[\![\sigma_{A=\text{“}a\text{”}}(E)]\!]^{\mathcal{I}} := \{t \in [\![E]\!]^{\mathcal{I}} \mid t(A) = a\}$.
- $[\![\sigma_{A=B}(E)]\!]^{\mathcal{I}} := \{t \in [\![E]\!]^{\mathcal{I}} \mid t(A) = t(B)\}$.
- $[\![\pi_X(E)]\!]^{\mathcal{I}} := \{t[X] \mid t \in [\![E]\!]^{\mathcal{I}}\}$.
- Soit $X = sorte(E_1)$ et $Y = sorte(E_2)$.
  Alors $[\![E_1 \bowtie E_2]\!]^{\mathcal{I}} := \{t \mid t[X] \in [\![E_1]\!]^{\mathcal{I}} \text{ et } t[Y] \in [\![E_2]\!]^{\mathcal{I}}\}$.
- Soit $X = sorte(E) \setminus \{A\}$.
  Alors $[\![\rho_{A \mapsto B}(E)]\!]^{\mathcal{I}} := \{t \mid$ il existe $s \in [\![E]\!]^{\mathcal{I}}$ tel que $s[X] = t[X]$ et $s[A] = t[B]\}$.
- $[\![E_1 \uplus E_2]\!]^{\mathcal{I}} := [\![E_1]\!]^{\mathcal{I}} \cup [\![E_2]\!]^{\mathcal{I}}$, avec $\cup$ l'union ensembliste.
- $[\![E_1 - E_2]\!]^{\mathcal{I}} := [\![E_1]\!]^{\mathcal{I}} \setminus [\![E_2]\!]^{\mathcal{I}}$, avec $\setminus$ la différence ensembliste.

Une expression algébrique $E$ est aussi appelée une *requête*, et $[\![E]\!]^{\mathcal{I}}$ 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

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

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

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

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

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

Cependant, cette substitution nuit à la lisibilité.

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

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

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

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

**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 $E_4$ rend la relation $\{\{\}\}$, i.e., la relation qui contient le tuple vide. S'il n'existe pas de buveur exigeant, alors l'évaluation de $E_4$ rend 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

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

La requête finale $E_2$ est comme suit :

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

## 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 inclue dans $E_2$, dénoté par $E_1 \sqsubseteq E_2$, si pour toute base de données $\mathcal{I}$, $[\![E_1]\!]^{\mathcal{I}} \subseteq [\![E_2]\!]^{\mathcal{I}}$.

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 $\mathit{sorte}(R) = \mathit{sorte}(S)$. Soit $X \subseteq \mathit{sorte}(R)$. On peut facilement prouver que $\pi_X(R \cup S) \equiv \pi_X(R) \cup \pi_X(S)$. □

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.** *L'équivalence des expressions algébriques en SPJRUD est indécidable.*

| $R$ | $A$ | $B$ |
|---|---|---|
| | $a$ | 1 |

| $R$ | $A$ | $B$ |
|---|---|---|
| | $a$ | 1 |
| | $a$ | 2 |

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 SPJRUD n'est redondant. Démontrons ici que la différence est un opérateur non-redondant. Dénotons 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 $[\![E_0]\!]^{\mathcal{I}} = \{\{(A, a)\}\}$ et $[\![E_0]\!]^{\mathcal{J}} = \{\}$. Remarquons que $\mathcal{J}$ ajoute un tuple à $\mathcal{I}$, mais que $[\![E_0]\!]^{\mathcal{J}} \subsetneq [\![E_0]\!]^{\mathcal{I}}$. 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, $[\![E]\!]^{\mathcal{I}} \subseteq [\![E]\!]^{\mathcal{J}}$. À partir de $[\![E]\!]^{\mathcal{I}} \subseteq [\![E]\!]^{\mathcal{J}}$ et $[\![E_0]\!]^{\mathcal{J}} \subsetneq [\![E_0]\!]^{\mathcal{I}}$, il est correct de conclure que soit $[\![E]\!]^{\mathcal{I}} \neq [\![E_0]\!]^{\mathcal{I}}$, soit $[\![E]\!]^{\mathcal{J}} \neq [\![E_0]\!]^{\mathcal{J}}$ (soit les deux), et donc $E \not\equiv E_0$.

On prouve maintenant, par induction, que pour toute expression algébrique $E$ de l'algèbre SPJRU, $[\![E]\!]^{\mathcal{I}} \subseteq [\![E]\!]^{\mathcal{J}}$. 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 $[\![R]\!]^{\mathcal{I}} = R^{\mathcal{I}}$ est un sous-ensemble de $[\![R]\!]^{\mathcal{J}} = R^{\mathcal{J}}$. Les autres possibilités pour $E$ sont comme suit :

- $E$ est une sélection $\sigma_{C=\text{“c”}}(E_1)$. Noter que même si $\mathit{sorte}(R) = \{A, B\}$, il est possible que $C \in \mathit{sorte}(E_1)$, car $E_1$ peut contenir un renommage. L'hypothèse d'induction est $[\![E_1]\!]^{\mathcal{I}} \subseteq [\![E_1]\!]^{\mathcal{J}}$, 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., $[\![\sigma_{C=\text{“c”}}(E_1)]\!]^{\mathcal{I}} \subseteq [\![\sigma_{C=\text{“c”}}(E_1)]\!]^{\mathcal{J}}$.
- $E$ 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 $[\![E_1]\!]^{\mathcal{I}} \subseteq [\![E_1]\!]^{\mathcal{J}}$ et $[\![E_2]\!]^{\mathcal{I}} \subseteq [\![E_2]\!]^{\mathcal{J}}$. Il est alors évident que $[\![E_1 \bowtie E_2]\!]^{\mathcal{I}} \subseteq [\![E_1 \bowtie E_2]\!]^{\mathcal{J}}$.
- $E$ 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 monotonicité 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 $[\![E]\!]^{\mathcal{I}} \subseteq [\![E]\!]^{\mathcal{J}}$. 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 $\mathit{sorte}(R) = \{A, B\}$. Soit

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

L'expression $E_1$ rend 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 $arité(R)$.

**Sélection "attribut égal constante"** Si $E$ est une expression algébrique d'arité $n$, $1 \leq i \leq n$, et $a$ une constante, alors $\sigma_{i=\text{“}a\text{”}}(E)$ 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 $1 \leq i_1, i_2, \ldots, i_k \leq n$, alors $\pi_{i_1,i_2,\ldots,i_k}(E)$ est une expression algébrique d'arité $k$.

**Produit cartésien** Si $E$ et $F$ sont des expressions algébriques d'arités $m$ en $n$, 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 $[\![E]\!]^{\mathcal{I}}$ est comme suit :

— Si $R$ est un nom de relation, alors $[\![R]\!]^{\mathcal{I}} = R^{\mathcal{I}}$.

— $[\![\sigma_{i=\text{“}a\text{”}}(E)]\!]^{\mathcal{I}} = \{\langle a_1, a_2, \ldots, a_n\rangle \in [\![E]\!]^{\mathcal{I}} \mid a_i = a\}$.

— $[\![\sigma_{i=j}(E)]\!]^{\mathcal{I}} = \{\langle a_1, a_2, \ldots, a_n\rangle \in [\![E]\!]^{\mathcal{I}} \mid a_i = a_j\}$.

— $[\![\pi_{i_1,i_2,\ldots,i_k}(E)]\!]^{\mathcal{I}} = \{\langle a_{i_1}, a_{i_2}, \ldots, a_{i_k}\rangle \mid \langle a_1, a_2, \ldots, a_n\rangle \in [\![E]\!]^{\mathcal{I}}\}$.

— $[\![E \times F]\!]^{\mathcal{I}} = \{\langle a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n\rangle \mid \langle a_1, \ldots, a_m\rangle \in [\![E]\!]^{\mathcal{I}}, \langle b_1, \ldots, b_n\rangle \in [\![F]\!]^{\mathcal{I}}\}$.

— $[\![E \cup F]\!]^{\mathcal{I}} = [\![E]\!]^{\mathcal{I}} \cup [\![F]\!]^{\mathcal{I}}$, où $\cup$ est l'union ensembliste.

— $[\![E - F]\!]^{\mathcal{I}} = [\![E]\!]^{\mathcal{I}} \setminus [\![F]\!]^{\mathcal{I}}$, où $\setminus$ est la différence ensembliste.

### 3.7.3 Exemple

La requête $E_3$ retourne les buveurs exigeants :

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

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

$R$

| 1 | 2 |
|---|---|
| $a$ | $b$ |
| $c$ | $d$ |

$\pi_{2,1,1}(R)$

| 1 | 2 | 3 |
|---|---|---|
| $b$ | $a$ | $a$ |
| $d$ | $c$ | $c$ |

# 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 :

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

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 dés le début l'importance de la partie droite. Substituons $b$ par la valeur "Senseo". [1] Comme Senseo n'est pas un buveur, la prémisse $\mathsf{ABUS}(\text{“Senseo”}, c, m)$ 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 $\exists c' \exists m' \mathsf{ABUS}(\text{“Senseo”}, c', m')$ est fausse, la conjonction sera finalement fausse.

## 4.1 Syntaxe

Supposons un nombre infini de *variables* $x, y, z, x_1, y_1, z_1 \ldots$ [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 $e_1, \ldots, e_k$ son des termes, alors $R(e_1, \ldots, e_k)$ est une formule (atomique).
- Si $\varphi_1$ et $\varphi_2$ sont des formules, alors $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ et $\neg\varphi_1$ sont des formules.

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 $\varphi$ est une formule et $x$ une variable, alors $\exists x\varphi$ et $\forall x\varphi$ sont des formules.

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

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

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

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

## 4.2 Sémantique

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

De façon plus formelle, le *domaine actif* d'une base de données $\mathcal{I}$, dénoté par $\mathit{adom}(\mathcal{I})$, 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

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

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

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

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

Le tuple $\vec{a}$ donne lieu à une *valuation* $\nu : \mathbf{d} \cup \{x_1, \ldots, x_n\} \to \mathbf{d}$ telle que pour chaque $i \in \{1, \ldots, n\}$, $\nu(x_i) = a_i$ 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 $\nu(t_1) = \nu(t_2)$ ;
— $\mathcal{I}, \mathbf{d} \models R(e_1, \ldots, e_k)$ si $\langle \nu(e_1), \ldots, \nu(e_k)\rangle \in R^{\mathcal{I}}$ ;
— 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 $\mathcal{I} \not\models \varphi_1(\vec{a})$ ;

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

— 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

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

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

Une légère extension consiste à permettre que la même variable apparaisse plusieurs fois à gauche de la bar verticale. Par exemple (voir aussi 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 teste $\mathcal{I}, \mathbf{d} \models \exists y \psi(y, \vec{a})$, on cherche la valeur de $y$ dans $\mathbf{d}$ ; et
3. dans le teste $\mathcal{I}, \mathbf{d} \models \forall y \psi(y, \vec{a})$, 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 :

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

Comme expliqué dans l'introduction de ce chapitre, la formule à droite de la bar 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 :

$$\{c \mid \forall m \mathsf{VINS}(c, m, \text{“A”})\}. \qquad \text{☹2}$$

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 $\mathsf{VINS}(c, m, \text{“A”})$ 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 sera toujours vide. Une solution pourrait consister à restreindre le domaine de la variable $m$ a 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

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

domaine de $m$. La requête suivante exige que $c$ a été de qualité A dans *tout millésime $m$ pour lequel la qualité de $c$ a été enregistré :*

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

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 *:*

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

Comme déjà illustré par la requête ☹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 :

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

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 $adom(\mathcal{I}) \subseteq \mathbf{d}_1, \mathbf{d}_2 \subseteq \mathbf{dom}$, 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 $adom(\mathcal{I})$ comme domaine d'interprétation.

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

**Exemple avec disjonction**

$$\{x \mid R(a) \vee R(b) \vee R(x)\} \qquad ☹4$$

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

$R$

| $A$ |
|---|
| $a$ |
| $c$ |

.

On a $adom(\mathcal{I}_1) = \{a, c\}$. 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 ☹4 n'est donc pas indépendante du domaine.

**Exemple avec négation**

$$\{x \mid \neg R(x)\} \qquad \text{☹5}$$

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

$$R \begin{array}{|c} \hline A \\ \hline a \\ \hline \end{array}.$$

On a $adom(\mathcal{I}_2) = \{a\}$. 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**

$$\{x \mid \forall y R(x, y)\} \qquad \text{☹6}$$

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

$$R \begin{array}{|cc} \hline A & B \\ \hline a & a \\ a & b \\ \hline \end{array}.$$

On a $adom(\mathcal{I}_3) = \{a, b\}$. 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 sur 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 $x_1, \ldots, x_n$ sont des variables distinctes, alors $R(x_1, \ldots, x_n)$ est une formule safe (atomique). La restriction que $x_1, \ldots, x_n$ soient distinctes fait que chaque variable correspond à exactement un attribut de $sorte(R)$.

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

**Projection** Si $\varphi$ est une formule safe et $x \in free(\varphi)$, alors $\exists x \varphi$ est une formule safe.

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

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

**Différence** Si $\varphi_1$ et $\varphi_2$ sont des formules safe avec $free(\varphi_1) = free(\varphi_2)$, alors $\varphi_1 \wedge \neg\varphi_2$ est une formule safe. La restriction que $\varphi_1$ et $\varphi_2$ aient exactement les mêmes variables libres provient de la restriction que la différence $E_1 - E_2$ est seulement définie si $sorte(E_1) = sorte(E_2)$. Intuitivement, la formule $\varphi_1$ est un garde-fou qui fixe le domaine des variables libres. Noter que la formule dans ☹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 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. Considérer une requête de la forme :

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

avec $\varphi_1$ et $\varphi_2$ des formules booléennes (i.e., sans variables libres) en SRC, et $\psi$ 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 $\varphi_1$ et $\varphi_2$ 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 $R(x, \text{“}a\text{”})$ par $R(x, y) \wedge (y = \text{“}a\text{”})$ ; remplacer $R(x, x)$ par $R(x, y) \wedge (y = x)$ ;

— remplacer chaque sous-formule $x = \text{“}a\text{”}$ de $\psi$ par $\alpha(x) \wedge (x = \text{“}a\text{”})$ ;

— remplacer chaque sous-formule $x = y$ de $\psi$ par $(\alpha(x) \wedge \alpha(y)) \wedge (x = y)$ ;

— remplacer chaque négation $\neg\varphi(x_1, \ldots, x_n)$ par $(\alpha(x_1) \wedge \cdots \wedge \alpha(x_n)) \wedge \neg\varphi(x_1, \ldots, x_n)$ ;

— 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 dénotations 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 suivant schéma-de-bases-de-données, pris 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 sait fournir quel produit, et en quelle quantité. Chaque quantité est en nombre entier strictement positif.

**Requête 7.7.4**

Trouver tout couple de villes tel qu'un fournisseur habitant la première ville sait 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 SRC :

$$\{s.4, p.5 \mid S(s) \wedge \mathsf{P}(p) \wedge \exists r\,(\mathsf{SP}(r) \wedge s.1 = r.1 \wedge r.2 = p.1)\}$$
$$\equiv$$
$$\{s.4, p.5 \mid \exists r\,(\mathsf{S}(s) \wedge \mathsf{P}(p) \wedge \mathsf{SP}(r) \wedge s.1 = r.1 \wedge r.2 = p.1)\}$$

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

$$\{s.\mathsf{City}, p.\mathsf{City} \mid S(s) \wedge \mathsf{P}(p) \wedge \exists r\,(\mathsf{SP}(r) \wedge s.\mathsf{S\sharp} = r.\mathsf{S\sharp} \wedge r.\mathsf{P\sharp} = p.\mathsf{P\sharp})\}$$
$$\equiv$$
$$\{s.\mathsf{City}, p.\mathsf{City} \mid \exists r\,(\mathsf{S}(s) \wedge \mathsf{P}(p) \wedge \mathsf{SP}(r) \wedge s.\mathsf{S\sharp} = r.\mathsf{S\sharp} \wedge r.\mathsf{P\sharp} = p.\mathsf{P\sharp})\}$$

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 savent 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 SRC :

$$\{s.2 \mid S(s) \wedge \forall p \in P\,(p.3 = \text{“red”} \rightarrow \exists r \in \mathsf{SP}\,(r.1 = s.1 \wedge r.2 = p.1))\}$$
$$\equiv$$
$$\{s.2 \mid S(s) \wedge \neg\exists p \in P\,(p.3 = \text{“red”} \wedge \neg\exists r \in \mathsf{SP}\,(r.1 = s.1 \wedge r.2 = p.1))\}$$

Avec les attributs :

$$\{s.\mathsf{Sname} \mid S(s) \wedge \forall p \in P\,(p.\mathsf{Color} = \text{“red”} \rightarrow \exists r \in \mathsf{SP}\,(r.\mathsf{S\sharp} = s.\mathsf{S\sharp} \wedge r.\mathsf{P\sharp} = p.\mathsf{P\sharp}))\}$$
$$\equiv$$
$$\{s.\mathsf{Sname} \mid S(s) \wedge \neg\exists p \in P\,(p.\mathsf{Color} = \text{“red”} \wedge \neg\exists r \in \mathsf{SP}\,(r.\mathsf{S\sharp} = s.\mathsf{S\sharp} \wedge r.\mathsf{P\sharp} = p.\mathsf{P\sharp}))\}$$

### 4.6.2 Syntaxe

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

- Chaque constante $c \in \mathbf{dom}$ est un terme.
- Si $t$ est une variable d'arité $n$ et $i \in \{1, 2, \ldots, n\}$, 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 $\varphi_1$ et $\varphi_2$ sont des formules, alors $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ et $\neg\varphi_1$ sont des formules.
- Si $\varphi$ 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 $\varphi$ une formule telles que

— pour chaque terme $r.i$ dans $L$, la variable $r$ est une variable libre de $\varphi$ ; et

— chaque variable libre de $\varphi$ apparaît dans un terme de $L$.

**Abréviations** On introduit un nombre d'abréviations :

— $\varphi_1 \to \varphi_2$ 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 $\forall r\,(R(r) \to \varphi)$ ;

— $u_1 \neq u_2$ est une abréviation de $\neg\,(u_1 = u_2)$.

Noter que ces abréviations sont cohérentes :

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

### 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 $r \in \mathbf{d}^n$ qui rend $\varphi$ vrai, avec $n$ l'arité de $r$.

— $\forall r\varphi$ est vrai si pour tout tuple $r \in \mathbf{d}^n$, $\varphi$ 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 :

$$\{r.1 \mid R(r) \vee \exists s\,(S(s))\}; \qquad ☹$$

$$\{r.1 \mid \neg R(r)\}. \qquad ☹$$

SQL est un mixe 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 des 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 :

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

Avec les attributs :

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

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 |
|---|---|---|---|
| S1 | Smith | 20 | London |
| S2 | Jones | 10 | Paris |
| S3 | Blake | 30 | Paris |
| S4 | Clark | 20 | London |
| S5 | Adams | 30 | Athens |

P

| P# | PNAME | COLOR | WEIGHT | CITY |
|---|---|---|---|---|
| P1 | Nut | Red | 12 | London |
| P2 | Bolt | Green | 17 | Paris |
| P3 | Screw | Blue | 17 | Rome |
| P4 | Screw | Red | 14 | London |
| 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 livre (=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 quadrimeste. 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 | Méchanique 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 engendrait un conflit horaire pour Albert Einstein.

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

$\mathsf{Cours} \to \mathsf{Prof}$ **:** À chaque cours ne peut correspondre qu'un seul professeur.

$\mathsf{Prof}, \mathsf{Heure} \to \mathsf{Cours}$ **:** 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 : $\mathsf{Cours}, \mathsf{Heure} \to \mathsf{Prof}$.

## 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 $X \to Y$ avec $X, Y \subseteq \mathcal{A}$.

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

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

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

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 11 illustrera que l'implication "finie" peut être différente de l'implication "non restreinte".

**Exemple 7.** Soit $\mathcal{A} = \{A, B, C\}$. On va démonter que $A \to C$ est une conséquence logique de $\{A \to B,\ B \to C\}$. Dans ce but, soit $R$ une relation quelconque qui respecte $A \to B$ et $B \to C$. 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 $R \models A \to B$, il est correct de conclure $s(B) = t(B)$. Ensuite, à partir de $s, t \in R$, $s(B) = t(B)$, et $R \models B \to C$, il est correct de conclure $s(C) = t(C)$. □

Soit $\Sigma_1, \Sigma_2$ deux ensembles de DFs sur $\mathcal{A}$. On écrira $\Sigma_1 \models \Sigma_2$ si pour toute DF $X \to Y$ dans $\Sigma_2$, on a $\Sigma_1 \models X \to Y$. On dira que $\Sigma_1$ et $\Sigma_2$ 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 $\{A \to B, B \to C\}$ et $\{A \to B, B \to C, A \to C\}$ sont équivalents. □

**Proposition 3** (Theorème de Heath)**.** *Soit $R$ une relation sur $\mathcal{A}$ telle que $R \models X \to Y$ 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, ils existent 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 $R \models X \to Y$, 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 $t \in R$. □

## 6.3 Fermeture d'un ensemble d'attributs

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

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

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

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 $\Sigma \models X \to Y$ si et seulement si $Y \subseteq X^{*,\Sigma}$.

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

**Algorithme 1.**

*Entrée : un ensemble $\Sigma$ de DFs, un ensemble $X$ d'attributs*
*Sortie : $X^{*,\Sigma}$*

`NonUtilisé` $:= \Sigma$
`Fermeture` $:= X$
*répéter aussi longtemps que* `NonUtilisé` *change*
  *si* $W \to Z \in$ `NonUtilisé` *et* $W \subseteq$ `Fermeture`
  *alors i.* `NonUtilisé` $:=$ `NonUtilisé` $\setminus \{W \to Z\}$
    *ii.* `Fermeture` $:=$ `Fermeture` $\cup Z$
*retourner* `Fermeture`

**Proposition 4.** *Avec $\Sigma$ est $X$ 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, $X^{*,\Sigma} \subseteq résultat$, soit $A$ un attribut quelconque tel que $A \notin résultat$. Il faut montrer que $A \notin X^{*,\Sigma}$, i.e., $\Sigma \not\models X \to A$. Construisons une relation $R$ comme suit :

| $\overbrace{\quad}^{\text{attributs de } résultat}$ | | | $\overbrace{\quad}^{\text{autres attributs}}$ | | |
|---|---|---|---|---|---|
| $A_1$ | $\dots$ | $A_\ell$ | $A_{\ell+1}$ | $\dots$ | $A_n$ |
| 0 | $\cdots$ | 0 | 0 | $\cdots$ | 0 |
| 0 | $\cdots$ | 0 | 1 | $\cdots$ | 1 |

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

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

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

## 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

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

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 | $b$ | $c$ | $\alpha$ |
| | 1 | $b$ | $c$ | $\beta$ |
| | 2 | $b$ | $c$ | $\alpha$ |
| | 2 | $b$ | $c$ | $\beta$ |
| | 3 | $b$ | $d$ | $\beta$ |
| | 3 | $b$ | $d$ | $\gamma$ |

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 $BC \twoheadrightarrow D$).

$\pi_{ABC}(R)$

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | $b$ | $c$ |
| 2 | $b$ | $c$ |
| 3 | $b$ | $d$ |

$\pi_{BCD}(R)$

| $B$ | $C$ | $D$ |
|---|---|---|
| $b$ | $c$ | $\alpha$ |
| $b$ | $c$ | $\beta$ |
| $b$ | $d$ | $\beta$ |
| $b$ | $d$ | $\gamma$ |

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 $R$. □

La proposition suivante exprime une interaction entre les DF et DMV.

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

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

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

| $\overbrace{\quad}^{\text{attributs de } X^{*,\Sigma}}$ | | | | | $\overbrace{\quad}^{\text{autres attributs}}$ | | | |
|---|---|---|---|---|---|---|---|---|
| $A_1$ | $\ldots$ | $A_\ell$ | $B$ | $C$ | $A_{\ell+1}$ | $\ldots$ | $A_n$ | |
| 0 | $\cdots$ | 0 | 0 | 0 | 0 | $\cdots$ | 0 | $(t_0)$ |
| 0 | $\cdots$ | 0 | 1 | 1 | 1 | $\cdots$ | 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 $t(C) = 1$. □

### 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

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

où

— $k \geq 1$ ;
— $R$ et $S$ sont des noms de relation dans $\mathcal{R}$ (il se peut que $R = S$) ;
— $A_1, \ldots, A_k$ sont des attributs distincts dans $sorte(R)$ ;
— $B_1, \ldots, B_k$ sont des attributs distincts dans $sorte(S)$.
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 $i \in \{1, \ldots, k\}$, $r(A_i) = s(B_i)$.

**Exemple 10.** Soit $\mathcal{I}$ la base de données contenant les relations suivantes :

$R$

| $A$ | $B$ |
|---|---|
| 1 | 2 |
| 2 | 3 |
| 3 | 2 |

$S$

| $C$ | $D$ |
|---|---|
| 2 | 1 |
| 2 | 3 |

Cette base de données satisfait les DIs 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 11.** Soit $\mathcal{R} = \{R\}$ un schéma-de-base-de-données avec $sorte(R) = \{A, B\}$. Soit $\Sigma = \{A \to B, R[A] \subseteq R[B]\}$. 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 $\Sigma$. Puisque $\mathcal{I}$ est finie, il existe un nombre $n$ tel que $|R^{\mathcal{I}}| = n$. Puisque $\mathcal{I}$ satisfait $A \to B$, 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 $\Sigma$ mais qui ne satisfait pas $R[B] \subseteq R[A]$ :

$R$

| $A$ | $B$ |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 3 | 2 |
| 4 | 3 |
| ⋮ | ⋮ |

□

Dans la section 6.3, on a vu qu'il existe un algorithme qui prend en entrée un ensemble $\Sigma$ de DFs et une DF $\sigma$, 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 $\Sigma$ peut contenir des DIs en plus des DFs.

**Proposition 6.** *Le problème suivant est indécidable : pour une DF $\sigma$ et un ensemble $\Sigma$ contenant des DFs et des IDs, 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 Cours → 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 | Méchanique 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 Cours → 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 $\Sigma$ 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 $\Sigma$, 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 $X \to Y \in \Sigma$ telle que $Y \not\subseteq X$, on a $\Sigma \models X \to \mathcal{A}$.

**Exemple 12.** Soit $\mathcal{A} = \{\mathsf{Prof}, \mathsf{Cours}, \mathsf{Heure}\}$ avec $\Sigma = \{\mathsf{Cours} \to \mathsf{Prof},\ \{\mathsf{Prof}, \mathsf{Heure}\} \to \mathsf{Cours}\}$. Ce schéma-DF n'est pas en BCNF car $\mathsf{Cours} \to \mathsf{Prof}$ appartient à $\Sigma$, mais $\Sigma \not\models \mathsf{Cours} \to \mathsf{Heure}$. □

**Exemple 13.** Soit $\mathcal{A} = \{\mathsf{Étudiant}, \mathsf{Dép}, \mathsf{Fac}\}$ avec $\Sigma = \{\mathsf{Étudiant} \to \mathsf{Dép},\ \mathsf{Dép} \to \mathsf{Fac}\}$. Le schéma-DF $(\mathcal{A}, \Sigma)$ n'est pas en BCNF car $\mathsf{Dép} \to \mathsf{Fac}$ appartient à $\Sigma$ et $\Sigma \not\models \mathsf{Dép} \to \mathsf{Étudiant}$. 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 |
|---|---|
| Anne | Info |
| Bert | Info |
| Carl | Info |
| Doris | Math |
| Els | Math |
| Fred | Marketing |

| Dép | Fac |
|---|---|
| Info | Sciences |
| Math | Sciences |
| Marketing | FWEG |

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. □

## 7.3 Décompositions

La définition suivante formalise la notion de décomposition.

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

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

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. □

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 $\Sigma_i$ encode tout DF sur $\mathcal{A}_i$ qui est une conséquence logique de $\Sigma$.

**Exemple 14.** Continuation de l'exemple 13. Soit $\mathcal{A} = \{\mathsf{Étudiant}, \mathsf{Dép}, \mathsf{Fac}\}$ avec $\Sigma = \{\mathsf{Étudiant} \rightarrow \mathsf{Dép}, \mathsf{Dép} \rightarrow \mathsf{Fac}\}$. Sur la base de Proposition 5, on peut conclure :

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

Introduisons trois schémas-DF :

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

Alors $\{ED, EF\}$ et $\{ED, DF\}$ sont deux décompositions. La première décomposition "perd" la DF $\mathsf{Dép} \rightarrow \mathsf{Fac}$. 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 $\Sigma \not\models \bowtie [\{\mathsf{Étudiant}, \mathsf{Fac}\}, \{\mathsf{Dép}, \mathsf{Fac}\}]$. □

## 7.4 Décomposition en BCNF

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

**Proposition 7.** *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 $\Sigma_1$ et $\Sigma_2$ 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 $(XA, \Sigma_1)$ et/ou $(XZ, \Sigma_2)$ 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. □

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

**Exemple 15.** Continuation de l'exemple 12. Soit $\mathcal{A} = \{\mathsf{Prof}, \mathsf{Cours}, \mathsf{Heure}\}$ avec $\Sigma = \{\mathsf{Cours} \rightarrow \mathsf{Prof}, \{\mathsf{Prof}, \mathsf{Heure}\} \rightarrow \mathsf{Cours}\}$. 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 :

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

Introduisons deux schémas-DF :

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

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

## 7.5 3NF

L'exemple 15 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 $A \in \mathcal{A}$, $\Sigma \models K \to A$. Un attribut $A$ est appelé *premier* s'il appartient à une ou plusieurs clés.

**Exemple 16.** Soit $\mathcal{A} = ABCDE$ et $\Sigma = \{A \to BCD, BC \to AD\}$. Les clés sont $AE$ et $BCE$. Les attributs premiers sont $A$, $B$, $C$ et $E$. □

Un schéma-DF $(\mathcal{A}, \Sigma)$ est en 3NF si pour toute DF $X \to Y \in \Sigma$ telle que $Y \not\subseteq X$, soit $\Sigma \models X \to \mathcal{A}$, 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 $\Sigma \models X \to A$ et $\Sigma \not\models X \to \mathcal{A}$. 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 15 où la décomposition $\{CP, CH\}$ divise la clé $\{\mathsf{Prof}, \mathsf{Heure}\}$ 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 $\Sigma \models X \to A$ et $\Sigma \models XA \to \mathcal{A}$ (car $XA$ contient une clé), on dérive $\Sigma \models X \to \mathcal{A}$, 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 $(XA, \Sigma_1)$ et $(XBZ, \Sigma_2)$ 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 8.** *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 15. 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 17.** Soit $\mathcal{A} = ABCD$ et $\Sigma = \{AC \to D, CD \to A\}$. 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 $\Sigma \models AC \to D$ mais $\Sigma \not\models AC \to B$.

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

$(ACD, \{AC \to D, CD \to A\})$
$(ABC, \emptyset)$

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 $\{AC \to D, CD \to A\} \models BCD \to ABCD$. □

## 7.6 Dépendances fonctionnelles singulières

Disons qu'une DF $X \to Y$ 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 $\Sigma$ est singulière. Alors,

— $(\mathcal{A}, \Sigma)$ est en BCNF si pour tout $X \to A$ dans $\Sigma$, on a $\Sigma \models X \to \mathcal{A}$.

— $(\mathcal{A}, \Sigma)$ est en 3NF si pour tout $X \to A$ dans $\Sigma$, soit $\Sigma \models X \to \mathcal{A}$, 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 $\Sigma$ est singulière.
2. Pour chaque DF $X \to A \in \Sigma$, aucun attribut de $X$ n'est superflu. De façon formelle, pour chaque DF $X \to A \in \Sigma$ et $B \in X$,
$$(\Sigma \setminus \{X \to A\}) \cup \{(X \setminus B) \to A\} \not\equiv \Sigma.$$
3. Aucune DF de $\Sigma$ n'est redondante. De façon formelle, pour chaque DF $X \to A \in \Sigma$, $(\Sigma \setminus \{X \to A\}) \not\equiv \Sigma$.

**Exemple 18.** $\Sigma_1 = \{A \to C, AB \to C\}$ n'est pas irréductible, parce que $\{A \to C\} \equiv \Sigma$.

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

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 19.** Soit $\mathcal{A} = ABC$ et $\Sigma = \{A \to B,\ B \to AC, A \to C\}$. Soient

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

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

Proposition 8 est une conséquence logique de la proposition suivante.

**Proposition 9.** *Soit $(\mathcal{A}, \Sigma)$ un schéma-DF irréductible avec $\Sigma = \{X_1 \to A_1, \ldots, X_\ell \to A_\ell\}$. Soit $K$ une clé pour $(\mathcal{A}, \Sigma)$. L'ensemble suivant est une décomposition de $(\mathcal{A}, \Sigma)$ en 3NF qui préserve les DFs :*

$$\{(K, \Sigma_0), (X_1A_1, \Sigma_1), \ldots, (X_\ell A_\ell, \Sigma_\ell)\},$$

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

Il est trivial de voir que la décomposition de la Proposition 9 préserve les DFs. Il est plus difficile de démontrer que $\Sigma \models\!\bowtie [K, X_1, \ldots, X_\ell]$ et que chaque composant est en 3NF.

**Exemple 20.** Cet exemple montre l'importance du composant $(K, \Sigma_0)$ dans l'énoncé de la Proposition 9. Soit $\mathcal{A} = \{\mathsf{Étudiant}, \mathsf{Dép}, \mathsf{Langue}\}$. Un tuple $\{(\mathsf{Étudiant}, \text{“Ed”}), (\mathsf{Dép}, \text{“Info”}), (\mathsf{Langue}, \text{“espagnol”})\}$ signifie que Ed est un étudiant au département d'Info qui sait s'exprimer en espagnol. L'ensemble des DF est le singleton $\Sigma = \{\mathsf{Étudiant} \to \mathsf{Dép}\}$. En conséquence, $\{\mathsf{Étudiant}, \mathsf{Langue}\}$ est la seule clé. Le schéma-DF $(\mathcal{A}, \Sigma)$ n'est pas en 3NF, parce que $\mathsf{Dép}$ n'est pas premier est un étudiant peut maîtriser plusieurs langues, i.e., $\Sigma \not\models \mathsf{Étudiant} \to \mathsf{Langue}$. L'application de la Proposition 9 nous donne la décomposition suivante en 3NF :

$$\{(\{\mathsf{Étudiant}, \mathsf{Langue}\}, \emptyset), (\{\mathsf{Étudiant}, \mathsf{Dép}\}, \{\mathsf{Étudiant} \to \mathsf{Dép}\})\}.$$

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

Le composant $(\{\mathsf{Étudiant}, \mathsf{Langue}\}, \emptyset)$ 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 $(\{\mathsf{Étudiant}, \mathsf{Dép}\}, \{\mathsf{Étudiant} \to \mathsf{Dép}\})$ 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. □

## 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] ☺ :

> *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 $\Xi$ un ensemble qui contient des DFs et DMVs sur $\mathcal{A}$.

Un schéma-DF-DMV $(\mathcal{A}, \Xi)$ est en 4NF si les deux conditions suivantes sont satisfaites pour tout $X, Y \subseteq \mathcal{A}$ :

1. si $\Xi \models X \to Y$ avec $Y \not\subseteq X$, alors $\Xi \models X \to \mathcal{A}$ ; et
2. si $\Xi \models X \twoheadrightarrow Y$ avec $Y \not\subseteq X$, alors $\Xi \models X \to \mathcal{A}$.

La première condition est égale à celle de BCNF. En s'appuyant sur la Proposition 3, on peut affirmer :

> Un schéma-DF-DMV $(\mathcal{A}, \Xi)$ est en 4NF si et seulement s'il existe un schéma-DF [1] $(\mathcal{A}, \Sigma)$ en BCNF tel que $\Sigma \equiv \Xi$.

**Exemple 21.** Soit $\mathcal{A} = \{\mathsf{Étudiant}, \mathsf{Langage}, \mathsf{Plat}\}$. Un tuple $\{(\mathsf{Étudiant}, \text{“Ed”}), (\mathsf{Langage}, \text{“Java”}), (\mathsf{Plat}, \text{“spaghetti”})\}$ signifie qu'Ed est un étudiant qui sait programmer en Java et qui sait préparer un spaghetti.

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

La table suivante satisfait $\Xi$, mais ne satisfait aucune DF singulière. En conséquence, aucun ensemble de DFs ne peut être équivalent à $\Xi$, 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 $\{\mathsf{Étudiant}, \mathsf{Langage}\}$ et $\{\mathsf{Étudiant}, \mathsf{Plat}\}$. Intuitivement, 4NF nous dit de ne jamais stocker, dans une même relation, deux renseignements n'ayant rien à voir l'un avec l'autre. □

### 7.8.3 5NF

Pour rappel, chaque DMV est une DJ. Cependant, il existe des DJs $\bowtie [X_1, \ldots, X_\ell]$ avec $\ell \geq 3$ que l'on ne peut pas exprimer comme DMV (voir exemple 23). La notion de 5NF est une généralisation de 4NF qui prend en compte aussi les DJs qui ne sont pas des DMVs.

Un schéma-DF-DJ est un couple $(\mathcal{A}, \Xi)$ avec $\mathcal{A}$ un ensemble d'attributs et $\Xi$ 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 $\Sigma \equiv \Xi$.

1. Donc, tout élément de $\Sigma$ est une DF.

**Exemple 22.** Soient $\mathcal{A} = ABCD$ et $\Xi = \{A \to BCD, \bowtie [AB, AC, AD]\}$. Il est facile de démontrer (cf. Proposition 3) que $\{A \to BCD\} \models\!\!\bowtie [AB, AC, AD]$. Donc, $\Xi \equiv \{A \to BCD\}$. Il est correct de conclure que le schéma-DF-DJ $(\mathcal{A}, \Xi)$ est en 5NF. □

**Exemple 23.** Exemple issu de [9]. Soit $\mathcal{A} = \{\mathsf{Agent}, \mathsf{Company}, \mathsf{Product}\}$. Un tuple $\{(\mathsf{Agent}, \text{“Smith”}), (\mathsf{Company}, \text{“Ford”}), (\mathsf{Product}, \text{“car”})\}$ signifie que Smith est un agent qui vend des voitures pour la marque Ford.

Soit $\Xi = \{\bowtie [\{\mathsf{Agent}, \mathsf{Company}\}, \{\mathsf{Agent}, \mathsf{Product}\}, \{\mathsf{Company}, \mathsf{Product}\}]\}$, 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 $\Xi$, mais ne satisfait aucune DF singulière. En conséquence, aucun ensemble de DFs ne peut être équivalent à $\Xi$, 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 $\{\mathsf{Agent}, \mathsf{Company}\}$, $\{\mathsf{Agent}, \mathsf{Product}\}$ et $\{\mathsf{Company}, \mathsf{Product}\}$ : la DJ de $\Xi$ exprime que cette décomposition est sans perte !

$R_1$

| Agent | Company |
|---|---|
| Smith | Ford |
| Smith | GM |
| Jones | Ford |
| Brown | Ford |
| Brown | GM |
| Brown | Toyota |

$R_2$

| Agent | Product |
|---|---|
| Smith | car |
| Smith | truck |
| Jones | car |
| Jones | truck |
| Brown | car |
| Brown | bus |

$R_3$

| Company | Product |
|---|---|
| Ford | car |
| Ford | truck |
| GM | car |
| GM | truck |
| Toyota | car |
| Toyota | bus |

Observer :

— Jones vend des voitures et GM fabrique des voitures, mais Jones ne travaille pas pour GM. Donc, $R \not\models\!\!\bowtie [\{\mathsf{Product}, \mathsf{Agent}\}, \{\mathsf{Product}, \mathsf{Company}\}]$.

— Brown travaille pour Ford et Ford fabrique des camions, mais Brown ne vend pas de camions. Donc, $R \not\models\!\!\bowtie [\{\mathsf{Company}, \mathsf{Agent}\}, \{\mathsf{Company}, \mathsf{Product}\}]$.

— Brown travaille pour Ford et Brown vend des autobus, mais Ford ne fabrique pas d'autobus. Donc, $R \not\models\!\!\bowtie [\{\mathsf{Agent}, \mathsf{Company}\}, \{\mathsf{Agent}, \mathsf{Product}\}]$.

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

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

avec trois composants, ne peut pas être exprimée comme une ou plusieurs DMVs. □

# Chapitre 8

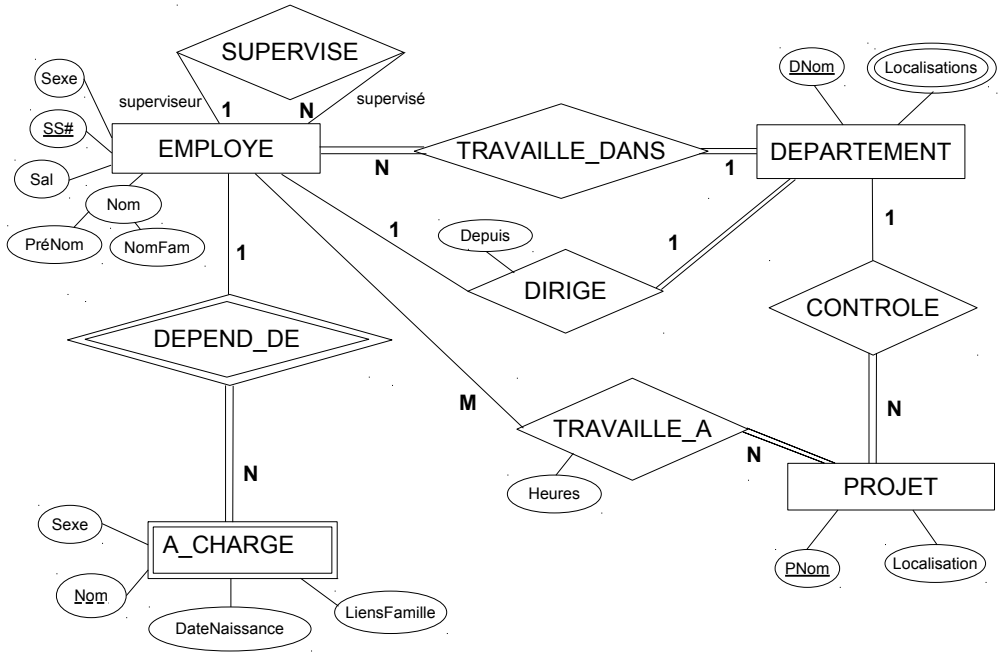
# Le Modèle Entité-Association

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, SupervisePar)
   PK(SS#)
   FK(SupervisePar) 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, ContrôlePar)
   PK(PNom)
   FK(ContrôlePar) 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 $A, B, C, \ldots$ En pratique, un objet peut prendre différentes formes, telle 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 :

$$R_1(A)R_1(B)_{R_2(A)}W_1(A)W_1(B)_{R_2(B)}W_2(C)\cdot \tag{9.1}$$

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 évènements (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 ne soit "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 $\sigma_1$ et $\sigma_2$ ont les mêmes effets sur toute base de données.

Premièrement, deux lectures juxtaposées peuvent être permutées :

si $i \neq j$, 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 :

si $i \neq j$ et $X \neq Y$, 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, 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 est *sérialisable* s'il existe une exécution sérielle $\sigma'$ telle que $\sigma \sim \sigma'$.

Le *graphe de précédence* d'une exécution $\sigma$ est un graphe orienté $G$ dont les sommets sont les transactions de $\sigma$. Le graphe de précédence 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 10.** *Une exécution est sérialisable si et seulement si son graphe de précédence est acyclique.*

Par ailleurs, si le graphe de précédence $G$ d'une exécution $\sigma$ 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 :

$$R_3(A)W_4(A)W_3(A)W_5(A). \tag{9.2}$$

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 24.** 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 précédée 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. □

**Lemme 1.** *Soit $\sigma$ une exécution qui est conforme au protocole 2PL. Si le graphe de précédence de $\sigma$ 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 $\sigma$ contient une arête orientée de $T_i$ à $T_j$. Alors, l'exécution a une des trois formes suivantes :

$$\alpha R_i(A)\beta W_j(A)\gamma$$
$$\alpha W_i(A)\beta R_j(A)\gamma$$
$$\alpha W_i(A)\beta W_j(A)\gamma$$

Dans les trois cas, un relâchement $U_i(A)$ doit être ajouté aux actions de $\beta$, 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 $T_j$. □

**Proposition 11.** *Si une exécution $\sigma$ est conforme au protocole 2PL, alors $\sigma$ est sérialisable.*

*Démonstration.* Preuve par contraposition. Soit $\sigma$ une exécution qui n'est pas sérialisable. Sur base de la Proposition 10, le graphe de précédence de $\sigma$ contient un cycle $\langle T_{i_1}, T_{i_2}, T_{i_3}, \ldots, T_{i_\ell}, T_{i_1}\rangle$.

On démontrera que $\sigma$ n'est pas conforme au protocole 2PL.

Nous procédons par contradiction : supposons que $\sigma$ 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 $T_{i_3}\ldots$ En conséquence, le lock point de $T_{i_1}$ précède le lock point de $T_{i_1}$, une contradiction. □

L'exécution de l'exemple 24 est sérialisable mais n'est pas conforme au protocole 2PL. L'inverse de la Proposition 11 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*).

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 refusée 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 25.** 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 $S_1(A)S_2(A)X_3(A)X_1(A)$. □

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 11 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 $T_{i+1}$, $T_{i+2}$,... 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 annulées doivent être "défaites". Il s'agit du principe d'*atomicité* : soit toutes les modifications d'une transaction sont effectuées, soit aucune ne l'est.

Considérer, 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 durabilité). Par contre, si le système se plante au milieu du processus de virement, vous vous attendez à ce que 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 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 $R_3(A)$.

$$X_1(A)X_1(B)R_1(A)R_1(B)W_1(A)U_1(A)_{X_3(A)R_3(A)W_3(A)U_3(A)C_3}\lightning W_1(B)U_1(B)C_1$$

Supposons, de nouveau, que la transaction $T_1$ est annulée après $U_1(A)$ mais avant $W_1(B)$, à l'endroit du symbole $\lightning$. 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 :

$$X_1(A)X_1(B)R_1(A)R_1(B)W_1(A)U_1(A)_{S_4(A)R_4(A)X_4(B)W_4(B)}\lightning W_1(B)U_1(B)C_1$$

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

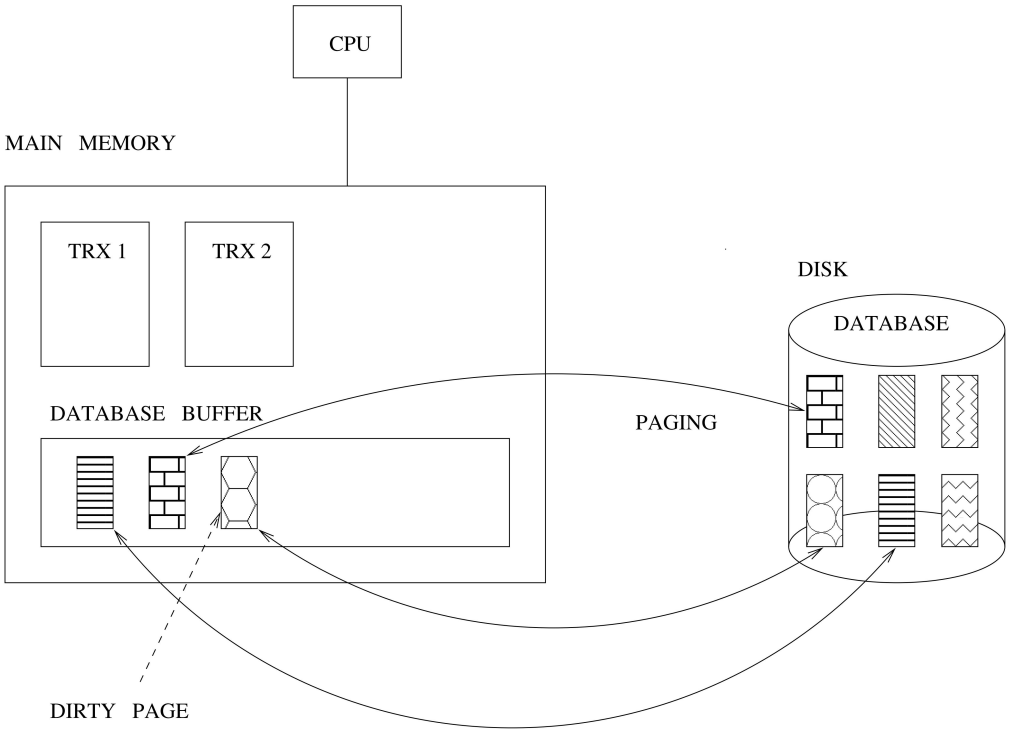



FIGURE 10.1 – Le buffer.

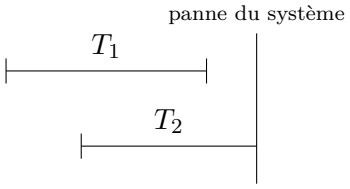



FIGURE 10.2 – Transaction $T_1$ est à refaire, $T_2$ à défaire.

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.
- `(`$T$`, `$A$`, image_avant, 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 $A$.

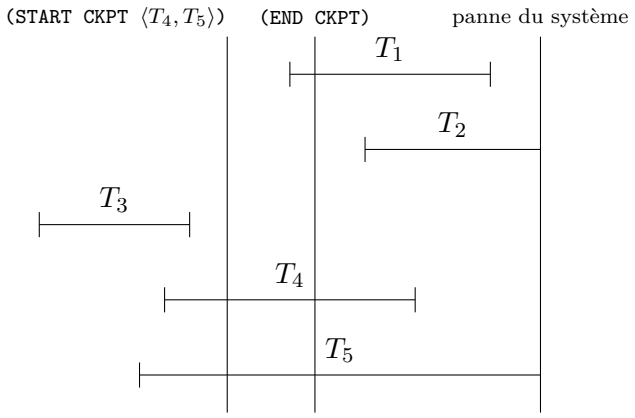



FIGURE 10.3 – Transactions $T_1$ et $T_4$ sont à refaire, $T_2$ et $T_5$ à défaire.

| $T_1$ | $T_2$ | $A = 3, B = 7$ | Enregistrements dans le journal |
|---|---|---|---|
| **begin** | | | $(T_1, \texttt{begin})$ |
| **read** $A \to x$ | | $x = 3$ | |
| $x \leftarrow x + 1$ | | $x = 4$ | |
| **write** $x \to A$ | | $A = 4$ | $(T_1, A, 3, 4)$ |
| **commit** | | | $(T_1, \texttt{commit})$ |
| | **begin** | | $(T_2, \texttt{begin})$ |
| | **read** $B \to y$ | $y = 7$ | |
| | $y \leftarrow y - 1$ | $y = 6$ | |
| | **write** $y \to B$ | $B = 6$ | $(T_2, B, 7, 6)$ |
| | **read** $A \to z$ | $z = 4$ | |
| | $z \leftarrow z + 1$ | $z = 5$ | |

↯ `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` $\langle T_1, \ldots, T_k \rangle$`)`, où $T_1, \ldots, T_k$ 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 ($\geq 1$). 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 | Gagneur | 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 |
|---|---|---|---|
| | E1 | Info | 40 |
| | E1 | Bio | 60 |
| | E2 | Eco | 100 |
| | E3 | Bio | 50 |
| | E3 | Eco | 50 |
| | E4 | Eco | 100 |
| | E5 | Eco | 50 |
| | E5 | Bio | 25 |
| | E5 | Info | 25 |

| DÉPARTEMENT | NomDept | Budget | Chef |
|---|---|---|---|
| | Info | 5000 | E1 |
| | Bio | 3500 | E1 |
| | Eco | 4000 | E5 |

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 |
|---|---|---|---|
| | Ed | Jeux | 8 jan 1982 |
| | Ed | MIS | 11 jan 1997 |
| | An | Jeux | 13 sep 2001 |
| | Pierre | MIS | 23 oct 1992 |
| | Pierre | Jeux | 11 nov 1995 |

| DÉPARTEMENTS | Dept | Chef |
|---|---|---|
| | Jeux | An |
| | MIS | Pierre |

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 |
|---|---|---|---|---|---|
| | 2004 | Papillon | 200 | M | 17/8 |
| | 2004 | Libre | 200 | M | 16/8 |
| | 2000 | Libre | 50 | F | 23/9 |
| | 2000 | Papillon | 200 | M | 23/9 |
| | ⋮ | | | | |

| VILLES | Année | Ville |
|---|---|---|
| | 2004 | Athènes |
| | 2000 | Sydney |
| | ⋮ | |

| NAT | Année | Athlète | Pays |
|---|---|---|---|
| | 2004 | M. Phelps | USA |
| | 2004 | T. Yamamoto | JAP |
| | 2004 | S. Parry | GBR |
| | 2000 | I. de Bruijn | NED |
| | 2000 | M. Phelps | GBR |
| | ⋮ | | |

| ATHLÈTES | Athlète | Sexe |
|---|---|---|
| | M. Phelps | M |
| | T. Yamamoto | M |
| | S. Parry | M |
| | I. de Bruijn | F |
| | ⋮ | |

| 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 |
|---|---|
| Oncle Urbain | Duvel |
| Oncle Urbain | Leffe |
| Oncle Urbain | Orval |
| Caroline | Leffe |
| Caroline | Carlsberg |
| Jim | Grimbergen |
| An | Leffe |
| Pierre | Leffe |
| Eric | Chimay |

MARIAGES

| Homme | Femme |
|---|---|
| Oncle Urbain | Tante Odette |
| Jim | An |
| Pierre | Caroline |

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 $\{(\mathsf{Country}, x), (\mathsf{Capital}, y)\}$ signifie que $y$ est la capitale de $x$ :

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

La requête demandée :

$$\pi_{\mathsf{Population}}((\rho_{\mathsf{Name}\mapsto\mathsf{Capital}}(\mathsf{Cities})) \bowtie (\sigma_{\mathsf{Country}=\text{“Mali”}}(\mathsf{CC}))).$$

□

**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 $A, B \in \mathit{sorte}(E)$. Définissons la requête CURRENCIES comme suit :

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

Alors la requête demandée est :

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

□

**Question 13.** Base de données de la question 3.

Quel chauffeur a été à toutes les destinations ?

**Réponse algèbre.** Soit

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

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

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

□

**Question 14.** Base de données de la question 3.

Quelles destinations n'ont jamais été visitées par John ?

**Réponse algèbre.**

$$\mathsf{DESTINATIONS} - \rho_{\mathsf{Destination}\mapsto\mathsf{Name}}(\pi_{\mathsf{Destination}}(\sigma_{\mathsf{Driver}=\text{“John”}}(\mathsf{TRIPS}))).$$

□

**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.**

$$\begin{array}{lll}\{x \mid \exists y & ( & \exists u(\exists v(\mathsf{TRIPS}(u, y, x, \text{“AntwerpZoo”}, v))) \\ & & \wedge \\ & & \exists w(\exists z(\mathsf{BUSES}(y, w, \text{“Renault”}, z))) \\ & ) & \\ \} & & \end{array}$$

□

**Question 16.** Base de données de la question 4.

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

**Réponse calcul.**

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

□

**Question 17.** Base de données de la question 4.

Quel entrepôt ne stocke aucun produit rouge ? $\rightsquigarrow$ D1 etD3

**Réponse calcul.** On utilisera $\mathsf{is\_a\_warehouse}(x)$ comme abbréviation pour :

$$\exists v(\exists w(\mathsf{WAREHOUSES}(x, v, w))).$$

La requête demandée :

$$\{x \mid \mathsf{is\_a\_warehouse}(x) \wedge \neg\exists y(\exists q(\mathsf{STOCK}(x, y, \text{“red”}, q)))\} \tag{10.1}$$

Une autre solution est :

$$\{x \mid \mathsf{is\_a\_warehouse}(x) \wedge \forall y(\forall z(\forall q(\neg\mathsf{STOCK}(x, y, z, q) \vee \neg(z = \text{“red”}))))\} \tag{10.2}$$

□

**Question 18.** Base de données de la question 4.

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

**Réponse calcul.** Il suffit de remplacer $\neg(z =$ "red") par $(z =$ "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 {(Pilote, M. Häkkinen),(GP, Belgique),(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 gagneur et le deuxième faisaient partie de la même écurie ? $\rightsquigarrow$ 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. $\rightsquigarrow$ 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 ? $\rightsquigarrow$ E3

**Question 23.** Base de données de la question 6.

Qui sont les chefs de E3 ? $\rightsquigarrow$ 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. $\rightsquigarrow$ An

**Question 25.** Base de données de la question 8.

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

**Question 26.** Base de données de la question 8.

Donnez tout individu qui n'est pas marié. $\rightsquigarrow$ 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.

Donne les noms des hommes mariés dont le mariage n'est pas superbe. $\rightsquigarrow$ 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. $\rightsquigarrow$ 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 $\mathcal{I}$,

$$\begin{aligned}
[\![\sigma_{A \leq B}(E)]\!]^{\mathcal{I}} &:= \{t \in [\![E]\!]^{\mathcal{I}} \mid t(A) \leq t(B)\};\\
[\![\sigma_{A \leq \text{“}a\text{”}}(E)]\!]^{\mathcal{I}} &:= \{t \in [\![E]\!]^{\mathcal{I}} \mid t(A) \leq a\};\\
[\![\sigma_{A < B}(E)]\!]^{\mathcal{I}} &:= \{t \in [\![E]\!]^{\mathcal{I}} \mid t(A) < t(B)\};\\
[\![\sigma_{A < \text{“}a\text{”}}(E)]\!]^{\mathcal{I}} &:= \{t \in [\![E]\!]^{\mathcal{I}} \mid t(A) < a\}.
\end{aligned}$$

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 |
|---|---|---|
| Génie logiciel | Mens | 7 |
| Gestion de projets | Mens | 4 |
| Bases de données | Wijsen | 10 |
| ⋮ | | |

NOTES

| Nom | C | Note |
|---|---|---|
| Ed | Génie logiciel | 13 |
| Ed | Bases de données | 15 |
| Tim | Bases de données | 13 |
| ⋮ | | |

```
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 |
|---|---|---|
| Taco | Anvers | mexicain |
| Naxos | Anvers | grec |
| Trevi | Anvers | italien |
| Campos | Mons | italien |
| Pronto | Mons | italien |

VISITES

| Nom | Restaurant |
|---|---|
| Jean | Naxos |
| Jean | Trevi |
| Jean | Pronto |
| Pierre | Naxos |
| Pierre | Trevi |
| An | Campos |
| An | Pronto |

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

**Question 34.** Base de données de la question 33.

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

**Question 35.** Base de données de la question 33.

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

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

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

**Réponse algèbre.** Soit

$$J := \pi_{\mathsf{Cru},\mathsf{Millésime}}(\sigma_{\mathsf{Qualité}=\text{“A”}}(\mathsf{VINS}) \bowtie \rho_{\mathsf{Mill}\mapsto\mathsf{Millésime}}(\sigma_{\mathsf{Buveur}=\text{“Jean”}}(\mathsf{ABUS}))),$$

les vins de qualité A bus par Jean.
Soit

$$P := \pi_{\mathsf{Cru,Millésime}}(\mathsf{VINS} \bowtie \rho_{\mathsf{Mill}\mapsto\mathsf{Millésime}}(\sigma_{\mathsf{Buveur}=\text{“Pierre”}}(\mathsf{ABUS}))),$$

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

$R$

| $A$ | $B$ |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |
| 1 | 3 |

, le résultat est

| $A$ | $B$ |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |

.

**Réponse algèbre.** $R \bowtie \rho_{AB\mapsto BA}(R)$. □

**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 ($\geq 1$).

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 |
|---|---|---|
| | Anne | pomme |
| | Anne | steak |
| | Anne | tomate |
| | Bill | pomme |
| | Bill | tomate |
| | Ed | pomme |
| | Ed | orange |
| | Ed | salade |
| | Ed | poulet |

| NOURRITURE | Aliment | Catégorie |
|---|---|---|
| | pomme | fruit |
| | steak | viande |
| | tomate | légume |
| | orange | fruit |
| | salade | légume |
| | poulet | viande |
| | poire | fruit |

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

**Question 40.** Base de données de la question 39.

Qui mange au moins deux fruits différents ? $\rightsquigarrow$ 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 $\{(\texttt{F\#}, f), (\texttt{P\#}, p)\}$ 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,

$R$

| $A$ |
|---|
| 7 |
| 8 |
| 10 |
| 17 |
| 23 |

$\rightsquigarrow$ $q(R)$

| $A$ | $B$ |
|---|---|
| 7 | 8 |
| 8 | 10 |
| 10 | 17 |
| 17 | 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. $\rightsquigarrow$

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 orienté (ou arc) du sommet $a$ vers le sommet $b$. Traduisez la requête $\pi_{AB}(\rho_{B\mapsto C}(R) \bowtie \rho_{A\mapsto C}(R)) - R$ 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. $\pi_X(R \cup S) \sqsubseteq \pi_X(R) \cup \pi_X(S)$
2. $\pi_X(R) \cup \pi_X(S) \sqsubseteq \pi_X(R \cup S)$
3. $\pi_X(R \bowtie S) \sqsubseteq \pi_X(R) \bowtie \pi_X(S)$
4. $\pi_X(R) \bowtie \pi_X(S) \sqsubseteq \pi_X(R \bowtie S)$

**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 $[\![\pi_X(R \bowtie S)]\!]^{\mathcal{I}} = \emptyset$ et $[\![R]\!]^{\mathcal{I}} \neq \emptyset$, il est correct de conclure $R \not\equiv \pi_X(R \bowtie S)$. □

**Question 50.** Soit $R$ un nom de relation. Soient $X, Y$ des ensembles tels que $sorte(R) = X \cup Y$. Prouvez ou réfutez :

1. $\pi_X(R) \bowtie \pi_Y(R) \sqsubseteq R$
2. $R \sqsubseteq \pi_X(R) \bowtie \pi_Y(R)$

**Question 51.** Pour la base de données

| $R$ | $A$ | $B$ |
|---|---|---|
| | 1 | 1 |
| | 1 | 2 |
| | 2 | 1 |
| | 1 | 3 |

, donnez le résultat de :

$$\rho_{C\mapsto B}(\pi_{AC}(\rho_{A\mapsto C}(R) \bowtie R)).$$

**Réponse.**

| $A$ | $B$ |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |
| 2 | 2 |

□

**Question 52.** Soient $R$ et $S$ des noms de relation avec $sorte(R) = \{A, B\}$ et $sorte(S) = \{B, C\}$ Prouvez :

$$\pi_A(R \bowtie S) \sqsubseteq \pi_A(\sigma_{B=C}(R \bowtie (\rho_{B \mapsto C}(\pi_B(S))))).$$

**Réponse.** Soit $\mathcal{I}$ une base de données dont le schéma contient $R$ et $S$. Soit $\{(A, a)\}$ n'importe quel tuple de $[\![\pi_A(R \bowtie S)]\!]^{\mathcal{I}}$.
Donc, il existe $b, c$ tels que $\{(A, a), (B, b), (C, c)\} \in [\![R \bowtie S]\!]^{\mathcal{I}}$.
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 $\{(B, b)\} \in [\![\pi_B(S)]\!]^{\mathcal{I}}$.
Donc, il existe $b$ tel que $\{(A, a), (B, b)\} \in R^{\mathcal{I}}$ et $\{(C, b)\} \in [\![\rho_{B \mapsto C}(\pi_B(S))]\!]^{\mathcal{I}}$.
Donc, il existe $b$ tel que $\{(A, a), (B, b), (C, b)\} \in [\![R \bowtie \rho_{B \mapsto C}(\pi_B(S))]\!]^{\mathcal{I}}$.
Donc, il existe $b$ tel que $\{(A, a), (B, b), (C, b)\} \in [\![\sigma_{B=C}(R \bowtie \rho_{B \mapsto C}(\pi_B(S)))]\!]^{\mathcal{I}}$.
Donc, $\{(A, a)\} \in [\![\pi_A(\sigma_{B=C}(R \bowtie (\rho_{B \mapsto C}(\pi_B(S)))))]\!]^{\mathcal{I}}$. □

**Question 53.** Soient

$$\begin{aligned} E_1 &:= \pi_{\mathsf{Cru,Qualité}}(\mathsf{VINS}) \bowtie \rho_{\mathsf{Qualité} \mapsto \mathsf{Qual}}(\pi_{\mathsf{Cru,Qualité}}(\mathsf{VINS})) \\ E_2 &:= \pi_{\mathsf{Cru}}(\mathsf{VINS}) - \pi_{\mathsf{Cru}}(E_1 - \sigma_{\mathsf{Qualité}=\mathsf{Qual}}(E_1)) \end{aligned}$$

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 $sorte(R) = sorte(S)$. 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 $\mathcal{I}$, $|[\![E_1]\!]^{\mathcal{I}}| \leq |[\![E_2]\!]^{\mathcal{I}}|$.

Il est facile de prouver que pour toutes expressions $E$, $E_1$, $E_2$,

$$|\sigma_{A=B}(E)| \leq |E| \tag{10.3}$$
$$|\sigma_{A=\text{“}a\text{”}}(E)| \leq |E| \tag{10.4}$$
$$|\pi_X(E)| \leq |E| \tag{10.5}$$
$$|E_1 \bowtie E_2| \leq |E_1| \times |E_2| \tag{10.6}$$
$$|\rho_{A \mapsto B}(E)| \leq |E| \tag{10.7}$$
$$|E_1 - E_2| \leq |E_1| \tag{10.8}$$

Soient $R$ et $S$ deux noms de relation tels que $sorte(R) = sorte(S) = \{A\}$. Soit $\mathcal{I}$ la base de données suivante :

$R$

| $A$ |
|---|
| 0 |

$S$

| $A$ |
|---|
| 1 |

Évidemment, $[\![R \cup S]\!]^{\mathcal{I}} = \{\{(A,0)\},\{(A,1)\}\}$, un ensemble avec deux tuples. Soit $E$ une expression quelconque en algèbre SPJRD. On démontre que $|[\![E]\!]^{\mathcal{I}}| \leq 1$.

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 $|[\![E]\!]^{\mathcal{I}}| = 1$. Ensuite, *l'étape* d'induction :

**CAS** $E \equiv \sigma_{A=B}(E_1)$**.** Par hypothèse d'induction, $|[\![E_1]\!]^{\mathcal{I}}| \leq 1$. Par (10.3), $|[\![E]\!]^{\mathcal{I}}| \leq 1$.

**CAS** $E \equiv \sigma_{A=\text{“}b\text{”}}(E_1)$**.** Analogue.

**CAS** $E \equiv \pi_X(E_1)$**.** Analogue.

**CAS** $E \equiv \rho_{A\mapsto B}(E_1)$**.** Analogue.

**CAS** $E \equiv E_1 \bowtie E_2$**.** Par hypothèse d'induction, $|[\![E_1]\!]^{\mathcal{I}}| \leq 1$ et $|[\![E_2]\!]^{\mathcal{I}}| \leq 1$. Par (10.6), $|[\![E]\!]^{\mathcal{I}}| \leq 1$.

**CAS** $E \equiv E_1 - E_2$**.** Par hypothèse d'induction, $|[\![E_1]\!]^{\mathcal{I}}| \leq 1$. Par (10.8), $|[\![E]\!]^{\mathcal{I}}| \leq 1$.

Donc $|[\![E]\!]^{\mathcal{I}}| \leq 1$. Par conséquent, $E \not\equiv R \cup S$. □

**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 $sorte(R) = \{A\}$ et $sorte(S) = \{B\}$, 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 : $R \bowtie \rho_{B\mapsto A}(S)$.

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 $\{x \mid \exists y(R(x, \text{“}a\text{”}, y) \wedge \neg\exists z(R(x, \text{“}b\text{”}, z)))\}$ en algèbre relationnelle ?

**Réponse.** En supposant $sorte(R) = ABC$ :

$$\pi_A(\sigma_{B=\text{“}a\text{”}}R) - \pi_A(\sigma_{B=\text{“}b\text{”}}R).$$

□

# Exercices sur le Chapitre 4

**Question 57.** Soit $R$ un nom de relation avec $sorte(R) = \{A, B\}$. 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.**

$$\{x \mid \exists y(\exists z(\mathsf{VINS}(x,y,z) \wedge \neg(\exists v(\exists w(\mathsf{VINS}(x,v,w) \wedge \neg(z=w))))))\}$$

Si on prend la liberté d'alléger la syntaxe :

$$\{x \mid \exists y,z(\mathsf{VINS}(x,y,z) \wedge \neg\exists v,w(\mathsf{VINS}(x,v,w) \wedge z \neq w))\}$$

En mots simples : donnez tout $x$ tel qu'il existe un tuple $\langle x,y,z\rangle$ en $\mathsf{VINS}$ 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 $w \neq z$). □

**Question 59.** Soient $R$ et $S$ deux noms de relation d'arité 1. Quelle 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. □

**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 $\{x \mid R(x, \text{“}a\text{”}) \wedge \exists y(R(\text{“}b\text{”}, y))\}$ 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 ? $\rightsquigarrow$ 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 ? $\rightsquigarrow$ 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 ? $\rightsquigarrow$ 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 ? $\rightsquigarrow$ 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 ? $\rightsquigarrow$ 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, 10h15, 13h15 ou 15h15, 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. Wijsen ; 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. Wijsen | 2 | Jeudi | 13h15 | 3E11/P | S/2I/17 |
| S/3I/2 | Systèmes d'information | J. Wijsen | 2 | Jeudi | 13h15 | 3E11/P | S/3I/5 |
| S/2I/17 | Fichiers et Bases de Données | J. Wijsen | 1 | Vendredi | 10h15 | 3E11/P | S/1I/3 |
| S/3I/5 | Structure de l'information | V. Bruyère | 1 | Vendredi | 10h15 | 3E10/P | S/1I/3 |
| S/1I/3 | Informatique I | P. Dufour | 1 | Vendredi | 10h15 | 211/VI | — |
| S/1I/3 | Informatique I | P. Dufour | 1 | Mardi | 15h15 | 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 régisseurs 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 $\Sigma$ 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.**

$$\begin{aligned}\Sigma \ = \{ \ & TD \to SPM, \\ & DP \to T, \\ & AP \to TD \ \}\end{aligned}$$

Les clés sont $TDA$ et $AP$. Cette relation n'est pas en 3NF parce que $\Sigma \models TD \to S$, $\Sigma \not\models TD \to A$ 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 $\Sigma \models\!\bowtie [TDSPM, TDA]$.

| Composant | Schéma | DF | Clé(s) | BCNF ? |
|---|---|---|---|---|
| 1 | $TDSPM$ | $\{TD \to SPM, DP \to T\}$ | $TD, DP$ | oui |
| 2 | $TDA$ | $\{\}$ | $TDA$ | oui |

Il est clair que cette décomposition ne préserve pas les DFs $AP \to T$ et $AP \to D$.

Essayons ensuite d'ajouter au deuxième composant l'attribut $P$ afin de préserver $AP \to TD$ :

| Composant | Schéma | DF | Clé(s) | BCNF ? | 3NF ? |
|---|---|---|---|---|---|
| 1 | $TDSPM$ | $\{TD \to SPM, DP \to T\}$ | $TD, DP$ | oui | oui |
| 2 | $TDAP$ | $\{AP \to TD, TD \to P, DP \to T\}$ | $AP, TDA$ | non | oui |

Notez que $\Sigma \models TDP \to SM$, et donc $\Sigma \models\!\bowtie [TDSPM, TDAP]$. 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 | $TDSM$ | $\{TD \to SM\}$ | $TD$ | oui | oui |
| 2 | $TDAP$ | $\{AP \to TD, TD \to P, DP \to T\}$ | $AP, TDA$ | 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 → Rue

— Rue, Ville → CinémaNom

— Téléphone → CinémaNom, Ville

— CinémaNom, Ville, Heure → 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 $\langle a_1, a_2, a_3, \ldots, a_8 \rangle$ 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

$$\text{CinémaNom}, \text{Rue}, \text{Ville} \twoheadrightarrow \text{Téléphone},$$

qui est équivalente à

$$\text{CinémaNom}, \text{Rue}, \text{Ville} \twoheadrightarrow \text{Titre}, \text{Régisseur}, \text{Durée}, \text{Heure}.$$

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*. □

**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 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 a changé 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 des 200 mètres papillon messieurs a eu lieu le 17/8. Dans une course, le temps du gagneur (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 qu'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 |
| | | | | | ⋮ | | | | |

- Quelles sont les DFs pour ce schéma-de-relation ? Écrivez les DFs en format $X \to A$ 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 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'enregistra 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 en 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 remporté 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 !**<br>Ch. Michaux<br>*math* | **P=NP ?**<br>R. Astier<br>*info* | **Les abeilles**<br>P. Falmagne<br>*bio* |
| 10h30 | **L'élément H**<br>M. Hecq<br>*chimie* | **La résonance**<br>P. Gillis<br>*physique* | **Fermat**<br>R. Astier<br>*math* |
| 13h00 | **La vie sur Mars**<br>M. Wautelet<br>*physique* | **Devenir 100 ans**<br>P. Rasmont<br>*bio* | **Le recyclage**<br>Ph. Dubois<br>*chimie* |
| 14h30 | **Les bourdons**<br>P. Rasmont<br>*bio* | **Vivre en 2100**<br>M. Wautelet<br>*physique* | **La récursivité**<br>V. Bruyère<br>*info* |

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 $X \to A$ 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 benzine ;

— 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 $X \to A$ 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 Taton" 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 $X \to A$ 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 94.** Une DF $X \to Y$ sur un ensemble $\mathcal{A}$ d'attributs est *triviale* si $X \to Y$ est satisfaite par toute relation sur $\mathcal{A}$. Prouvez : $X \to Y$ est triviale si et seulement si $Y \subseteq X$.

**Réponse.** L'implication $\Longleftarrow$ est triviale. On démontrera $\Longrightarrow$ par contraposition. Soit $X \to Y$ 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 $X \to Y$. Donc, si $X \to Y$ n'est pas triviale, on peut toujours construire une relation $R$ qui ne satisfait pas $X \to Y$. □

**Question 95.** Développez un algorithme pour décider si deux ensembles $\Sigma_1$ et $\Sigma_2$ de DFs sont équivalents.

**Question 96.** Soit $\mathcal{A}$ un ensemble d'attributs et $X, Y \subseteq \mathcal{A}$. Prouvez $\{X \to Y\} \models\!\!\bowtie [XY, X(\mathcal{A} \setminus Y)]$.

**Question 97.** 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 98.** Soit $\mathcal{A} = ABCD$ un ensemble d'attributs. Donnez un ensemble $\Sigma$ 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 $\{A \to \mathcal{A}, B \to \mathcal{A}\}$ 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 :

$$\{AB \to \mathcal{A}, AC \to \mathcal{A}, AD \to \mathcal{A}, BC \to \mathcal{A}, BD \to \mathcal{A}, CD \to \mathcal{A}\}$$

     résulte en six clés :

$$AB, AC, AD, BC, BD, CD.$$

□

**Question 99.** Soit $(\mathcal{A}, \Sigma)$ un schéma-DF avec $\mathcal{A} = ABCDEF$ et

$$\Sigma = \{AB \to CD, BC \to DE, CD \to EF, DE \to FA, EF \to B\}.$$

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 $AB \to CD$. $AB$ détermine donc $ABCD$. Dès lors, $AB$ détermine $DE$ et $EF$ à cause de $BC \to DE$ et $CD \to EF$. $AB$ 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 $\Sigma \models EF \to B$ et $\Sigma \not\models EF \to A$. Ce schéma-DF est en 3NF puisque tout attribut fait partie d'une clé. □

**Question 100.** Trouver un ensemble irréductible équivalent à $\{A \to BC,\ B \to C,\ A \to B,\ AB \to C,\ AC \to D\}$.

**Question 101.** Considérez le schéma-DF $(\mathcal{A}, \Sigma)$ où

$$\mathcal{A} = \{\mathsf{W\sharp}, \mathsf{Address}, \mathsf{City}, \mathsf{Product}, \mathsf{Color}, \mathsf{Qty}, \mathsf{Weight}\}$$

et

$$\begin{aligned} \Sigma \quad = \{ \quad & \mathsf{W\sharp} \to \mathsf{Address}, \mathsf{City}; \\ & \mathsf{Address}, \mathsf{City} \to \mathsf{W\sharp}; \\ & \mathsf{W\sharp}, \mathsf{Product}, \mathsf{Color} \to \mathsf{Qty}; \\ & \mathsf{Product} \to \mathsf{Weight} \ \}. \end{aligned}$$

— Quelles sont les clés de ce schéma-DF ?

— Expliquez pourquoi ce schéma-DF n'est pas en 3NF.

**Question 102.** Pour le schéma-DF proposé en question 101, donnez une décomposition en 3NF qui préserve les DFs. Est-ce que cette décomposition est en BCNF ?

**Question 103.** Soit $(\mathcal{A}, \Sigma)$ un schéma-DF avec $\mathcal{A} = \{A, B, C\}$ et $\Sigma = \{A \to BC, B \to C\}$. Soit $\{(\mathcal{A}_1, \Sigma_1), (\mathcal{A}_2, \Sigma_2)\}$ une décomposition de ce schéma-DF avec :

— $\mathcal{A}_1 = \{A, B\}$,

— $\Sigma_1 = \{A \to B\}$,

— $\mathcal{A}_2 = \{B, C\}$,

— $\Sigma_2 = \{B \to C\}$.

Est-ce que cette décomposition préserve toutes les DFs du schéma-DF original ? Expliquez.

**Question 104.** Soit donné le schéma-DF $(ABCDEF, \Sigma)$ avec $\Sigma = \{ABC \to E,\ BCD \to F,\ EF \to ABC,\ CEF \to D\}$. 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 $ABC \to E$ et $\Sigma \not\models ABC \to F$. □

**Question 105.** Soit donné le schéma-DF $(ABCDE, \{AB \to CD, C \to A, D \to CB\})$.

— 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 :

$$(ABCD, \{AB \to C, B \to D\})$$
$$(ABCDEF, \{ABC \to D, D \to E, E \to AD\})$$
$$(ABCDE, \{AB \to C, C \to BDE, D \to C\})$$
$$(ABCD, \{A \to B, B \to D, C \to AD, D \to C\})$$
$$(ABCDE, \{AB \to C, CD \to E, E \to AC\})$$
$$(ABCDE, \{AB \to CD, C \to A, D \to B, C \to E\})$$

**Question 106.** Voici trois schémas-DF :

$$(\mathcal{A}_1, \Sigma_1) = (\ ABCDE,\quad \{A \to B, AB \to C, D \to AC, D \to E\}\ )$$
$$(\mathcal{A}_2, \Sigma_2) = (\ ABCDE,\quad \{A \to BC, D \to AE\}\ )$$
$$(\mathcal{A}_3, \Sigma_3) = (\ ABCDEF,\ \{AB \to C, C \to A, BC \to D, ACD \to B$$
$$BE \to C, CE \to FA, CF \to BD, D \to EF\}\ )$$

1. Est-ce que $\Sigma_1$ et $\Sigma_2$ 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 107.** Pour le schéma-DF $(\mathcal{A}, \Sigma)$ où $\mathcal{A} = \{A, B, C, D\}$ et $\Sigma = \{A \to B,\ B \to A,\ C \to D,\ D \to C\}$, 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 108.** Pour le schéma-DF $(\mathcal{A}, \Sigma)$ où $\mathcal{A} = \{A, B, C, D, E\}$ et $\Sigma = \{A \to C,\ B \to D,\ CD \to E\}$, 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 109.** Soit $\mathcal{R} = \{R\}$ un schéma-de-base-de-données avec $sorte(R) = \{A, B\}$. Soit $\Sigma = \{A \to B, R[A] \subseteq R[B]\}$. Soit $\sigma = B \to A$.

— Démontrer $\Sigma \models \sigma$.

— Démontrer qu'il existe une relation *infinie* qui satisfait $\Sigma$ mais qui ne satisfait pas $\sigma$.

# Exercices sur le Chapitre 8

Développer des schémas Entité-Association (Entity-Relationship) pour les énoncés ci-après. Traduire les schémas en modèle relationnel.

**Question 110.** 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 enregistra alors la raison de son absence (par exemple, fièvre, examen à l'université...).

La base de données doit pouvoir répondre, parmi d'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 111.** 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 112.** 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.

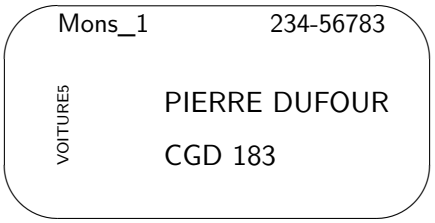
Les centres de service sont bien répandus en Belgique. Chaque centre est identifié par le nom de la ville suivi d'un numéro : Bruxelles_1, Bruxelles_2, Mons_1, Ninove_1, ... Pour chaque centre, il faut stocker l'adresse et un ou plusieurs numéros de téléphone. Chaque centre embauche un exploitant en 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 113.** 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 émail. 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 émail 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, émail, 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 114.** 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'envoie 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 115.** 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 chronométreur 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 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 116.** 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 connait 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 117.** 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 118.** 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 119.** 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 refusée, 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 120.** 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 121.** 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 122.** Comme la question 121, en remplaçant 2PL par Strict 2PL.

**Question 123.** 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 tâches [1] 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 $X_3(A)C_3$. □

**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 refusée 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 124.** Démontrez qu'il existe une exécution sérialisable qui n'est pas conforme au protocole 2PL.

**Question 125.** 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 126.** 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 127.** Voici le contenu d'un journal de type Undo/Redo après une panne de système :

[START $T_1$ ]
[START $T_2$ ]
[START $T_3$ ]
[START $T_4$ ]
[$T_1, A, 0, 1$ ]
[$T_2, B, 0, 1$ ]
[$T_3, C, 0, 1$ ]
[$T_4, D, 0, 1$ ]
[COMMIT $T_1$ ]
[START CKPT $\langle T_2, T_3, T_4 \rangle$]
[START $T_5$ ]
[START $T_6$ ]
[START $T_7$ ]
[$T_5, E, 0, 1$ ]
[$T_6, F, 0, 1$ ]
[$T_7, G, 0, 1$ ]
[COMMIT $T_2$ ]
[COMMIT $T_5$ ]
[END CKPT ]
[COMMIT $T_3$ ]
[COMMIT $T_6$ ]

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 128.** Après une panne du système, le contenu du log est comme suit :

$(T_1, \textbf{begin})$
$(T_1, A, 4, 5)$
$(T_2, \textbf{begin})$
$(T_1, \textbf{commit})$
$(T_2, B, 9, 10)$
$(T_2, A, 5, 6)$
$(T_3, \textbf{begin})$
$(T_3, C, 19, 20)$
$(T_3, \textbf{commit})$

Quelles sont les valeurs pour A, B et C à restaurer dans la base de données pendant la reprise ?

**Question 129.** Comment la procédure de reprise après panne doit-elle être adaptée afin de tenir compte des enregistrements `START CKPT` et `END CKPT` ?

**Question 130.** 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 131.** Comment la procédure de *checkpointing* doit-elle être adaptée au protocole Undo/No-Redo ? Et au Redo/No-Undo ?

**Question 132.** 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 131).

**Question 133.** 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.

# 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 :

**Classe**
    **Ordre**
        **Espèce**

Mammifères
    Carnivores
        Lion
        Loup
        Tigre
        Hyène
    Artiodactyles
        Girafe
        Hippopotame
Reptiles
    Serpents
        Cobra

FIGURE A.1 – Classification hiérarchique des animaux.

*Où peut-on trouver des lions ?*

Par contre, pour répondre à la question :

*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 mettra 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",

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.

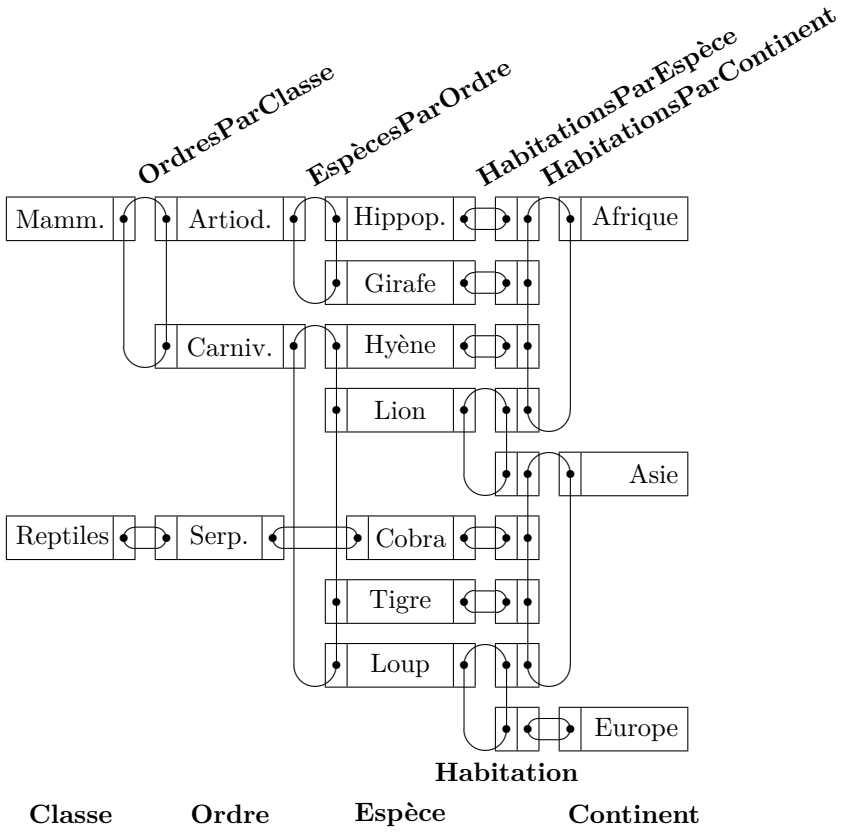



FIGURE A.3 – Classification et répartition des animaux dans une BD de type réseau.

```
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.

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 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.

**EO**

| Espèce | Ordre |
|---|---|
| Lion | Carnivores |
| Loup | Carnivores |
| Tigre | Carnivores |
| Hyène | Carnivores |
| Girafe | Artiodactyles |
| Hippopotame | Artiodactyles |
| Cobra | Serpents |

**OC**

| Ordre | Classe |
|---|---|
| Carnivores | Mammifères |
| Artiodactyles | Mammifères |
| Serpents | Reptiles |

**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.

## 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 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**

**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**.

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')
`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 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

```
<H1> &Eacute;nigmes </H1> <OL>
<LI> Comment faire entrer
quatre &eacute;l&eacute;phants
dans une fiat panda?
</LI> <LI> Comment un
&eacute;l&eacute;phant se
mouche-t-il? </LI> </OL>
```

**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).

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 `<OL>` et `</OL>` (*Ordered List*) délimitent le début et la fin d'une liste ordonnée ; chaque article de la liste se trouve entre les balises `<LI>` et `</LI>` (*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-navigationnelle, 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...

**Bases de Données I**

15 janvier 2007 à 8h15, local 020/IX

Jef Wijsen

NOM + PRENOM :

Orientation + Année :

Examen à cahier ouvert. Durée: 3 heures. Répondez dans les cadres prévus. L'agrafe ne peut pas être enlevée. Cet examen contient 7 pages.

**Question 1** Un restaurant de pizzas à emporter enregistre les commandes dans trois tables. Chaque commande est identifiée par une référence unique (attribut *Ref*). Les commandes, faites par téléphone, sont stockées dans la table COMMANDES : la date et le moment (format HH:MM:SS) du début de l'appel sont enregistrés dans les colonnes *FaiteLe* et *Heure*; la colonne *Personne* stocke le nom de la personne faisant la commande. La pizzeria ne dispose que d'un seul téléphone et est donc incapable d'accepter plusieurs appels entrants simultanément. Deux personnes différentes peuvent porter le même nom, mais chaque commande est faite par une seule personne. Deux valeurs sont possibles pour la colonne *État* : une commande est "en cours" jusqu'au moment où les pizzas commandées sont "emportées" par le client.

Les pizzas existent en différentes sortes (Pepperoni, Forestière,...) et trois grandeurs (small, medium, large). La table LIGNES_DE_COMMANDE stocke le nombre de pizzas commandées, par sorte et taille. Pour une même commande, on n'aura jamais deux lignes portant les mêmes sorte et grandeur. Par exemple, la commande R456 comprend deux larges pizzas Pepperoni, une medium pizza Pepperoni et une medium pizza Forestière.

Le prix d'une pizza dépend de sa sorte et sa grandeur. Les prix en Euros sont stockés dans la table PRIX.

LIGNES_DE_COMMANDE

| Ref | Pizza | Grandeur | Quantité |
|---|---|---|---|
| Q123 | Forestière | medium | 1 |
| R456 | Pepperoni | large | 2 |
| R456 | Pepperoni | medium | 1 |
| R456 | Forestière | medium | 1 |
| S789 | Pepperoni | large | 3 |

COMMANDES

| Ref | Personne | FaiteLe | Heure | État |
|---|---|---|---|---|
| Q123 | D. Dufour | 24/12/2006 | 13:25:12 | Emportées |
| R456 | D. Dufour | 5/1/2007 | 19:02:52 | En cours |
| S789 | P. Poulet | 5/1/2007 | 15:21:53 | Emportées |

PRIX

| Pizza | Small | Medium | Large |
|---|---|---|---|
| Forestière | 13 | 14 | 15 |
| Pepperoni | 11 | 13 | 15 |
| Marguerita | 9 | 9.50 | 10 |

Complétez ce schéma avec les contraintes de type:

```
<nom-table> PRIMARY KEY (...)
<nom-table> UNIQUE (...)
<nom-table> FOREIGN KEY (...) REFERENCES ...
```



.../2

**Question 2** Au début du mois de janvier, le Lycée Royal Marguerite Bervoets organise chaque année une soirée pendant laquelle les parents des élèves peuvent rencontrer les professeurs pour discuter des résultats de leur(s) enfant(s). Pour organiser ces rencontres, les parents doivent remplir et renvoyer avant le 1er janvier un formulaire avec les rencontres souhaitées. Voici un tel formulaire rempli :

126

Les parents de ...Pierre Dupont... (prénom et nom), élève de la classe ...4Ma..., souhaitent rencontrer les professeurs suivants pour les cours indiqués (cocher zéro, une ou deux cases par ligne):

| | | Mère | Père |
|---|---|---|---|
| K. Depoortere | mathématiques | x | |
| K. Depoortere | physique | x | |
| K. Depoortere | technologie | | |
| E. Flavius | français | | |
| Y. Leterme | français | | x |
| P. Pivert | français | | |
| ⋮ | | | |
| P. Stanislav | biologie | x | x |

Notez que le formulaire ci-dessus permettra que madame Dupont rencontre le professeur Depoortere pendant que monsieur rencontre le professeur Leterme. Par contre, la dernière ligne indique qu'ils souhaitent être ensembles pour rencontrer le professeur Stanislav.

Bien sûr, les Dupont peuvent avoir plusieurs enfants inscrits à cette école; ils doivent alors remplir un formulaire par enfant. Deux élèves peuvent porter le même (pré)nom. Néanmoins, pour des raisons administratives évidentes, le lycée assure qu'aucune classe ne contienne deux élèves avec le même nom et le même prénom. Donc, s'il y avait un autre Pierre Dupont dans cette école, il serait dans une autre classe. Chaque élève ne fait partie que d'une seule classe. Chaque élève n'a qu'un seul père et une seule mère.

Un professeur peut enseigner plusieurs branches et le lycée a plusieurs professeurs par branche. Néanmoins, dans une même classe, chaque branche est enseignée par un seul professeur (mais un même professeur peut enseigner deux branches différentes dans une même classe, par exemple, mathématiques et physique). Les professeurs du lycée ont tous un nom différent. Il est sous-entendu que les parents ne cochent que des cours suivis par leur enfant. Par exemple, on peut conclure du formulaire précédent que la classe 4Ma a Y. Leterme comme professeur de français; les parents de Pierre Dupont ne demanderont donc pas de voir un autre professeur de français.

Suite aux formulaires, le lycée construit un horaire pour la soirée, tenant compte de la contrainte qu'aucune personne (professeur ou parent) ne peut participer simultanément à deux rencontres différentes. Les rencontres durent 15 minutes et commencent à 19h00, 19h15, 19h30, 19h45, 20h00, 20h15,...

La table suivante donne un exemple d'un tel horaire : P. Stanislav rencontrera la mère d'Eric Dufour à 19h30, puis les parents (mère et père) de Pierre Dupont; K. Depoortere rencontrera la mère de Pierre Dupont – à 19h15 pour le cours de physique et à 19h30 pour le cours de mathématiques – puis les parents d'Eric Dufour,... Notez que deux valeurs sont possibles pour les colonnes Père et Mère : la valeur "–" indique une absence, "x" indique une présence.

| Prénom | Nom | Classe | Mère | Père | Branche | Professeur | Heure |
|---|---|---|---|---|---|---|---|
| Pierre | Dupont | 4Ma | x | – | physique | K. Depoortere | 19h15 |
| Pierre | Dupont | 4Ma | x | – | mathématiques | K. Depoortere | 19h30 |
| Pierre | Dupont | 4Ma | – | x | français | Y. Leterme | 19h30 |
| Pierre | Dupont | 4Ma | x | x | biologie | P. Stanislav | 19h45 |
| Eric | Dufour | 5Lb | x | x | physique | K. Depoortere | 19h45 |
| Eric | Dufour | 5Lb | x | – | biologie | P. Stanislav | 19h30 |

1. Donnez les DF de ce schéma, dans la forme $X \to A$ avec $A$ un seul attribut. Évitez des DF redondantes. Les DF incorrectes seront pénalisées.
2. Donnez la clé (toutes les clés s'il y en a plusieurs) de ce schéma.
3. Est-ce que ce schéma est en 3NF ? Expliquez.

.../5

1. Les DF :



.../1

2. Les clés :

.../1

3. ☐ Oui, ce schéma est en 3NF. ☐ Non, ce schéma n'est pas en 3NF.

Cochez une case. Explication :

127

**Question 3** Pour la base de données de la question 1, donnez une requête, d'abord en algèbre SPJRUD, puis en calcul relationnel pour la question:

> Donnez, pour chaque rangée de la table LIGNES_DE_COMMANDES, le prix de la pizza commandée .

Notez que cette requête est monotone. Pour la base de données montrée, la réponse est :

| Ref | Pizza | Grandeur | Quantité | PrixUnité |
|---|---|---|---|---|
| Q123 | Forestière | medium | 1 | 14 |
| R456 | Pepperoni | large | 2 | 15 |
| R456 | Pepperoni | medium | 1 | 13 |
| R456 | Forestière | medium | 1 | 14 |
| S789 | Pepperoni | large | 3 | 15 |

Le calcul relationnel ne permettant pas de donner un nom aux colonnes, la colonne PrixUnité sera simplement la cinquième colonne dans la table qui résulte de la requête en calcul.

Vous pouvez utiliser les abréviations suivantes : L pour LIGNES_DE_COMMANDE, C pour COMMANDES, P pour PRIX. Soignez la lisibilité de vos requêtes. Pour la requête en algèbre, il est recommandé de faire un découpage en sous-requêtes (soit $R = \ldots$, soit $S = \ldots$, etc.).

Algèbre :

. . . /3

Calcul :

. . . /3



**Question 4** Considérez l'exécution $R_1(A)R_2(A)R_3(A)R_2(B)W_1(A)W_3(B)$. Est-ce que cette exécution peut se produire dans un système 2PL ? En d'autres termes, imaginez-vous un observateur externe qui enregistre toutes les écritures et lectures d'un système 2PL concret. Est-ce que cet observateur peut jamais enregistrer 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 de manière précise pourquoi l'exécution n'est pas possible en 2PL.

. . . /3

128

**Question 5** Pour la base de données de la question 1, donnez une requête, d'abord en algèbre SPJRUD, puis en calcul relationnel pour la question :

Donnez les références des commandes qui ne contiennent qu'une seule pizza.

Les requêtes peuvent utiliser l'égalité (=) mais pas d'ordre (donc, l'usage de $<, \leq, >, \geq$ est interdit). Pour la base de données montrée, la réponse est :

| Q123 |
|---|

Notez que la commande R456 contient 4 pizzas et n'est donc pas dans la réponse.

. . . /3

Algèbre :

. . . /3

Calcul :

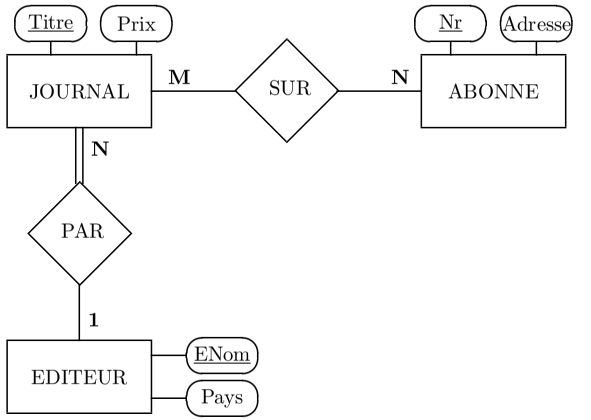


**Bases de Données I (J. Wijsen)**

18 janvier 2008

| | |
|---|---|
| NOM + PRENOM : | |
| Orientation + Année : | |

**Question 1** Donnez la traduction en modèle relationnel du schéma Entité-Association montré ci-après. Indiquez les clés primaires et étrangères.




129

. . . /4

**Question 2** Une entreprise de location de voitures permet à ses clients possédant un login de faire des réservations de voiture en ligne. Après login, le client est demandé :

1. de choisir un type de voiture;
2. de saisir une date de début de location; et
3. de saisir une date de fin de location.

Si la réservation est acceptée, un numéro unique y est accordé et les données sont stockées dans une table avec neuf colonnes :

| | |
|---|---|
| Nr | le numéro accordé à cette réservation |
| Type | le type de voiture choisi |
| Châssis# | le numéro de châssis de la voiture que sera mise à la disposition du client |
| DateRes | la date où la réservation est faite |
| DateLoc | la date de location |
| Login | un numéro unique identifiant le client |
| Nom | le nom du client |
| Statut | le statut du client. Par exemple, "A" pour les bons clients. |
| Prix | le prix journalier de location |

Chaque ligne correspond à une seule date de location. L'exemple montre une réservation faite par J. Bidon à la date du 22 août 2007 : la location concerne une Peugeot 404 pour quatre jours, du 24 au 27 août 2007. Le numéro de réservation attribué est 111; une nouvelle ligne est créée pour chaque jour de location. Le même jour (22 août 2007), P. Tape a introduit une réservation pour le weekend du 15 septembre; il s'agit également d'une Peugeot 404.

Le prix journalier dépend de trois facteurs : la date de réservation (les prix peuvent augmenter d'un jour au lendemain), le type de voiture et le statut du client. Par exemple, la table montre qu'à la date du 22 août 2007, le prix journalier pour une Peugeot 404 était de 93 EUR pour les clients avec un statut de A (peu importe la date de location), et 102 EUR pour les moins bons clients avec le statut D.

La colonne Statut donne le statut du client à la date de réservation. Le statut d'un client est unique pour un jour donné, mais peut évoluer d'un jour à l'autre. Le login est unique pour chaque client et ne change pas. Deux clients peuvent avoir le même nom.

Chaque voiture est identifiée par son numéro de châssis. Le type d'une voiture ne change jamais. Le système de réservation veille à ce qu'une même voiture ne puisse être louée qu'une seule fois pour une date donnée.

| Nr | Type | Châssis# | DateRes | DateLoc | Login | Nom | Statut | Prix |
|---|---|---|---|---|---|---|---|---|
| 111 | Peugeot 404 | C183 | 22/08/2007 | 24/08/2007 | 50123 | J. Bidon | A | 93 |
| 111 | Peugeot 404 | C183 | 22/08/2007 | 25/08/2007 | 50123 | J. Bidon | A | 93 |
| 111 | Peugeot 404 | C183 | 22/08/2007 | 26/08/2007 | 50123 | J. Bidon | A | 93 |
| 111 | Peugeot 404 | C183 | 22/08/2007 | 27/08/2007 | 50123 | J. Bidon | A | 93 |
| 222 | Peugeot 404 | S689 | 22/08/2007 | 15/09/2007 | 60456 | P. Tape | D | 102 |
| 222 | Peugeot 404 | S689 | 22/08/2007 | 16/09/2007 | 60456 | P. Tape | D | 102 |

Quelles sont les DF pour cette table ?



. . . /10

**Question 3** Soit $U = ABCDE$ et $\Sigma = \{AB \to CD, CD \to E, CE \to A\}$.

1. Démontrez que la décomposition en $ABCE$ et $CDE$ n'est pas *lossless join* (c'est-à-dire, ne préserve pas le contenu).
2. Donnez une seule DF $\sigma$ telle que la décomposition en $ABCE$ et $CDE$ est lossless join par rapport à $\Sigma \cup \{\sigma\}$.

L30

. . . /8

La preuve que le schéma n'est pas *lossless join* :

La DF $\sigma$ à ajouter :

**Question 4** Considérez le schéma avec attributs $ABCDEFGH$ et les DF suivantes :

| | | | | |
|---|---|---|---|---|
| $A \to E$ | $BE \to D$ | $AD \to BE$ | $BDH \to E$ | $CD \to A$ |
| $AC \to E$ | $F \to A$ | $E \to B$ | $D \to H$ | $BG \to F$ |

1. Expliquez pourquoi ce schéma n'est pas en 3NF.
2. Donnez une décomposition de ce schéma en 3NF; cette décomposition doit préserver à la fois le contenu et les DF. Expliquez vos calculs.

. . . /15

Ce schéma n'est pas en 3NF parce que :

Les calculs pour arriver à un schéma en 3NF :



**Question 5** Supposons que dans un système 2PL, une transaction $T_1$ effectue deux lectures du même objet $A$ (i.e. $T_1 = \ldots R_1(A) \ldots R_1(A) \ldots$). Entre ces deux lectures, la transaction $T_1$ n'effectue aucune écriture ni lecture de $A$. Supposons que la première lecture trouve que $A = 100$. Est-il possible que la deuxième lecture trouve une autre valeur pour $A$ (i.e. $A \neq 100$) ? Expliquez de façon précise.

. . . /8

131

**Question 6** Voici trois tables de la base de données du Lycée Royal M. Berrevoets.

- La table ELEVES mémorise les élèves. Chaque classe est identifiée par un code unique (1A, 1B,. . . ). Les élèves d'une classe sont identifiés de façon unique par un numéro d'ordre alphabétique (1, 2, 3, . . . ). Deux élèves peuvent porter le même nom. Néanmoins, les élèves d'une même classe ont tous un nom distinct.
- La première ligne de la table MATIERES indique que le professeur d'histoire de la classe 1A s'appelle C. Piot. Il n'y a pas deux professeurs avec le même nom. Un professeur peut enseigner plusieurs matières. Dans une classe, il n'y a qu'un seul professeur par matière enseignée. Les matières ne sont pas les mêmes pour toutes les classes (par exemple, certaines classes n'ont pas de cours de latin).
- La table BULLETIN indique, pour chaque étudiant, la note obtenue dans chacune des matières. Par exemple, la première ligne nous apprend qu'Ed Abicht a obtenu une note de 67% en histoire. Chaque étudiant aura une (et une seule) note pour chaque matière enseignée dans sa classe.

ELEVES

| Classe | Nr | Nom |
|---|---|---|
| 1A | 1 | Ed Abicht |
| 1A | 2 | Tom Beaudot |
| | ⋮ | |
| 1A | 23 | Jan Wouters |
| 1B | 1 | Piet Abeloos |
| 1B | 2 | Fred Duchamp |
| | ⋮ | |
| 1B | 22 | Pierre Watelet |
| 2A | 1 | Fred Apers |
| | ⋮ | |

MATIERES

| Classe | Matière | Prof |
|---|---|---|
| 1A | histoire | C. Piot |
| 1A | latin | A. De Vilder |
| 1A | gymnastique | A. Einstein |
| | ⋮ | |
| 1B | histoire | C. Piot |
| 1B | latin | N. Leterme |
| | ⋮ | |
| 2A | français | N. Leterme |
| 2A | grec | P. Jannssens |
| 2A | latin | N. Leterme |
| | ⋮ | |

BULLETIN

| Classe | Nr | Matière | Pourcentage |
|---|---|---|---|
| 1A | 1 | histoire | 67 |
| 1A | 1 | latin | 64 |
| | | ⋮ | |
| 1B | 2 | histoire | 100 |
| | | ⋮ | |
| 2A | 1 | grec | 45 |
| 2A | 1 | latin | 51 |
| | | ⋮ | |

Pour ces trois tables, donnez toutes les contraintes de type PRIMARY KEY, FOREIGN KEY et UNIQUE.

. . . /5



**Question 7** Voir question 6. Un étudiant a réussi si (et seulement si) il a obtenu 50% ou plus dans chacune des matières. Écrivez une requête SQL qui crée une liste des étudiants réussis. Pour chaque étudiant réussi, affichez sa classe, son numéro alphabétique, son nom et son pourcentage moyen (sur toutes les matières). Les étudiants non réussis n'apparaissent pas dans cette liste! Évidemment, la moyenne d'un étudiant réussi sera toujours supérieure ou égale à 50%, mais une moyenne au-dessus de 50% n'implique pas forcément la réussite. La réponse sera dans le format suivant :

| Classe | Nr | Nom | Moyenne |
|---|---|---|---|
| 1A | 1 | Ed Abicht | 56 |
| 1A | 2 | Tom Beaudot | 73 |
| | ⋮ | | |

. . . /6

132

**Question 8** Voir question 6. Écrivez une requête, d'abord en algèbre et puis en calcul, pour la question suivante :

> *Les langues classiques sont le latin et le grec. Pour chaque étudiant qui étudie toutes ces deux langues, donnez sa classe, son nom et son pourcentage en latin et en grec.*

Par exemple,

| Classe | Nom | Latin | Grec |
|---|---|---|---|
| 2A | Fred Apers | 51 | 45 |
| | ⋮ | | |

La requête en algèbre doit être **aussi succincte que possible.**

. . . /6

Algèbre :

. . . /6

Calcul :



**Question 9** Voir question 6. Écrivez une requête, d'abord en algèbre et puis en calcul, pour la question suivante :

> *Donnez les noms des professeurs qui n'enseignent qu'une seule matière.*

Par exemple, N. Leterme n'est certainement pas dans la réponse parce qu'il enseigne deux matières : latin et français. Si C. Piot n'enseigne que l'histoire, il sera dans la réponse.

. . . /6

Algèbre :

133

. . . /6

Calcul :

**Bases de Données I (J. Wijsen)**

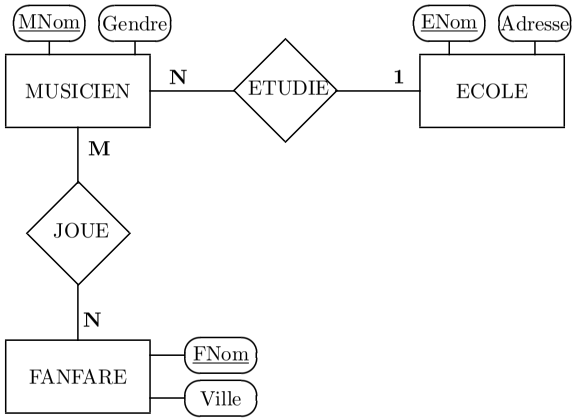
23 janvier 2009

NOM + PRENOM :

Orientation + Année :

Cet examen contient 11 questions.

**Question 1** Donnez la traduction en modèle relationnel du schéma Entité-Association montré ci-après. Indiquez les clés primaires et étrangères.




. . . /5



**Question 2** La table suivante stocke les excès de vitesse enregistrés par radar fixe :

| Date | Heure | NrRadar | RueRadar | VilleRadar | MaxVit | EffVit | Plaque |
|---|---|---|---|---|---|---|---|
| 2008-07-31 | 16:46:43 | R111 | Carrefour Léonard | Bruxelles | 90 | 134 | ABC789 |
| 2008-07-31 | 16:47:22 | R222 | Carrefour Léonard | Bruxelles | 70 | 105 | XYZ123 |
| 2008-07-31 | 22:34:16 | R111 | Carrefour Léonard | Bruxelles | 90 | 105 | ABC789 |
| 2008-08-01 | 06:13:13 | R222 | Carrefour Léonard | Bruxelles | 50 | 77 | DEF456 |

La première ligne indique qu'à la date du 31 juillet 2008 à 16 heures 46 minutes et 43 secondes, le radar avec numéro R111 a enregistré le passage du véhicule immatriculé ABC789 à une vitesse de 134 km/h (attribut `EffVit`, la vitesse effective). Ce radar se trouve au Carrefour Léonard à Bruxelles où la vitesse à cette date était limitée à 90 km/h (attribut `MaxVit`). Le même véhicule s'est fait "prendre" quelques heures plus tard au même endroit (voir troisième ligne).

Un autre radar au Carrefour Léonard est numéroté R222 et surveille les véhicules venant d'une autre direction. La limitation de vitesse n'est pas la même dans les deux directions. Notez que la limite de vitesse peut varier d'une date à l'autre : auprès du radar R222, la limite a été diminuée de 70 km/h (31 juillet) à 50 km/h (1 août) pour des raisons de travaux.

- Chaque radar a un numéro unique `NrRadar`.
- Les radars sont fixes et ne changent jamais d'endroit (rue + ville).
- Deux radars peuvent se trouver dans la même rue.
- Un radar ne sait pas enregistrer deux infractions au même moment.
- Il est impossible que la même plaque soit enregistrée au même moment par deux radars différents.
- La vitesse maximale `MaxVit` d'un radar ne change pas au cours d'une même date. Cependant, les limites de vitesses peut être adaptées d'un jour au lendemain (par exemple, pour raison de smog).
- La vitesse maximale peut varier au sein d'une même rue. Deux radars peuvent donc viser une autre vitesse maximale, même s'ils se trouvent dans la même rue.
- Les radars n'enregistrent que des excès de vitesse. La vitesse effective sera donc toujours supérieure à la vitesse maximale.

Quelles sont les dépendances fonctionnelles pour cette table ?

. . . /5

**Question 3** *Formule 1* est une chaîne française d'hôtels de passage. Chaque hôtel a un code et un nom unique. Par exemple, l'hôtel avec code 3488 s'appelle *Paris Porte de Montreuil*. Les chambres de chaque hôtel sont numérotées 1, 2, 3,... Chaque séjour est couvert par un (et un seul) numéro de carte bancaire; les clients mandatent *Formule 1* pour débiter ce compte de la somme due.

La table HOTELS enregistre le nom et le nombre de chambres de chaque hôtel. La table CHAMBRES enregistre le nombre de lits par chambre. La colonne `Fumer` indique si fumer dans la chambre est permis ou interdit. La table SEJOURS enregistre quelles chambres sont occupées à quelle date, avec la carte de crédit à débiter. Une même carte peut couvrir plusieurs séjours avec la même date (par exemple, une grande famille peut occuper deux chambres payées avec la même carte bancaire). Néanmoins, pour des raisons de gestion des risques, une même carte ne peut pas couvrir deux séjours simultanés (i.e. avec la même date) dans deux hôtels différents. La colonne `Porteur` enregistre le nom unique imprimé sur la carte bancaire. Noter qu'une personne peut posséder plusieurs cartes.

HOTELS

| Code | Nom | NombreDeChambres |
|---|---|---|
| 3488 | Paris Porte de Montreuil | 228 |
| 2539 | Paris Porte de Saint Ouen | 386 |
| ⋮ | | |

CHAMBRES

| CodeHotel | NrChambre | NombreDeLits | Fumer |
|---|---|---|---|
| 3488 | 1 | 2 | permis |
| 3488 | 2 | 4 | interdit |
| 2539 | 1 | 4 | interdit |
| ⋮ | | | |

SEJOURS

| CodeHotel | NrChambre | Date | CarteBancaire | Porteur |
|---|---|---|---|---|
| 3488 | 1 | 2008-08-23 | 1111 2222 3333 4444 | Jimmy Johnson |
| 3488 | 2 | 2008-08-23 | 1111 2222 3333 4444 | Jimmy Johnson |
| 3488 | 2 | 2008-08-24 | 1111 2222 3333 4444 | Jimmy Johnson |
| 2539 | 1 | 2008-08-24 | 5555 6666 7777 8888 | Sven Smith |
| ⋮ | | | | |

Pour ces trois tables, donnez toutes les contraintes de type PRIMARY KEY, FOREIGN KEY et UNIQUE.

. . . /5



**Question 4** Est-ce que l'exécution $R_3(D)W_1(A)R_2(B)W_1(B)R_2(D)W_3(B)$ est possible en 2PL dans exactement cet ordre ?

. . . /5

135

**Question 5** Considérez le schéma avec attributs $ABCDEFG$ et les DF suivantes :

$$\Sigma = \{\ AB \to C,\quad C \to A,\quad ABC \to D,\quad ACD \to B,\quad D \to E,\quad DE \to G,\quad BE \to C,\quad CG \to B,\quad BCG \to D,\quad CE \to A,\quad ACE \to G\ \}$$

Notez qu'il n'y a aucune DF qui contient $F$.

1. Donnez un ensemble $\Sigma'$ de DF tel que $\Sigma' \equiv \Sigma$ et $\Sigma'$ est irréductible.
2. Est-ce que ce schéma est en 3NF ? Expliquez de manière précise.

. . . /10



**Question 6** L'opérateur *semijoin*, dénoté par $\ltimes$, est défini comme suit:

**Syntax:** $R \ltimes S$, sans restriction sur les schémas de $R$ et $S$.

**Sémantique:** Soit $R$ une relation sur $U$ et $S$ une relation sur $W$. Soit $V = U \cap W$. Alors,

$$R \ltimes S := \{t \in R \mid \exists s \in S : s[V] = t[V]\}$$

Est-ce que l'opérateur $\ltimes$ peut être exprimé en algèbre SPJRUD ? Si oui, donnez une expression équivalente en algèbre SPJRUD. Si non, expliquez pourquoi c'est impossible.

. . . /5

136

**Question 7** Donnez une exécution *sérialisable* qui contient deux transactions $T_i$ et $T_k$ telle que :

1. la dernière action de $T_i$ précède la première action de $T_k$, et
2. le graphe de précédence contient un chemin orienté de $T_k$ vers $T_i$.

Ou expliquez pourquoi une telle exécution n'existe pas. . . Notez que l'exécution peut contenir des transactions autres que $T_i$ et $T_k$.

. . . /5

**Question 8** On peut distinguer deux types de transactions :

- Les transactions *read-only* : les transactions qui n'effectuent aucune écriture.
- Les transactions *not-read-only* : les transactions qui effectuent au moins une écriture.

La notion d'*exécution sérialisable* vue au cours ne distingue pas entre ces deux types de transactions. Néanmoins, après de longues réflexions nocturnes, un étudiant (appelons-le étudiant X) conclut que, en ce qui concerne la "sérialisabilité", on pourrait ignorer les transactions *read-only* pour la simple raison que ces transactions ne modifient pas la base de données et ne peuvent donc pas y causer des "dégâts".

Soit $E$ une exécution de transactions. Soit $\overline{E}$ l'exécution que l'on obtient quand on enlève de $E$ toutes les actions de chaque transaction qui est *read-only*. Par exemple, puisque $T_1$ et $T_3$ sont *read-only* dans l'exécution

$$E_0 = R_1(A)R_2(A)R_2(B)W_2(A)W_2(B)R_1(B)R_3(C)R_4(C)W_2(D)W_4(C) \ ,$$

on obtient

$$\overline{E_0} = R_2(A)R_2(B)W_2(A)W_2(B)R_4(C)W_2(D)W_4(C) \ .$$

L'étudiant X appelle une exécution $E$ *semi-sérialisable* si $\overline{E}$ est sérialisable. Il est facile de vérifier que l'exécution $E_0$ ci-dessus est semi-sérialisable dans ce sens, mais pas sérialisable. Évidemment, chaque exécution sérialisable est aussi semi-sérialisable.

Puisque les transactions *read-only* ne peuvent pas modifier la base de données, l'étudiant X conclut que chaque exécution semi-sérialisable est correcte (et que la sérialisabilité classique vue au cours de Bases de Données I est donc une exigence trop forte). Cochez une case : ☐ l'étudiant X a raison
☐ l'étudiant X se trompe

Argumentez votre choix.

. . . /5



**Question 9** Voir question 3. Donnez une requête SQL qui donne le nom de l'hôtel avec le plus grand nombre de lits. Notez que si l'hôtel $A$ a plus de chambres que l'hôtel $B$, $A$ n'a pas forcément plus de lits.

. . . /5

137

**Question 10** Voir question 3. Écrivez une requête, d'abord en algèbre et puis en calcul, pour la question suivante :

*Les deux types de chambres sont "fumer permis" et "fumer interdit". Donnez les noms des hôtels où toutes les chambres ne sont pas de même type.*

. . . /5

Algèbre :

. . . /5

Calcul :



**Question 11** Voir question 3. Écrivez une requête, d'abord en algèbre et puis en calcul, pour la question suivante :

*Donnez les noms des hôtels où chaque chambre "fumer permis" ne contient qu'un seul lit.*

Dans un tel hôtel, chaque fumeur qui souhaite fumer dans sa chambre, sera seul dans sa chambre.

. . . /5

Algèbre :

138

. . . /5

Calcul :

## Bases de Données I (J. Wijsen)

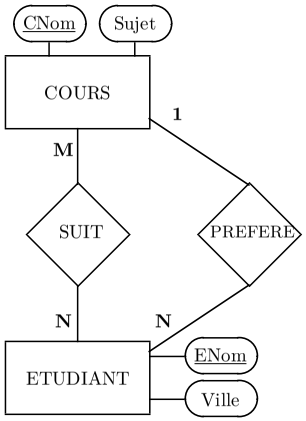
22 janvier 2010

NOM + PRENOM :

Orientation + Année :

Cet examen contient 10 questions. Durée : 3 heures.

**Question 1** Donnez la traduction en modèle relationnel du schéma Entité-Association montré ci-après. Indiquez les clés primaires et étrangères.




. . . /4



**Question 2** Considérez le schéma avec attributs $ABCDEI$ et les DF suivantes :

$$A \to C \qquad AB \to C \qquad C \to DI$$
$$EI \to C \qquad CD \to I \qquad EC \to AB$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Ce schéma est en BCNF.
- ☐ Ce schéma est en 3NF.
- ☐ Ce schéma n'est pas en 3NF.

Détaillez les calculs qui mènent à cette conclusion.

. . . /6

139

**Question 3** Si $E$ est une requête SPJRUD et **bd** une base de données, alors on écrit $E(\mathbf{bd})$ pour dénoter le résultat de la requête $E$ sur **bd**. Disons que deux requêtes SPJRUD $E_1$ et $E_2$ sont *équivalentes*, dénoté par $E_1 \equiv E_2$, si pour toute base de données **bd**, $E_1(\mathbf{bd}) = E_2(\mathbf{bd})$. Soient $R$ et $S$ deux noms de relation sur le même schéma $AB$. Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ $\pi_A(R \bowtie S) \equiv \pi_A(R) \bowtie \pi_A(S)$
- ☐ $\pi_A(R \cup S) \equiv \pi_A(R) \cup \pi_A(S)$
- ☐ $\pi_A(R \setminus S) \equiv \pi_A(R) \setminus \pi_A(S)$

Détaillez votre réponse (i.e. pour chaque expression, expliquez pourquoi elle est correcte ou fausse).

.../6



**Question 4** Considérez les requêtes :

$$q_1 = \{x \mid \exists y(R(x,y) \wedge \neg S(y))\} \text{ et } q_2 = \{x \mid \exists y(R(x,y) \vee S(y))\}$$

Les requêtes $q_1$ et $q_2$ ne sont pas dans le *Safe Relational Calculus* (SRC) présenté au cours, parce que :

1. $q_1$ contient $R(x,y) \wedge \neg S(y)$ et $\{x,y\} = \mathit{free}(R(x,y)) \neq \mathit{free}(S(y)) = \{y\}$.
2. $q_2$ contient $R(x,y) \vee S(y)$ et $\{x,y\} = \mathit{free}(R(x,y)) \neq \mathit{free}(S(y)) = \{y\}$.

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ $q_1$ est *domain independent*.
- ☐ $q_1$ n'est pas *domain independent*.
- ☐ $q_2$ est *domain independent*.
- ☐ $q_2$ n'est pas *domain independent*.

Détaillez votre réponse.

.../6

140

**Question 5** Le protocole 2PL contient trois règles, à savoir :

**R1** Dans une même transaction, chaque lecture d'un objet est précédée par la demande d'un verrou de type $S$ sur cet objet, et suivie d'un relâchement de ce verrou. Également, chaque écriture d'un objet est précédée par la demande d'un verrou de type $X$ sur cet objet, et suivie d'un relâchement de ce verrou.

**R2** Dans une même transaction, le relâchement d'un verrou ne peut pas être suivi d'une demande de verrou (sur le même objet ou sur un autre objet).

**R3** Si une transaction possède un verrou de type $X$ sur un objet, aucune autre transaction ne peut posséder un verrou (de type $S$ ou $X$) sur ce même objet.

Appelons ProL le protocole qui contient les règles R1 et R3, et qui remplace R2 par une nouvelle règle Rbis :

**Rbis** Dans une même transaction, le relâchement d'un verrou sur un objet ne peut pas être suivi d'une demande de verrou *sur le même objet*.

Donc, la règle Rbis permet $\ldots U_1(A) \ldots X_1(B) \ldots$, parce que $A \neq B$. La règle Rbis interdit $\ldots U_1(A) \ldots X_1(A) \ldots$

Démontrez **de façon précise** que le protocole ProL={R1,Rbis,R3} ne garantit pas que les exécutions soient sérialisables.

.../6



**Question 6** Les trois tables suivantes concernent l'organisation des cours et travaux pratiques au sein de l'Institut d'Informatique de l'UMons. La table COURS enregistre, pour chaque cours, le professeur, le nombre d'heures de théorie et le nombre d'heures de travaux pratiques. Les professeurs enseignent la théorie ; des assistants peuvent intervenir pour l'organisation des travaux pratiques. La table TP enregistre la répartition des heures de travaux pratiques parmi les professeurs et les assistants. Notez que les professeurs peuvent assurer des travaux pratiques. Par exemple, Pierre Lesage assure 10 heures de travaux pratiques dans un cours de Paul Dumont ; et Paul Dumont assure une partie des travaux pratiques de son cours de Programmation. Finalement, la table PERSONNEL donne le statut et le genre de chaque personne affiliée à l'Institut. Supposons qu'il n'y ait jamais deux personnes avec le même nom dans l'Institut.

COURS

| Cours | Prof | Théorie | TravauxPratiques |
|---|---|---|---|
| XML | Pierre Lesage | 30 | 20 |
| Data mining | Pierre Lesage | 20 | 20 |
| Calculabilité | Patricia Lion | 30 | 45 |
| SQL | Paul Dumont | 30 | 60 |
| Programmation | Paul Dumont | 30 | 30 |
| Algorithmique | Paul Dumont | 30 | 30 |
| Réseaux | Pierre Piston | 30 | 30 |

TP

| Personne | Cours | Heures |
|---|---|---|
| Patricia Lion | XML | 10 |
| Arie Jon | XML | 10 |
| Anne Dupont | Data mining | 20 |
| Alex Dupré | Calculabilité | 45 |
| Anne Dupont | SQL | 35 |
| Alex Dupré | SQL | 15 |
| Pierre Lesage | SQL | 10 |
| Paul Dumont | Programmation | 15 |
| Alex Dupré | Programmation | 15 |
| Arie Jon | Algorithmique | 15 |
| Anne Dupont | Algorithmique | 20 |
| Anne Dupont | Réseaux | 20 |
| Astrid Six | Réseaux | 10 |

PERSONNEL

| Nom | Statut | Genre |
|---|---|---|
| Patricia Lion | prof | F |
| Paul Dumont | prof | M |
| Pierre Lesage | prof | M |
| Pierre Piston | prof | M |
| Astrid Six | assistant | F |
| Alex Dupré | assistant | M |
| Anne Dupont | assistant | F |
| Arie Jon | assistant | M |
| Marie Geraerts | secrétaire | F |

141

Pour ces trois tables, donnez toutes les contraintes de type PRIMARY KEY, FOREIGN KEY et UNIQUE.

.../6

**Question 7** Écrivez une requête en **algèbre relationnelle** pour répondre à la question suivante :

Donnez le nom de chaque professeur qui n'assure qu'un seul cours.

Pour l'exemple, la réponse est *Patricia Lion* et *Pierre Piston*.

.../4

**Question 8** Le secrétariat doit vérifier que pour chaque cours, la somme des heures de travaux pratiques dans la table TP est égale à la valeur de l'attribut `TravauxPratiques` dans la table COURS. Donnez une requête SQL qui donne le nom de chaque cours pour lequel la somme des heures de travaux pratiques ne correspond pas à la valeur dans la table COURS.

Dans l'exemple, seul le cours d'*Algorithmique* pose problème parce que les assistants assurent $15 + 20 = 35$ heures pour ce cours, bien que le cours ne contienne que 30 heures de travaux pratiques.

.../4



**Question 9** Écrivez une requête en **Safe Relational Calculus** pour répondre à la question suivante :

Donnez le nom de chaque assistant qui n'intervient pas dans le cours d'SQL.

Pour l'exemple, la réponse est *Astrid Six* et *Arie Jon*.

.../4

**Question 10** La table suivante stocke l'historique des **matchs de foot de qualification** pour le championnat du monde (Mondial). La phase de qualification se déroule pendant les vingt-quatre mois qui précèdent la phase finale. La phase finale du Mondial 2010 aura lieu en Afrique du Sud du 11 juin jusqu'au 11 juillet 2010. Les précédentes éditions étaient le Mondial 2006, le Mondial 2002, le Mondial 1998,...(i.e. tous les quatre ans). On a donc :

| Phase de qualification | | Mondial |
|---|---|---|
| 2008-2010 | → | Mondial 2010 |
| 2004-2006 | → | Mondial 2006 |
| 2000-2002 | → | Mondial 2002 |
| ⋮ | | ⋮ |

Notez que seuls les matchs de qualification sont stockés dans la bases de données ; les matchs de la phase finale n'y figurent pas. Par ailleurs, on ne stocke que les matchs des pays européens.

Dans la phase de qualification de chaque Mondial, les pays européens (une cinquantaine) sont divisés en groupes de cinq ou six pays. Ces groupes sont le résultat d'un tirage au sort. Chaque groupe est identifié par un chiffre (groupe 1, 2, 3,...). Pour le Mondial 2010, la Belgique faisait partie du groupe 5 ; les autres pays dans ce groupe étaient l'Arménie, la Bosnie-Herzégovine, l'Espagne, l'Estonie et la Turquie. La France et l'Autriche faisaient partie du groupe 7. Les groupes ne changent pas pendant la phase de qualification. Chaque pays joue deux fois contre chaque autre pays de son groupe, une fois à la maison et une fois en déplacement. Le gagneur de chaque groupe se qualifie pour la phase finale ; les deuxièmes jouent des matchs de barrage (mais ces matchs de barrage ne sont pas enregistrés dans la bases de données). Évidemment, la division en groupes change d'un mondial à l'autre. Pour le Mondial 2006, la Belgique faisait partie du groupe 7 avec, entre autres, l'Espagne et la Serbie.

Le premier enregistrement indique que le 6 septembre 2008, l'Autriche a battu la France 3 buts à 1 dans un match de qualification pour le Mondial 2010. Ce match avait lieu à Vienne. Un pays ne joue qu'un seul match par jour. Un match entre pays X et pays Y a toujours lieu dans une grande ville du pays X. Supposons que le nom de chaque ville est unique.

| Année | Jour | Maison | Visiteurs | Pour | Contre | Tournoi | Groupe | Ville |
|---|---|---|---|---|---|---|---|---|
| 2008 | 6 sep | Autriche | France | 3 | 1 | Mondial 2010 | 7 | Vienne |
| 2008 | 6 sep | Belgique | Estonie | 3 | 2 | Mondial 2010 | 5 | Liège |
| 2008 | 10 sep | Turquie | Belgique | 1 | 1 | Mondial 2010 | 5 | Istanbul |
| 2009 | 10 oct | Belgique | Turquie | 2 | 0 | Mondial 2010 | 5 | Bruxelles |
| 2009 | 14 oct | Estonie | Belgique | 2 | 0 | Mondial 2010 | 5 | Tallinn |
| 2009 | 14 oct | France | Autriche | 3 | 1 | Mondial 2010 | 7 | Paris |
| | | | | | | | | |
| 2004 | 17 nov | Belgique | Serbie | 0 | 2 | Mondial 2006 | 7 | Bruxelles |
| 2005 | 8 oct | Belgique | Espagne | 0 | 2 | Mondial 2006 | 7 | Bruxelles |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Quelles sont les dépendances fonctionnelles pour cette table ?

.../10



## Bases de Données I (J. Wijsen), 20 janvier 2011

NOM + PRENOM :

Orientation + Année :

Cet examen contient 11 questions. Durée : 3 heures.

**Question 1** Soit $R$ un nom de relation avec schéma $AB$. Soit $S$ un nom de relation avec schéma $B$. Soit $T$ un nom de relation avec schéma $B$. Considérez la requête $q_1$ :

$$q_1 = \{x \mid \exists y\Big(R(x,y) \wedge \neg\big(S(y) \vee T(y)\big)\Big)\}$$

Donnez une requête en algèbre relationnelle qui est équivalente à $q_1$. Détaillez votre réponse.

.../5

143

**Question 2** Considérez le schéma avec attributs $ABCDEF$ et les DF suivantes :

$$\begin{array}{lll} ABCEF \to D & ABD \to C & EF \to ABC \\ AC \to E & EB \to F & AF \to BC \end{array}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Ce schéma est en BCNF.
- ☐ Ce schéma n'est pas en BCNF.
- ☐ Ce schéma est en 3NF.
- ☐ Ce schéma n'est pas en 3NF.

Détaillez les calculs qui mènent à cette conclusion.

.../10



**Question 3** Considérez la requête :

$$\{x \mid \Big(\forall y\big(R(x,y) \to \neg T(y)\big) \wedge \exists z\big(R(x,z)\big)\Big) \vee T(x)\}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Cette requête est *domain independent*.
- ☐ Cette requête n'est pas *domain independent*.

Expliquez en détail.

.../5

144

**Question 4** Considérez la requête :

$$\{x, y \mid \big(R(x, y) \wedge \neg S(x)\big) \vee \big(\neg R(x, y) \wedge S(x)\big)\}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

☐ Cette requête est *domain independent*.

☐ Cette requête n'est pas *domain independent*.

Expliquez en détail.

.../5



**Question 5** Si $E$ est une requête SPJRUD et **bd** une base de données, alors on écrit $E(\mathbf{bd})$ pour dénoter le résultat de la requête $E$ sur **bd**. Disons que deux requêtes SPJRUD $E_1$ et $E_2$ sont *équivalentes*, dénoté par $E_1 \equiv E_2$, si pour toute base de données **bd**, $E_1(\mathbf{bd}) = E_2(\mathbf{bd})$. Soient $R$ et $S$ deux noms de relation avec le même schéma $AB$. Soit $T$ un nom de relation avec schéma $BC$. Quelles expressions sont correctes ? Cochez deux cases.

| | **Vrai** | **Faux** |
|---|---|---|
| $\pi_{AB}(R \bowtie \pi_C(T)) \equiv R$ | ☐ | ☐ |
| $(R - S) \bowtie T \equiv (R \bowtie T) - (S \bowtie T)$ | ☐ | ☐ |

Détaillez votre réponse (i.e. pour chaque expression, expliquez pourquoi elle est vraie ou fausse).

.../10

145

**Question 6** Quelle est la motivation principale pour passer de 2PL à Strict 2PL ? Expliquez de façon précise à l'aide d'un exemple.

.../10



**Question 7** Le site web `www.obay.com` sert à la vente des articles aux enchères. Les candidats vendeurs et acheteurs doivent s'inscrire une fois comme membre du site. Au moment de l'inscription, ils choisissent un pseudonyme unique. La table `ÉTALAGE` enregistre les articles à vendre. Chaque article est identifié par un nombre unique (`NrArticle`) ; on enregistre la date où l'article a été mis en vente, le membre qui a mis l'article, et le prix souhaité. La table `OFFRES` enregistre qui veut acheter quel article et à quel prix ; on enregistre aussi la date de l'offre. Notez qu'une même personne peut faire plusieurs offres au cours du temps. Cependant, pour ne pas exciter le marché, un membre ne peut pas faire plusieurs offres par jour *pour le même article*.

`ÉTALAGE`

| Vendeur | NrArticle | Description | Date | PrixDemandé |
|---|---|---|---|---|
| Jessica | 12 | vase chinoise | 12 juin 2010 | 100 |
| Tintin | 13 | agrafeuse | 13 juin 2010 | 5 |
| Tintin | 14 | perforeuse | 13 juin 2010 | 6 |
| Jessica | 15 | peinture Van Gogh | 13 juin 2010 | 50 |

`OFFRES`

| Enchérisseur | NrArticle | Date | PrixOffert |
|---|---|---|---|
| Geoffrey | 12 | 13 juin 2010 | 101 |
| Tintin | 12 | 14 juin 2010 | 102 |
| Geoffrey | 12 | 15 juin 2010 | 103 |
| Geoffrey | 14 | 15 juin 2010 | 6 |
| Geoffrey | 15 | 13 juin 2010 | 50 |
| Tintin | 15 | 13 juin 2010 | 55 |

`MEMBRES`

| Pseudonyme | Statut | Prénom | Nom | Ville |
|---|---|---|---|---|
| Jessica | A | Virgina | Wulf | Namur |
| Geoffrey | B | Geoffrey | Dupont | Dinant |
| Tintin | A | Jean | Hergé | Namur |
| Pirlo | A | Pierre | Louagie | Mons |

Pour ces trois tables, donnez toutes les contraintes de type PRIMARY KEY, FOREIGN KEY et UNIQUE.

146 .../5

**Question 8** Écrivez une requête en **Safe Relational Calculus** pour répondre à la question suivante :

Donnez le numéro de chaque article pour lequel le prix de chaque offre est différent du prix demandé.

Pour l'exemple, 12 apparaît dans la réponse, parce que les prix offerts pour cet article (101, 102, 103) sont tous différents du prix demandé (100). Par contre, 15 n'apparaît pas dans la réponse, parce qu'il y a eu une offre dont le prix est égal au prix demandé (50).

.../5

**Question 9** Supposez que chaque produit est vendu au prix de l'offre la plus élevée. Donnez une requête SQL qui donne pour chaque membre la somme qu'il touchera, pourvu que cette somme soit non nulle. Pour l'exemple, la réponse est :

| | |
|---|---|
| Jessica | 158 |
| Tintin | 6 |

.../5



**Question 10** Écrivez une requête en **algèbre relationnelle** pour répondre à la question suivante :

Donnez le nom de chaque membre qui n'a jamais fait deux offres pour un même article.

Pour l'exemple, la réponse est *Jessica*, *Tintin* et *Pirlo*. Notez que *Geoffrey* n'est pas dans la réponse, parce qu'il a fait deux offres pour l'article 12.

.../5

147

**Question 11** La table `EXAMENS` stocke le calendrier des examens de la seconde session. Chaque étudiant est inscrit dans une et une seule section parmi bio, chimie, info, math, physique. Chaque étudiant est inscrit dans une et une seule année parmi BAC1, BAC2, BAC3, MAS1, MAS2. Chaque étudiant est identifié par un numéro unique (`E#`). Chaque cours est identifié par un intitulé unique et est enseigné par un seul professeur. Un cours peut être enseigné dans deux sections différentes. Par exemple, le cours d'histologie est au programme en BAC2 bio et BAC3 chimie. Un cours n'est jamais au programme de deux années différentes de la même section. Pour chaque cours enseigné dans une section, le professeur a le choix entre soit un examen oral soit un examen écrit (jamais les deux). Notez que l'examen d'histologie est écrit pour la section bio et oral pour la section chimie, ce qui est tout à fait permis. L'examen oral d'un cours peut être organisé sur plusieurs journées. Par exemple, l'examen oral d'histologie en section chimie aura lieu le 3 et 4 septembre. Cependant, les professeurs ne préparent qu'un seul examen écrit par cours; tous les étudiants doivent passer cet examen au même moment. Par exemple, puisque l'examen de BDI est un examen écrit dans les sections math et info, les étudiants de ces deux sections passent cet examen le même jour. Un professeur n'organisera jamais deux examens de deux cours différents au même jour. Également, un professeur n'organisera jamais un oral et un écrit au même jour. Un étudiant ne peut pas passer deux fois l'examen oral d'un même cours. Un étudiant n'a jamais deux examens au même jour.

| EXAMENS | E# | Section | Année | Cours | Prof | Date | Oral | Ecrit |
|---|---|---|---|---|---|---|---|---|
| | 111 | bio | BAC2 | histologie | Radoux | 2 sep | non | oui |
| | 222 | bio | BAC2 | histologie | Radoux | 2 sep | non | oui |
| | 333 | chimie | BAC3 | histologie | Radoux | 3 sep | oui | non |
| | 444 | chimie | BAC3 | histologie | Radoux | 4 sep | oui | non |
| | 555 | info | BAC2 | Prolog | Mens | 1 sep | non | oui |
| | 555 | info | BAC2 | BDI | Wijsen | 3 sep | non | oui |
| | 666 | math | BAC2 | BDI | Wijsen | 3 sep | non | oui |

Quelles sont les dépendances fonctionnelles pour cette table ?

.../10



## Bases de Données I (J. Wijsen)

27 janvier 2012

NOM + PRENOM :

Orientation + Année :

Cet examen contient 14 questions. Durée : 2 heures et 30 minutes.

**Question 1** Soit $R$ un nom de relation avec schéma $AB$. Soit $S$ un nom de relation avec schéma $A$. Considérez la requête $q_1$ :

$$q_1 = \{x \mid \exists y\big(R(x,y) \wedge \neg(S(x) \wedge S(y))\big)\}$$

Donnez une requête en algèbre relationnelle qui est équivalente à $q_1$. Détaillez votre réponse.

.../5

Chaque tuple $\{A:a, B:b\}$ dans $R$ contribue "a" à la réponse sauf si $S$ contient à la fois $\{A:a\}$ et $\{A:b\}$.
Les tuples de $R$ qui ne contribuent pas à la réponse sont donc tous dans :

$$S \bowtie \rho_{A \to B} S .$$

La requête suivante est donc équivalente à $q_1$ :

$$\pi_A\left(R - \left(S \bowtie \rho_{A \to B} S\right)\right) .$$

148

**Question 2** Considérez la requête :

$$\{x, y \mid R(x, y) \vee \exists y(R(x, y) \wedge \neg S(x))\}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Cette requête est *domain independent*.
- ☒ Cette requête n'est pas *domain independent*.

Expliquez en détail.

…/5

Prenons R | 1 2 | a c  S | 1 | → S est vide

Pour cette base de données, ⟨a, b⟩ est dans la réponse pour un b quelconque, car $\exists y\,(R(a, y) \wedge \neg S(a))$ est vrai.

x → a, y → b

↳ Notez que ceci n'est pas "le même y" que celui au début de la requête !

**Question 3** Considérez la requête :

$$\{y \mid \exists x \neg \exists z(\neg R(x, y) \vee R(z, y))\}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☒ Cette requête est *domain independent*.
- ☐ Cette requête n'est pas *domain independent*.

Expliquez en détail.

…/5

Pour toute base de données, la réponse sera toujours vide. Supposons que la réponse n'est pas vide pour une bd. Alors, il existe b et a tel que la bd satisfait :

$$\neg \exists z\,(\neg R(a, b) \vee R(z, b)),$$

ce qui est équivalent à :

$$R(a, b) \wedge \forall z\,(\neg R(z, b)).$$

Donc, bd doit contenir R(a, b) et ne pas contenir R(a, b), une contradiction.

x → a, y → b



**Question 4** Quelle est la motivation principale pour passer de 2PL à Strict 2PL ? Expliquez de façon précise à l'aide d'un exemple.

…/5

Voir Syllabus.

149

**Question 5** Prouvez qu'il existe une requête $q$ en algèbre SPJRUD qui n'est équivalente à aucune requête en algèbre SPJRD (i.e., l'algèbre sans l'union). N'oubliez pas de préciser les attributs des relations qui figurent dans $q$.

.../5

Cette question est résolue dans le Syllabus.



**Question 6** Considérez le schéma avec attributs $ABCDEFGH$ et les DF suivantes :

$$\Sigma = \begin{array}{lll} AB \to CD & EF \to GH & BCD \to A \\ G \to E & B \to F & F \to A \end{array}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Ce schéma est en BCNF.
- ☒ Ce schéma n'est pas en BCNF.
- ☐ Ce schéma est en 3NF.
- ☒ Ce schéma n'est pas en 3NF.

Détaillez les calculs qui mènent à cette conclusion.

.../5

Chaque clé doit contenir B, parce que B n'apparaît pas à droite d'une flèche.

$\Sigma \vdash B \to BFACD$

Donc, aucune clé ne contient un attribut parmi A, C, D, F.

$\Sigma \vdash BE \to BFACD(E)GH$ BE est une clé

$\Sigma \vdash BG \to BFACD(G)EH$ BG est une clé

$\Sigma \not\vdash BH \to G$, donc BH n'est pas une clé

Il est correct de conclure que BE et BG sont les seules clés.

Ce schéma n'est pas en 3NF (et donc pas en BCNF) parce que

$\Sigma$ contient $B \to F$,
B n'est pas supercle, et
F ne fait partie d'aucune clé.

150

**Question 7** Considérez le schéma avec attributs $ABCDE$ et les DF suivantes :

$$\Sigma = \{ AB \to D \qquad ABC \to E \qquad CDE \to AB \qquad ADE \to C \}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Ce schéma est en BCNF.
- ☒ Ce schéma n'est pas en BCNF.
- ☒ Ce schéma est en 3NF.
- ☐ Ce schéma n'est pas en 3NF.

Détaillez les calculs qui mènent à cette conclusion.

.../5

$\Sigma \models ABC \to DE$, donc ABC est supercle.

On peut vérifier que AB, AC, BC ne sont pas superclés. Donc ABC est une clé.

$\Sigma \models CDE \to AB$, donc CDE est supercle.

On peut vérifier que CD, CE, DE ne sont pas superclés. Donc CDE est une clé.

Le schéma est en 3NF, parce que chaque attribut fait partie d'une clé.

Pas en BCNF, car $\Sigma$ contient $AB \to D$, mais AB n'est pas supercle.



**Question 8** Les *20 km de Bastogne* est une course à pieds organisée annuellement depuis 1980. La 33ème édition aura lieu le 27 mai 2012. La table RESULTATS stocke les résultats de toutes les éditions passées :

- Chacun des athlètes porte un dossard avec un numéro unique. Le dossard d'un athlète peut varier d'une année à l'autre.
- `CP` est le code postal de la localité belge de l'athlète ; pour les athlètes non domiciliés en Belgique, cette colonne donne le pays de l'athlète. Supposons que pour une même édition, les attributs `Nom` et `CP` constituent une identification unique pour chaque athlète. Donc, si deux participants à une même édition ont le même nom, ils auront une valeur différente pour `CP`. Par contre, il n'est pas sûr que l'athlète Anne Dua avec code postal 3700 en 1980 est la même personne physique qu'Anne Dua avec code postal 3700 en 2011.
- `Nat` donne la nationalité de l'athlète (B=Belgique, F=France,...).
- Pour chaque édition, on fait deux classements : un pour les athlètes masculins et un pour les athlètes féminins. Pour une édition donnée, le `Rang` d'un athlète masculin est $n$ s'il y a exactement $n-1$ athlètes masculins ayant réalisé un meilleur temps. De façon similaire, le `Rang` d'une femme est $n$ s'il y a exactement $n-1$ femmes ayant couru plus vite. Notez qu'en 2011, Hugo Claus et Ivo Michiels ont réalisé le même temps et sont classés 2èmes tous les deux ; par conséquent, aucun athlète n'avait le `Rang` 3 en 2011.
- `Moyenne` donne la vitesse moyenne (en km/h) de chaque athlète. Le parcours a toujours été exactement 20 km ; la vitesse moyenne pourrait donc être calculée à partir du `Temps`.

RESULTATS

| Edition | Année | Dossard | Nom | CP | Nat | Sexe | Rang | Temps | Moyenne |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1980 | 17 | Anne Dua | 3700 | B | F | 1 | 1:15:00 | 16.00 |
| 1 | 1980 | 29 | Eric Point | 8100 | F | M | 1 | 1:00:00 | 20.00 |
| 32 | 2011 | 37 | Tom Michiels | 9473 | B | M | 1 | 1:00:00 | 20.00 |
| 32 | 2011 | 17 | Hugo Claus | France | B | M | 2 | 1:01:15 | 19.59 |
| 32 | 2011 | 105 | Ivo Michiels | 9000 | F | M | 2 | 1:01:15 | 19.59 |
| 32 | 2011 | 39 | Jef Geeraerts | 9000 | B | M | 4 | 1:01:19 | 19.57 |
| 32 | 2011 | 99 | Jules Verne | France | F | M | 934 | 1:30:00 | 13.33 |
| 32 | 2011 | 123 | Caroline Smets | 2100 | B | F | 200 | 1:30:00 | 13.33 |
| 32 | 2011 | 1957 | Anne Dua | 3700 | B | F | 722 | 1:45:00 | ~~12.63~~ → 11.43 |
| ⋮ | | | | | | | | | |

Quelles sont les dépendances fonctionnelles pour cette table ?

151

.../10

Edition → Année

Année → Edition

Edition, Dossard → Nom, CP, Nat, Sexe, Temps

(les dossards servent à identifier les athlètes dans la course)

Edition, Nom, CP → Dossard

Edition, Sexe, Rang → Temps

Edition, Sexe, Temps → Rang

Temps → Moyenne

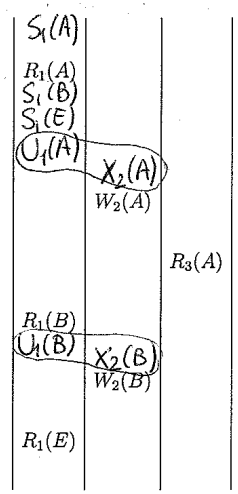
(Je préfère ne pas mettre Moyenne → Temps, parce que 1:45:00 et 1:45:01 donnent la même valeur pour moyenne).

**Question 9** Considérez l'exécution suivante : → NON

$$R_1(A)W_2(A)R_3(A)R_1(B)W_2(B)R_1(E)$$

Est-ce que cette exécution est possible en 2PL ? Complétez l'exécution avec des demandes de verrous ou argumentez pourquoi cette exécution n'est pas possible en 2PL.

.../5




On n'a pas beaucoup de liberté pour mettre les verrous :

$U_2(A)$ doit se faire après $X_2(B)$.

Alors $T_3$ effectue $R_3(A)$ au moment où $T_2$ possède $X_2(A)$, ce qui est IMPOSSIBLE en 2PL.

**Question 10** Cochez la case qui précède une expression correcte :

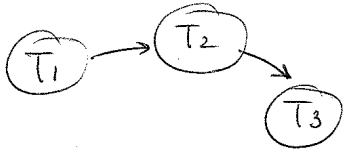
☒ L'exécution de la question 9 est sérialisable.

☐ L'exécution de la question 9 n'est pas sérialisable.

Argumentez votre réponse.

.../5

Le graphe de précédence ne contient pas de cycle :






**Question 11** La société *Rainparcs* exploite plusieurs villages de vacances. Ces villages s'appellent Borinage, Ardennes, Côte, etc. Chaque village de vacances contient plusieurs bungalows de vacances. Chaque bungalow est d'un certain type (hyacinthe, rose, tulipe, etc.). La première ligne de la table TYPE indique que les bungalows de type "hyacinthe" peuvent accueillir 6 personnes, contiennent une télévision mais pas de lave-vaisselle. La première ligne de la table VILLAGES indique que le village "Borinage" contient 12 bungalows de type "hyacinthe". Les bungalows peuvent être loués pour des périodes d'une semaine commençant le lundi. Les premières deux lignes de la table LOCATIONS indiquent que le client 111 a loué 3 bungalows de type "tulipe" pour les semaines commençant le 1 août 2011 et le 8 août 2011. Un même client ne peut pas louer deux bungalows dans deux villages différents pour la même période (mais il peut louer plusieurs bungalows dans un même village). Chaque client est identifié par un numéro unique. La table CLIENT donne les noms et adresses des clients.

TYPE

| Type | Personnes | Télévision | LaveVaisselle |
|---|---|---|---|
| hyacinthe | 6 | 1 | 0 |
| rose | 6 | 1 | 1 |
| dahlia | 6 | 0 | 1 |
| tulipe | 2 | 1 | 1 |
| tournesol | 4 | 0 | 0 |

VILLAGES

| Village | Type | Nombre |
|---|---|---|
| Borinage | hyacinthe | 12 |
| Borinage | tulipe | 15 |
| Borinage | tournesol | 35 |
| Ardennes | rose | 27 |
| Ardennes | tulipe | 5 |
| Ardennes | dahlia | 5 |
| Côte | tournesol | 17 |
| Côte | rose | 23 |

LOCATIONS

| Client | Village | Nombre | Type | Semaine |
|---|---|---|---|---|
| 111 | Borinage | 3 | tulipe | 1 août 2011 |
| 111 | Borinage | 3 | tulipe | 8 août 2011 |
| 111 | Borinage | 2 | tournesol | 1 août 2011 |
| 111 | Ardennes | 1 | tulipe | 15 août 2011 |
| 222 | Ardennes | 2 | rose | 15 août 2011 |
| 333 | Borinage | 1 | hyacinthe | 1 août 2011 |
| 444 | Borinage | 1 | hyacinthe | 22 août 2011 |

CLIENTS

| Client | Nom | Rue | CodePostal | Ville | Pays |
|---|---|---|---|---|---|
| 111 | M. Ed Green | 5 Av. de Binche | 7000 | Mons | Belgique |
| 222 | Mme Anne Dupont | 10 Champ de Mars | 7000 | Mons | Belgique |
| 333 | Mme Alice Dufour | 20 Av. d'Anvers | 1000 | Bruxelles | Belgique |
| 444 | M. Jean Pinto | 20 Av. de Charleroi | 7000 | Mons | Belgique |

152

Pour ces quatre tables, donnez toutes les contraintes de type PRIMARY KEY, FOREIGN KEY et UNIQUE.

.../5

TYPE PK ( Type )

VILLAGES PK ( Village, Type )
FK ( Type ) REFS TYPE

LOCATIONS PK ( Client, Type, Semaine )
FK ( Client ) REFS CLIENTS
FK ( Village, Type ) REFS VILLAGES

CLIENTS PK ( Client )

**Question 12** Écrivez une requête en **calcul relationnel** pour répondre à la question suivante :

Donnez les villages où chaque type de bungalow est présent.

Pour l'exemple, la réponse est vide : Borinage n'a pas de bungalow de type *rose*, Ardennes n'a pas de bungalow de type *hyacinthe*, Côte n'a pas de bungalow de type *hyacinthe*.

.../5

$$\{ v \mid (\exists t \exists n \ \text{VILLAGES}(v,t,n)) \wedge (\forall y \forall u_1 \forall u_2 \forall u_3 \ \text{TYPE}(y,u_1,u_2,u_3) \rightarrow \exists t' \exists m' \ \text{VILLAGES}(v,t',m')) \}$$

**Question 13** Écrivez une requête en **algèbre relationnelle** pour répondre à la question suivante :

Donnez le nom de chaque personne qui n'a jamais loué un bungalow de type *hyacinthe*.

Pour l'exemple, la réponse contient *M. Ed Green* et *Mme Anne Dupont*.

.../5

Soit $R := \pi_{\text{Client}} ( \sigma_{\text{Type="hyacinthe"}} \text{LOCATIONS} )$

R donne les clients ayant loué "une hyacinthe"

Soit $S := (\pi_{\text{Client}} \text{CLIENTS}) - R$,

les autres clients.

La requête demandée :

$$\pi_{\text{Nom}} ( \text{CLIENTS} \bowtie S )$$



**Question 14** Écrivez **une seule** requête en **SQL** pour répondre à la question suivante :

Donnez le village avec le plus grand nombre de bungalows de type *tulipe*.

Pour l'exemple, c'est le village *Borinage* avec 15 bungalows de type *tulipe*.

.../5

```
SELECT v.Village
FROM VILLAGES AS v
WHERE v.Type = "tulipe"
AND NOT EXISTS
    ( SELECT *
      FROM VILLAGES AS w
      WHERE w.Type = "tulipe"
        AND w.Nombre > v.Nombre)
```

153

**Bases de Données I (J. Wijsen)**

18 janvier 2013

NOM + PRENOM :

Orientation + Année :

Cet examen contient 14 questions. Durée : 2 heures et 30 minutes.

**Question 1** Soit $R$ un nom de relation avec schéma $ABC$. Considérez la requête $q_1$ :

$$q_1 = \{x, y \mid \exists z\Big(R(x, y, z) \wedge \neg\exists z(R(z, y, x))\Big)\}$$

Donnez une requête en algèbre relationnelle qui est équivalente à $q_1$. Détaillez votre réponse.

../5

Noter que la sous-formule $\Big(R(x, y, z) \wedge \neg\exists z(R(z, y, x))\Big)$ contient une occurrence libre de $z$ et une occurrence liée de $z$. La requête $q_2$ ci-dessus est équivalente à $q_1$ :

$$q_2 = \{x, y \mid \exists z(R(x, y, z)) \wedge \neg\exists u(R(u, y, x))\}.$$

Cette requête se traduit en

$$\pi_{AB}(R) - \rho_{C\to A}(\pi_{BC}(R)).$$

154

**Question 2** Considérez la requête :

$$\{x, y \mid \exists u \exists w (R(x, y, u) \vee R(w, x, y))\}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Cette requête est *domain independent*.
- ☐ Cette requête n'est pas *domain independent*.

Expliquez en détail.

../2

La requête est équivalente à

$$\{x, y \mid \exists u(R(x, y, u)) \vee \exists w(R(w, x, y))\}.$$

Puisque les sous-formules $\exists u(R(x, y, u))$ et $\exists w(R(w, x, y))$ ont les mêmes variables libres, la requête est *safe* et donc *domain-independent*.

**Question 3** Considérez la requête :

$$\{y \mid \neg \exists x (S(x, y) \wedge \neg R(y))\}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Cette requête est *domain independent*.
- ☐ Cette requête n'est pas *domain independent*.

Expliquez en détail.

../3

La requête est équivalente à

$$\{y \mid \neg \exists x(S(x, y)) \vee R(y)\}.$$

Soit $a$ une constante qui n'appartient pas au domaine actif. Puisque $\neg \exists x(S(x, a))$ est Vrai, on a que $\neg \exists x(S(x, a)) \vee R(a)$ est Vrai. La réponse contient donc chaque constante qui n'appartient pas au domaine actif. En conséquence, cette requête n'est pas *domain independent*.



**Question 4** Détaillez le protocole Wait-Die et argumentez pourquoi ce protocole évite les verrous mortels.

../5

Pour gérer la concurrence, le protocole 2PL introduit des verrous. Si une transaction $T_i$ fait une demande $S_i(A)$ ou $X_i(A)$, elle risque d'être mise "en attente", ce qui peut engendrer des verrous mortels. Un verrou mortel s'est produit si le WAIT-FOR graphe contient un cycle.
Supposons que l'indice (ou l'estampille) $i$ dans $T_i$ dénote le temps de démarrage de la transaction. On peut donc dire que $T_i$ est plus âgée que $T_j$ si $i < j$. L'objectif du protocole Wait-Die est de garantir que pour toute arête $T_i \to T_j$ dans le WAIT-FOR graphe, $T_i$ est plus âgée que $T_j$, i.e., $i < j$. Cela évitera les verrous mortels, car au long de chaque chemin dans le WAIT-FOR graphe, les indices ne feront qu'augmenter. Un chemin qui part de $T_i$ ne peut donc pas revenir en $T_i$.
Le protocole Wait-Die stipule alors qu'une transaction $T_i$ est annulée si elle fait une demande de verrou qui conduirait à une situation où $T_i$ devrait attendre (le relâchement d'un verrou par) une transaction $T_j$ plus âgée (i.e., $i > j$).
Si une transaction $T_i$ annulée est redémarrée, elle gardera l'estampille $i$. De cette façon, elle "veillit" et est sûre d'aboutir un moment.
Le désavantage de ce protocole est le grand nombre d'annulations de transactions.

155

**Question 5** Soit $\mathcal{A}$ l'algèbre qui contient tous les opérateurs de SPJRUD **sauf la sélection** $\sigma_{A=B}$. Noter que la sélection $\sigma_{A=\text{“}c\text{”}}$ fait partie de $\mathcal{A}$. Soit $R$ une relation avec un seul attribut $A$ et soit $q_0$ la requête SPJRUD $\sigma_{A=B}(R \bowtie \rho_{A\to B}R)$. Prouvez que l'algèbre $\mathcal{A}$ ne contient aucune requête qui est équivalente à $q_0$.

*Suggestion :* Il semble vrai que pour chaque requête $q_1$ en $\mathcal{A}$ qui rend une relation avec schéma $AB$, il existe deux constantes $c, d$ ($c \neq d$) telles que pour la relation $R = \{\{A : c\}, \{A : d\}\}$, on a que $\{A : c, B : c\} \in q_1(R)$ implique $\{A : c, B : d\} \in q_1(R)$.

…/5

On regarde d'abord si ce qu'il "semble vrai" est effectivement vrai…
Soit $q$ une requête quelconque dans $\mathcal{A}$. Il suffit de prouver que $q$ n'est pas équivalente à $\sigma_{A=B}(R \bowtie \rho_{A\to B}R)$. Puisque $q$ est de taille finie, on peut toujours trouver deux constantes qui n'apparaissent pas dans $q$. Soient 0, 1 deux constantes qui n'apparaissent pas dans $q$. Disons qu'une relation avec $n$ atrributs est *complète* si elle contient toutes les $2^n$ combinaisons de 0 et 1. Les relations suivantes sont donc complètes.

| $A_1$ |
|---|
| 0 |
| 1 |

| $A_1$ | $A_2$ |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

| $A_1$ | $A_2$ | $A_3$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

…

Soit $R$ la relation $\begin{array}{|c|}\hline A \\ \hline 0 \\ 1 \\ \hline \end{array}$. On démontre par induction sur la syntaxe de $q$ que la relation $q(R)$ (i.e., le résultat d'appliquer $q$ sur $R$) est soit vide soit complète. C'est évidemment le cas pour la base de l'induction où $q$ est simplement $R$, i.e., $q(R) = R$. Autrement, $q$ est d'une des six formes ci-dessus. L'hypothèse d'induction est que chacune des relations $q_0(R)$ et $q_1(R)$ est soit vide soit complète.

**Cas où $q$ est de la forme $\sigma_{A=\text{“}c\text{”}}(q_0)$.** Puisque 0, 1 n'apparaissent pas dans $q$, on sait $0 \neq c \neq 1$. Évidemment, $q(R)$ est vide.

**Cas où $q$ est de la forme $\pi_X(q_0)$.** Il est facile à vérifier que la projection d'une relation complète est complète. Il est trivial que la projection d'une relation vide est vide. Puisque la relation $q_0(R)$ est vide ou complète par l'hypothèse d'induction, on aura que $q(R)$ est vide ou complète.

**Cas où $q$ est de la forme $\rho_{A\to B}(q_0)$.** Ce cas est facile à traiter.

**Cas où $q$ est de la forme $q_0 \bowtie q_1$.** Il est facile à vérifier que la jointure de deux relations complètes rend une relation complète. Il est trivial qu'une jointure est vide si un des deux arguments est une relation vide. Puisque chacune des relations $q_0(R)$ et $q_1(R)$ est vide ou complète par l'hypothèse d'induction, on aura que $q(R)$ est vide ou complète.

**Cas où $q$ est de la forme $q_0 \cup q_1$.** Ce cas est facile à traiter : $q(R)$ sera vide si $q_0(R)$ et $q_1(R)$ sont toutes les deux vides ; autrement $q(R)$ sera complète.

**Cas où $q$ est de la forme $q_0 - q_1$.** Ce cas est facile à traiter : $q(R)$ sera complète si $q_0(R)$ est complète et $q_1(R)$ est vide ; autrement $q(R)$ sera vide.

On peut donc conclure que la relation $q(R)$ est soit vide soit complète.

L'expression $\sigma_{A=B}(R \bowtie \rho_{A\to B}R)$ rend la relation $\begin{array}{|cc|}\hline A & B \\ \hline 0 & 0 \\ 1 & 1 \\ \hline \end{array}$, qui est ni vide ni complète. Cette relation n'est donc pas équivalente à $q$.



**Question 6** Considérez le schéma avec attributs $ABCDEF$ et les DF suivantes :

$$\begin{array}{llll} CD \to B & EF \to A & ABE \to FC & AE \to F \\ A \to E & C \to D & AB \to C & \end{array}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Ce schéma est en BCNF.
- ☐ Ce schéma n'est pas en BCNF.
- ☐ Ce schéma est en 3NF.
- ☐ Ce schéma n'est pas en 3NF.

Détaillez les calculs qui mènent à cette conclusion.

…/10

D'abord, on peut simplifier l'ensemble des DF comme suit :

$$\begin{array}{lll} C \to B & EF \to A & A \to F \\ A \to E & C \to D & AB \to C \end{array}$$

Il y a deux façons de démontrer que $D$ ne fait partie d'aucune clé.

**Démonstration par raisonnement.** Supposons par contradiction que $K$ est une clé qui contient $D$. On sait que $K$ détermine tous les attributs. Cependant, puisque $D$ n'apparaît pas à gauche d'une flêche, il faut que $K \setminus \{D\}$ détermine tous les attributs, ce qui contredit le fait que $K$ est une clé.

**Démonstration par calcul brut.** Par calcul brut, on trouve que l'ensemble des clés est $\{AB, AC, BEF, CEF\}$.

$C$ détermine $BCD$. Pour la DF $C \to D$, on a que $C$ ne détermine pas tous les attributs et $D$ ne fait partie d'aucune clé. Le schéma n'est donc pas en 3NF.
Puisque chaque schéma en BCNF est en 3NF, le schéma n'est pas en BCNF.

156

**Question 7** Soit $U$ un ensemble d'attributs et $\Sigma$ un ensemble de dépendances fonctionnelles sur $U$. Argumentez qu'il existe une décomposition de $(U, \Sigma)$ en BCNF qui préserve le contenu.

.../5

Soit $\Sigma^+ = \{X \to A \mid X \subseteq U \text{ et } A \in U \setminus X \text{ et } \Sigma \models X \to A\}$, l'ensemble de toutes les DF singulières qui sont une conséquence logique de $\Sigma$.
Supposons que $(U, \Sigma)$ n'est pas en BCNF. Alors, $\Sigma^+$ contient une DF $Y \to B$ telle que $\Sigma \not\models Y \to U$. On décomposera $(U, \Sigma)$ en deux schémas $(U_1, \Sigma_1)$ et $(U_2, \Sigma_2)$ où $U_1 = YB$, $U_2 = U \setminus \{B\}$, $\Sigma_1 = \{X \to A \in \Sigma^+ \mid XA \subseteq U_1\}$ et $\Sigma_2 = \{X \to A \in \Sigma^+ \mid XA \subseteq U_2\}$. Le théorème de Heath nous garantit que cette décomposition est sans perte de contenu, i.e., pour chaque relation $R$ sur $U$ qui satisfait $\Sigma$, on a $R = \pi_{U_1}(R) \bowtie \pi_{U_2}(R)$.
Il est évident que $Y \to B \in \Sigma_1$ et $Y \to B \notin \Sigma_2$. Le point essentiel est de remarquer que la DF $Y \to B$ ne cause plus de violation de BCNF, parce que $\Sigma_1 \models Y \to U_1$.
Si $(U_1, \Sigma_1)$ ou $(U_2, \Sigma_2)$ n'est pas en BCNF, on applique une même décomposition. On continue à décomposer jusqu'au moment où chaque composant est en BCNF. Cette procédure se terminera, car $\Sigma^+$ est fini et chaque décomposition enlève une DF qui cause une violation de BCNF.
Supposons que la procédure se terminera avec les schémas $(U'_1, \Sigma'_1)$, $(U'_2, \Sigma'_2)$, ..., $(U'_n, \Sigma'_n)$, chacun en BCNF. Puisque chaque décomposition était sans perte de contenu, il sera vrai (pourquoi exactement ?) que pour chaque relation $R$ sur $U$ qui satisfait $\Sigma$, on aura $R = \pi_{U'_1}(R) \bowtie \pi_{U'_2}(R) \ldots \bowtie \pi_{U'_n}(R)$.



**Question 8** Le *Tour des Flandres* et *Liège-Bastogne-Liège* sont deux courses cyclistes organisées annuellement au printemps. La table suivante sert à stocker les résultats de ces deux courses. La première ligne, par exemple, indique que le 1er avril 2012, Philippe Gilbert a terminé 75ème dans le Tour des Flandres. Il a disputé cette course avec le dossard numéro 11. Un coureur peut changer d'équipe d'une année à l'autre, mais pas pendant les quelques semaines qui séparent le *Tour des Flandres* et *Liège-Bastogne-Liège*. Par exemple, Philippe Gilbert faisait partie de l'équipe OMEGA en 2011, et de BMC en 2012. Quelles sont les dépendances fonctionnelles (DF) que l'on peut raisonablement imposer sur cette table ? Pour chaque DF, **exprimez la signification en français**. Par example,

`Coureur, Année → Équipe` signifie qu'un coureur ne change pas d'équipe au cours d'une même année.

| Course | Coureur | Équipe | Année | Mois | Jour | Classement | Dossard |
|---|---|---|---|---|---|---|---|
| Tour des Flandres | GILBERT Philippe | BMC | 2012 | avril | 1 | 75 | 11 |
| Liège-Bastogne-Liège | GILBERT Philippe | BMC | 2012 | avril | 22 | 16 | 1 |
| Liège-Bastogne-Liège | IGLINSKIY Maxim | ASTANA | 2012 | avril | 22 | 1 | 25 |
| Liège-Bastogne-Liège | GILBERT Philippe | OMEGA | 2011 | avril | 24 | 1 | 101 |
| Tour des Flandres | GILBERT Philippe | OMEGA | 2010 | avril | 4 | 3 | 93 |

.../10

`Coureur, Année → Équipe`
`Course, Année, Dossard → Coureur`
`Course, Année, Coureur → Classement`
`Course, Année, Classement → Dossard`
`Course, Année → Mois, Jour`
`Année, Mois, Jour → Course`

157

**Question 9** Considérez l'exécution suivante :

$$W_2(A)R_1(A)R_3(B)W_2(B)R_1(B)$$

Est-ce que cette exécution est possible en 2PL ? Complétez l'exécution avec des demandes de verrous ou argumentez pourquoi cette exécution n'est pas possible en 2PL.

../5

| | | |
|---|---|---|
| | $X_2(A)$ | |
| | $W_2(A)$ | |
| | $X_2(B)$ | |
| | $U_2(A)$ | |
| $S_1(A)$ | | |
| $R_1(A)$ | | |
| | | $R_3(B)$ |
| | $W_2(B)$ | |
| $R_1(B)$ | | |

Le début (commandes en rouge) ne nous laisse peu de liberté. En tout cas, la transaction $T_2$ possèdera un verrou exclusif sur $B$ au moment où $T_3$ doit exécuter $R_3(B)$, ce qui est impossible en 2PL.

**Question 10** Cochez la case qui précède une expression correcte :

☐ L'exécution de la question 9 est sérialisable.

☐ L'exécution de la question 9 n'est pas sérialisable.

Argumentez votre réponse.

../5

Les arêtes dans le graphe de précédence sont $T_3 \to T_2$ et $T_2 \to T_1$. Puisque le graphe ne contient pas de cycle, l'exécution est sérialisable.



**Question 11** Les *Foulées montoises* est une série annuelle de plusieurs courses à pieds qui se disputent dans la région montoise. La table `COURSES` enregistre les villes, dates et distances des courses pour l'année 2012. La table `PARTICIPANTS` enregistre le nom, genre et année de naissance de chaque participant. Chaque participant est identifié par un numéro unique. La table `RESULTATS` enregistre les temps réalisés ; par exemple, la première ligne indique qu'Anne Dua a terminé le Challenge de la Citadelle en 38 minutes et 41 secondes. Il est possible qu'un coureur déclare forfait pour une ou plusieurs courses.

`PARTICIPANTS`

| Numéro | Nom | Genre | Naissance |
|---|---|---|---|
| 122 | Anne Dua | F | 1964 |
| 133 | Pierre Dupont | M | 1978 |
| 144 | Sven Nijs | M | 1978 |

`RESULTATS`

| Date | Coureur | Temps |
|---|---|---|
| 12/4 | 122 | 0:38:41 |
| 26/4 | 122 | 0:39:36 |
| 12/4 | 133 | 0:35:28 |
| 26/4 | 133 | 0:39:36 |
| 1/5 | 133 | 0:33:56 |
| 1/5 | 144 | 0:30:41 |

`COURSES`

| Ville | Jour | Distance | Nom |
|---|---|---|---|
| Mons | 12/4 | 10km | Challenge de la Citadelle |
| Casteau | 19/4 | 12km | Course du Ravel |
| Nimy | 26/4 | 10km | Mémorial Plisnier |
| Mons | 1/5 | 20km | Les 20km de Mons |

Pour ces trois tables, donnez toutes les contraintes de type PRIMARY KEY, FOREIGN KEY et UNIQUE.

../5

158

```
PARTICIPANTS
    PRIMARY KEY(Numéro)
RESULTATS
    PRIMARY KEY(Date, Coureur)
    FOREIGN KEY(Coureur) REFERENCES PARTICPANTS
    FOREIGN KEY(Date) REFERENCES COURSES
COURSES
    PRIMARY KEY(Jour)
    UNIQUE(Nom)
```

**Question 12** Écrivez une requête en **calcul relationnel** pour répondre à la question suivante :

Donnez les numéros des coureurs qui ont participé à toutes les courses.

Pour l'exemple, aucun coureur n'a participé à toutes les courses.

../5

$$\{x \mid \exists v_1 \exists v_2 \exists v_3 (PARTICIPANTS(x, v_1, v_2, v_3)) \wedge \forall y \forall v_4 \forall v_5 \forall v_6 \Big( COURSES(v_4, y, v_5, v_6) \rightarrow \exists v_7 RESULTATS(y, x, v_7) \Big) \}$$

**Question 13** Écrivez une requête en **algèbre relationnelle** pour répondre à la question suivante :

Donnez le nom de chaque coureur qui n'a participé à aucune course de 10km.

Pour l'exemple, la réponse contient *Sven Nijs*.

../5

Soit $R = \pi_{Coureur}(RESULTATS \bowtie \rho_{Jour \rightarrow Date}(\sigma_{Distance="10km"}(COURSES)))$.
$R$ contient $\{Coureur : x\}$ ssi $x$ est le numéro d'un coureur ayant participé à une course de 10km.

Soit $S = \pi_{Numero}(PARTICIPANTS) - \rho_{Coureur \rightarrow Numero}(R)$.
$S$ contient $\{Numero : x\}$ ssi $x$ est le numéro d'un coureur n'ayant participé à aucune course de 10km.

La requête demandée est :

$$\pi_{Nom}(PARTICIPANTS \bowtie S).$$



**Question 14** La table `PATIENTS` stocke les symptômes des patients. La table `MALADIES` stocke les symptômes des maladies.

`PATIENTS`

| P | Symptome |
|---|---|
| Anne | fièvre |
| Anne | maux de tête |
| Anne | perte d'appétit |
| Anne | constipation |
| Pierre | fièvre |
| Pierre | maux de tête |

`MALADIES`

| M | Symptome |
|---|---|
| grippe | fièvre |
| grippe | maux de tête |
| grippe | perte d'appétit |
| appendicite | constipation |
| appendicite | fièvre |
| appendicite | perte d'appétit |

Écrivez **une seule** requête en **SQL** pour répondre à la question suivante :

Donnez chaque paire $(p, m)$ telle que $p$ est le nom d'un patient qui a tous les symptômes de la maladie $m$.

Pour l'exemple, la réponse est composée des paires (Anne, grippe) et (Anne, appendicite).

**Explicitez la logique derrière votre requête SQL en montrant, par exemple, la requête en *tuple relational calculus* (TRC).**

../5

En *Tuple Relational Calculus*, en indiquant les attributs par leurs noms (au lieu de leurs positions) :

$$\{t.P, s.M \mid PATIENTS(t) \wedge MALADIES(s) \wedge \forall s' \in MALADIES \Big( s'.M = s.M \rightarrow \exists t' \in PATIENTS(t'.P = t.P \wedge t'.Symptome = s'.Symptome) \Big) \}$$

Ou encore :

$$\{t.P, s.M \mid PATIENTS(t) \wedge MALADIES(s) \wedge \neg \exists s' \in MALADIES \Big( s'.M = s.M \wedge \neg \exists t' \in PATIENTS(t'.P = t.P \wedge t'.Symptome = s'.Symptome) \Big) \}$$

159

En SQL :

```
SELECT t.P, s.M
FROM   PATIENTS AS t, MALADIES AS s
WHERE  NOT EXISTS ( SELECT *
                    FROM   MALADIES AS sprime
                    WHERE  sprime.M = s.M
                    AND    NOT EXISTS ( SELECT *
                                        FROM   PATIENTS AS tprime
                                        WHERE  tprime.P = t.P
                                        AND    tprime.Symptome = sprime.Symptome ))
```

**Bases de Données I (J. Wijsen)**

9 janvier 2014

NOM + PRÉNOM :

Orientation + Année :

Cet examen contient 10 questions. Durée : 2 heures et 30 minutes.

**Question 1** Considérez la requête

$$\{y \mid \exists x \exists z \Big( (R(x,y) \vee R(y,z)) \wedge \neg R(x,y) \Big) \}.$$

Cochez la case qui précède une expression correcte :

☐ Cette requête est *domain independent*.

☐ Cette requête n'est pas *domain independent*.

Expliquez en détail.

. . ./5



**Question 2** Soit SPJRU l'algèbre que l'on obtient on enlevant la Différence de l'algèbre SPJRUD. Soit SPJRU$^{\neq}$ l'algèbre qui étend l'algèbre SPJRU avec les sélections de forme $\sigma_{A \neq B}(R)$ et $\sigma_{A \neq c}(R)$, où $A, B$ sont des attributs et $c$ est une constante. La sémantique est évidente :

$$\begin{aligned} \sigma_{A \neq B}(R) &= \{t \in R \mid t(A) \neq t(B)\}; \\ \sigma_{A \neq c}(R) &= \{t \in R \mid t(A) \neq c\}. \end{aligned}$$

Noter que les algèbres SPJRU$^{\neq}$ et SPJRU ne contiennent pas la différence $R - S$ entre deux relations $R$ et $S$.

Démontrez que l'algèbre SPJRUD (avec la différence) est plus puissante que l'algèbre SPJRU$^{\neq}$.

. . ./7

160

**Question 3** La Ligue de diamant de l'IAAF est une compétition d'athlétisme organisée annuellement par l'Association internationale des fédérations d'athlétisme (IAAF). Elle est composée de 14 meetings dans 14 villes différentes. Toutes les épreuves sont individuelles. Les résultats sont stockées dans la table suivante. On apprend, par exemple, qu'en 2012, Kevin Borlée a gagné les 400m hommes à Bruxelles, alors qu'il était troisième à Lausanne. En 2011, Kevin était deuxième dans cette discipline à Bruxelles derrière son frère.

A chaque meeting, un athlète ne peut participer qu'à une seule discipline. Noter qu'en 2012, Usain Bolt a couru les 100m à Bruxelles et les 200m à Lausanne.

Un athlète est déterminé par son nom et prénom. Un athlète peut changer de pays d'une année à l'autre, mais il ne peut pas changer de pays pendant une même année civile. Un athlète ne change jamais de genre.

| *Ville* | *Date* | *Année* | *Nom* | *Prénom* | *Genre* | *Pays* | *Discipline* | *Résultat* | *Classement* |
|---|---|---|---|---|---|---|---|---|---|
| Bruxelles | 7 sep. | 2012 | Soumaré | Myriam | F | FRA | 200m | 22.63 | 1 |
| Bruxelles | 7 sep. | 2012 | Blake | Yohann | M | JAM | 200m | 19.54 | 1 |
| Bruxelles | 7 sep. | 2012 | Bolt | Usain | M | JAM | 100m | 9.86 | 1 |
| Bruxelles | 7 sep. | 2012 | Carter | Nesta | M | JAM | 100m | 9.96 | 2 |
| Bruxelles | 7 sep. | 2012 | Borlée | Kevin | M | BEL | 400m | 44.75 | 1 |
| Bruxelles | 7 sep. | 2012 | Borlée | Jonathan | M | BEL | 400m | 45.02 | 2 |
| Lausanne | 23 août | 2012 | Bolt | Usain | M | JAM | 200m | 19.58 | 1 |
| Lausanne | 23 août | 2012 | Borlée | Kevin | M | BEL | 400m | 45.27 | 3 |
| Bruxelles | 16 sep. | 2011 | Borlée | Jonathan | M | BEL | 400m | 44.78 | 1 |
| Bruxelles | 16 sep. | 2011 | Borlée | Kevin | M | BEL | 400m | 44.97 | 2 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Donnez les dépendances fonctionnelles que l'on peut raisonnablement imposer.

. . ./11



**Question 4** La table *BESTS* stocke la meilleure performance (*PB* = personal best) d'un(e) athlète dans une discipline. Chaque athlète est déterminé par son nom et prénom. La table *HOMMES* stocke les athlètes masculins ; la table *FEMMES* stocke les athlètes féminines. La colonne *Nat* donne la nationalité (dans cette question, on suppose que la nationalité d'un(e) athlète ne change pas). La table *RECORDS* stocke le record du monde dans chaque discipline. Pour chaque ligne dans la table *RECORDS*, on peut retrouver le nom du teneur (cq. de la teneuse) du record du monde en utilisant les tables *BESTS*, *HOMMES* et *FEMMES*. **Noter que le chiffre 18.29 intervient dans trois records :** dans le triple saut, le record du monde est de 18.29m pour les deux sexes, et 18.29s est le record du monde dans les 200m hommes.

*BESTS*

| *Nom* | *Prénom* | *Discipline* | *PB* |
|---|---|---|---|
| Bolt | Usain | 100m | 9.58 |
| Bolt | Usain | 200m | 18.29 |
| Borlée | Kevin | 400m | 44.56 |
| Borlée | Jonathan | 400m | 44.43 |
| Borlée | Jonathan | 200m | 20.31 |
| Griffith Joyner | Florence | 100m | 10.49 |
| Gevaert | Kim | 100m | 11.04 |
| Gevaert | Kim | 200m | 22.20 |
| Johnson | Michael | 400m | 43.18 |
| Edwards | Jonathan | Triple saut | 18.29 |
| Superwoman | Jessica | Triple saut | 18.29 |

*HOMMES*

| *Nom* | *Prénom* | *Nat* |
|---|---|---|
| Bolt | Usain | JAM |
| Borlée | Kevin | BEL |
| Borlée | Jonathan | BEL |
| Johnson | Michael | USA |
| Edwards | Jonathan | GB |

*RECORDS*

| *Discipline* | *Genre* | *Résultat* |
|---|---|---|
| 100m | M | 9.58 |
| 100m | F | 10.49 |
| 200m | M | 18.29 |
| 400m | M | 43.18 |
| Triple saut | M | 18.29 |
| Triple saut | F | 18.29 |

*FEMMES*

| *Nom* | *Prénom* | *Nat* |
|---|---|---|
| Griffith Joyner | Florence | USA |
| Gevaert | Kim | BEL |
| Superwoman | Jessica | JAP |

16/1

Donnez toutes les clés primaires et étrangères en utilisant la syntaxe du cours.

. . ./7

**Question 5** Pour la base de données de la question 4, écrivez une requête en algèbre SPJRUD qui donne pour chaque discipline la nationalité du teneur (de la teneuse) du record du monde, comme suit.

| *Discipline* | *Genre* | *Nat* |
|---|---|---|
| 100m | M | JAM |
| 100m | F | USA |
| 200m | M | JAM |
| 400m | M | USA |
| Triple saut | M | GB |
| Triple saut | F | JAP |

. . ./7



**Question 6** Pour la base de données de la question 4, écrivez une requête en calcul relationnel qui donne le nom de chaque homme qui ne possède aucun record du monde, comme suit.

| *Nom* | *Prénom* |
|---|---|
| Borlée | Kevin |
| Borlée | Jonathan |

. . ./7

**Question 7** Soient

$$\begin{aligned} U &= ABCDEFG \\ \Sigma &= \{ABC \to DEF, DE \to AB, EF \to BC, DF \to ACG, G \to C\} \end{aligned}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Le schéma $(U, \Sigma)$ est en BCNF.
- ☐ Le schéma $(U, \Sigma)$ n'est pas en BCNF.
- ☐ Le schéma $(U, \Sigma)$ est en 3NF.
- ☐ Le schéma $(U, \Sigma)$ n'est pas en 3NF.

Détaillez les calculs qui mènent à cette conclusion.

. . ./9



**Question 8** Les élèves d'une même classe sont évalués sur plusieurs branches. On dit qu'un élève $x$ *est dominé par* un élève $y$ si pour chaque branche, la note obtenue par $x$ est strictement inférieure à la note obtenue par $y$ dans cette même branche. Dans la table *Notes* suivante, les branches sont histoire, géographie et biologie. Bernard est dominé par Antoine. Christian n'est pas dominé par Antoine, car la note de Christian en géographie est supérieure à la note d'Antoine en géographie.

*Notes*

| *Élève* | *Branche* | *Note* |
|---|---|---|
| Antoine | histoire | 6 |
| Antoine | géographie | 6 |
| Antoine | biologie | 8 |
| Bernard | histoire | 5 |
| Bernard | géographie | 5 |
| Bernard | biologie | 7 |
| Christian | histoire | 4 |
| Christian | géographie | 9 |
| Christian | biologie | 4 |
| Didier | histoire | 3 |
| Didier | géographie | 3 |
| Didier | biologie | 3 |

Écrivez une requête SQL qui donne chaque paire $(x, y)$ telle qu'élève $x$ est dominé par $y$. Explicitez la logique derrière votre requête SQL en montrant la requête en *tuple relational calculus*. Pour l'exemple, le résultat est comme suit.

| *Élève* | *EstDominéPar* |
|---|---|
| Didier | Antoine |
| Didier | Bernard |
| Didier | Christian |
| Bernard | Antoine |

163

. . ./7

**Question 9** Détaillez le protocole Wound-Wait et argumentez pourquoi ce protocole évite les verrous mortels.

/5



**Question 10** Considérez l'exécution suivante :

$$R_1(A)R_3(B)W_2(A)R_1(B)R_3(A)R_1(C)W_1(C)W_3(C)$$

Est-ce que cette exécution est possible en 2PL ? Complétez l'exécution avec des demandes de verrous ou argumentez pourquoi cette exécution n'est pas possible en 2PL.

/7

| | | |
|---|---|---|
| $R_1(A)$ | | |
| | | $R_3(B)$ |
| | $W_2(A)$ | |
| $R_1(B)$ | | |
| | | $R_3(A)$ |
| $R_1(C)$ | | |
| $W_1(C)$ | | |
| | | $W_3(C)$ |

164

**Bases de Données I (J. Wijsen)**

13 janvier 2015

NOM + PRÉNOM :

Orientation + Année :

Cet examen contient 10 questions. Durée : 3 heures.

**Question 1** Considérez la requête

$$\{y \mid \exists x \exists z \Big( (R(x,y) \vee R(y,z)) \wedge \neg R(x,y) \Big) \}.$$

Cochez la case qui précède une expression correcte :

- ☐ Cette requête est *domain independent*.
- ☐ Cette requête n'est pas *domain independent*.

Expliquez en détail.

. . ./5



**Question 2** Soit SPJRU l'algèbre que l'on obtient on enlevant la Différence de l'algèbre SPJRUD. Démontrez que l'algèbre SPJRUD (avec la différence) est plus puissante que l'algèbre SPJRU.

. . ./5

**Question 3** On enregistre dans une table les prix journaliers de livres en ligne auprès de plusieurs vendeurs qui sont actifs sur le territoire européen. Chaque livre est identifié de façon unique par son *International Standard Book Number* (ISBN). La première ligne enregistre que le livre avec ISBN 0-486-43947-X coutait 20.00 EUR auprès d'Amazon à la date du 11 août 2014. Le prix d'un livre peut varier d'un jour au lendemain, mais ne change pas au cours d'une même journée. Évidemment, le prix d'un livre peut varier d'un vendeur à l'autre.

Le prix en Dollar (USD) est obtenu en multipliant le prix en Euro (EUR) par la parité EUR/USD en vigueur. Cette parité est fixée chaque jour par la Banque centrale européenne, mais peut varier d'un jour au lendemain. A partir de la première rangée de la table, on peut conclure qu'à la date du 11 août 2014, la parité EUR/USD était de $26.80/20.00 = 1.34$ (i.e., $1.00$ EUR valait $1.34$ USD). Le 12 août 2014, la parité était de $26.00/20.00 = 1.30$.

Un livre peut avoir un, deux, ou plusieurs auteurs. Par exemple, le livre de 2010 intitulé "Databases" est co-écrit par R. Elmasri et L. C. Grove. Deux livres différents peuvent avoir le même titre, mais l'agence officielle d'attribution des ISBN veille à ce qu'aucune personne ne soit auteur ou co-auteur de deux livres portant des ISBN différents mais ayant le même titre et année de publication (*PubYear*). Donc, les livres publiés dans une même année par une même personne auront tous des titres distincts.

Pour un livre à plusieurs auteurs, les auteurs doivent toujours être listés dans le même ordre, qui n'est pas forcément l'ordre alphabétique. L'attribut *Order* encode cet ordre : le premier auteur porte le chiffre 1, le deuxième auteur porte le chiffre 2… Si un livre n'a qu'un seul auteur, cet auteur est premier auteur.

Supposons que chaque auteur est identifié de façon unique par son nom.

| *ISBN* | *Title* | *Author* | *Order* | *PubYear* | *Vendor* | *Date* | *EUR* | *USD* |
|---|---|---|---|---|---|---|---|---|
| 0-486-43947-X | Algebra | L. C. Grove | 1 | 1983 | Amazon | 11 Aug 2014 | 20.00 | 26.80 |
| 0-486-43947-X | Algebra | L. C. Grove | 1 | 1983 | Proxis | 11 Aug 2014 | 21.00 | 28.14 |
| 0-486-43947-X | Algebra | L. C. Grove | 1 | 1983 | Amazon | 12 Aug 2014 | 20.00 | 26.00 |
| 0-486-43947-X | Algebra | L. C. Grove | 1 | 1983 | Proxis | 12 Aug 2014 | 20.00 | 26.00 |
| 978-0136086208 | Databases | R. Elmasri | 1 | 2010 | Amazon | 12 Aug 2014 | 80.00 | 104.00 |
| 978-0136086208 | Databases | L. C. Grove | 2 | 2010 | Amazon | 12 Aug 2014 | 80.00 | 104.00 |
| 978-0596809157 | Cookiebook | R. Elmasri | 1 | 2012 | Proxis | 12 Aug 2014 | 80.00 | 104.00 |

Quelles sont les dépendances fonctionnelles que l'on peut raisonnablement imposer sur ces données ?

../10



**Question 4** Les mêmes données de la question 3 sont maintenant stockées dans quatres tables $V$, $T$, $A$ et $E$. La table $V$ stocke les prix journaliers en EUR par vendeur. La table $T$ stocke les titres des livres, ainsi que leurs années de publication. La table $A$ stocke les auteurs des livres et indique l'ordre des auteurs pour des livres à plusieurs auteurs. Au lieu de stocker des prix en USD, la table $E$ stocke le cours de change entre EUR et USD.

$V$

| *ISBN* | *Vendor* | *Date* | *EUR* |
|---|---|---|---|
| 0-486-43947-X | Amazon | 11 Aug 2014 | 20.00 |
| 0-486-43947-X | Proxis | 11 Aug 2014 | 21.00 |
| 0-486-43947-X | Amazon | 12 Aug 2014 | 20.00 |
| 0-486-43947-X | Proxis | 12 Aug 2014 | 20.00 |
| 978-0136086208 | Amazon | 12 Aug 2014 | 80.00 |
| 978-0596809157 | Proxis | 12 Aug 2014 | 80.00 |

$E$

| *Date* | *EURvsUSD* |
|---|---|
| 11 Aug 2014 | 1.34 |
| 12 Aug 2014 | 1.30 |

$T$

| *ISBN* | *Title* | *PubYear* |
|---|---|---|
| 0-486-43947-X | Algebra | 1983 |
| 978-0136086208 | Databases | 2010 |
| 978-0596809157 | Cookiebook | 2012 |

$A$

| *ISBN* | *Author* | *Order* |
|---|---|---|
| 0-486-43947-X | L. C. Grove | 1 |
| 978-0136086208 | R. Elmasri | 1 |
| 978-0136086208 | L. C. Grove | 2 |
| 978-0596809157 | R. Elmasri | 1 |

Donnez toutes les clés primaires et étrangères en utilisant la syntaxe du cours.

../5

166

**Question 5** Pour la base de données de la question 4, écrivez une requête en algèbre SPJRUD pour répondre à la question suivante :

> Quel est le prix le plus élevé qui a jamais été demandé pour un livre écrit ou co-écrit par L. C. Grove ? Donnez aussi le titre du livre en question, le vendeur qui a demandé ce prix et la date à laquelle ce prix a été demandé.

À la date du 12 août 2014, Amazon a vendu le livre “Databases” de L. C. Grove au prix de 80 EUR. Aucun autre livre de L. C. Grove n’a jamais coûté plus cher. La réponse est donc :

| *Title* | *Vendor* | *Date* | *EUR* |
|---|---|---|---|
| Databases | Amazon | 12 Aug 2014 | 80.00 |

Pour cette question, on peut utiliser la sélection de type $\sigma_{A<B}(R)$, où $A$ et $B$ sont des colonnes de la relation $R$ qui contiennent des prix. La sémantique est comme attendue :

$$\sigma_{A<B}(R) \quad = \quad \{t \in R \mid t(A) < t(B)\}.$$

Expliquez le fonctionnement de votre requête aussi en français.

. . ./10



**Question 6** Pour la base de données de la question 4, écrivez une requête en calcul relationnel qui donne le nom de chaque auteur qui a été premier auteur de **tous** les livres dont il est auteur ou co-auteur. Pour la base de données de la question 4, la réponse contient R. Elmasri. Notez que L. C. Grove ne fait pas partie de la réponse, parce qu’il a aussi été deuxième auteur.

. . ./5

**Question 7** Soient

$$\begin{aligned} U &= ABCDEFG \\ \Sigma &= \{ABC \to DEF, DE \to AB, EF \to BC, DF \to ACG, G \to C\} \end{aligned}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Le schéma $(U, \Sigma)$ est en BCNF.
- ☐ Le schéma $(U, \Sigma)$ n'est pas en BCNF.
- ☐ Le schéma $(U, \Sigma)$ est en 3NF.
- ☐ Le schéma $(U, \Sigma)$ n'est pas en 3NF.

Détaillez les calculs qui mènent à cette conclusion.

../10



**Question 8** La table *Aime* enregistre qui aime quelle bière. On dit que la personne $x$ *est plus sélective que* la personne $y$ si $y$ aime toutes les bières que $x$ aime et, en plus, $y$ aime au moins une bière que $x$ n'aime pas. Par exemple, Maria est plus sélective que Pierre, car Pierre aime toutes les bières que Maria aime (notamment Leffe et Westmalle) et, en plus, Pierre aime une bière que Maria n'aime pas (notamment Orval).

*Aime*

| *Personne* | *Bière* |
|---|---|
| Pierre | Orval |
| Pierre | Leffe |
| Pierre | Westmalle |
| Jean | Orval |
| Jean | Leffe |
| Jean | Westmalle |
| Maria | Leffe |
| Maria | Westmalle |

Écrivez une requête SQL qui s'appuie sur `[NOT] EXISTS` pour rendre chaque personne $x$ qui est plus sélective que Pierre. Explicitez la logique derrière votre requête SQL en montrant la requête en *tuple relational calculus*. Pour l'exemple, le résultat est comme suit.

| *Personne* |
|---|
| Maria |

../10

168

**Question 9** Expliquez pourquoi, en pratique, les SGBD utilisent Stict 2PL au lieu de 2PL.

../5



**Question 10** Considérez l'exécution suivante :

$$R_1(A)R_3(B)W_2(A)R_1(B)R_3(A)R_1(C)W_1(C)W_3(C)$$

Est-ce que cette exécution est possible en 2PL ? Complétez l'exécution avec des demandes de verrous ou argumentez pourquoi cette exécution n'est pas possible en 2PL.

../5

| | | |
|---|---|---|
| $R_1(A)$ | | |
| | | $R_3(B)$ |
| | $W_2(A)$ | |
| $R_1(B)$ | | |
| | | $R_3(A)$ |
| $R_1(C)$ | | |
| $W_1(C)$ | | |
| | | $W_3(C)$ |

169