Université de Sherbrooke

Théorie et modèles relationnels

Fondements

TMR_02

Christina KHNAISSER *(christina.khnaisser@usherbrooke.ca)*

Luc LAVOIE *(luc.lavoie@usherbrooke.ca)*

(les auteurs sont cités en ordre alphabétique nominal)

Scriptorum/Scriptorum/TMR_02-Fondements, version 0.1.5.b, en date du 2025-01-09

— document de travail, ne pas citer —

Plan

Introduction

Le présent document a pour but de présenter une synthèse de la théorie relationnelle proposée par Edgar F. Codd et développée grâce, notamment, aux contributions de Christopher J. Date, Raymond F. Boyce, Nikos A. Lorentzos et Jeffrey D. Ullman.

La présentation repose sur des bases minimales :

- la logique du premier ordre,
- la théorie des ensembles (dont la théorie des types),
- l'arithmétique entière,
- les langages rationnels.

1. Présentation

1.1. Intuitions à l'origine de la théorie relationnelle

- Le point de départ : l'équivalence entre fait, proposition et tuple.
- Une généralisation : l'équivalence entre catégorie, prédicat et relation.
- Le couplage la théorie relationnelle à la théorie des types.
- L'automatisation du calcul.

1.2. Pourquoi le modèle relationnel ?

Lorsque Codd jette les bases de la théorie relationnelle en 1969 et propose un premier modèle opérationnalisable en 1970, il s'inscrit en continuité d'une histoire déjà longue dont voici quelques jalons choisis :

- Les fonctionnaires babyloniens (XVIII^{e} siècle avant notre ère).
- Ératosthène (III^{e} siècle avant notre ère) ;
- Les ingénieurs romains (I^{er} siècle) ;
- Cardan et Kepler (XVI^{e} siècle) ;

Ce n'est toutefois pas cette histoire qui permit à la théorie et au modèle de Codd de s'imposer rapidement à partir des anéées 1970 et de rester dominant jusqu'à nos jours, mais

- la capacité d'opérationnaliser des modèles de données tant simples que très complexes,
- la capacité de formuler la description d'un raisonnement, fondé sur la logique du premier ordre, dont le calcul est automatisable et
- la capacité d'évaluer objectivement plusieurs critères d'adéquation d'une solution en regard d'au problème.

Formalisation

- définissant l'équivalence des « faits » (proposition, variable, prédicat) ;
- définissant l'équivalence des opérateurs (relationnels et logiques) ;
- définissant les notions d'équivalence et d'égalité ;
- utilisant la notion d'ensemble ;
- utilisant des attributs dénotationnels plutôt que positionnels ;
- soutenant une algèbre complète au sens de Turing.

Adéquation

Des huit principaux critères d'adéquation (voir TMR_01), sept sont couverts de façon utile, voire complète, par la théorie relationnelle, ses bases et les théories informatiques complémentaires (telle que la théorie de la complexité). Le huitième critère, l'éthique (la conformité aux principes et règles de conduite d'une société humaine de référence), y échappe cependant.

Modèles et méta-modèles

On remarque que les modèles issus de la théorie relationnelle permettent la définition de modèles de données. Ce sont donc ce qu'il est convenu d'appeler des méta-modèles. De plus, ils ont souvent la particularité de pouvoir se décrire eux-mêmes ! Cette propriété incite donc la plupart des concepteurs de SBGB à y inclure une base de données, appelée catalogue, décrivant les bases de données gérées par le SGBD (dont le catalogue lui-même). Cette approche a l'avantage de permettre l'utilisation des opérateurs relationnels (et de tout l'environnement d'exploitation) lors de la gestion et de l'évolution des bases de données du SGBD.

2. Structure relationnelle

2.1. Définition

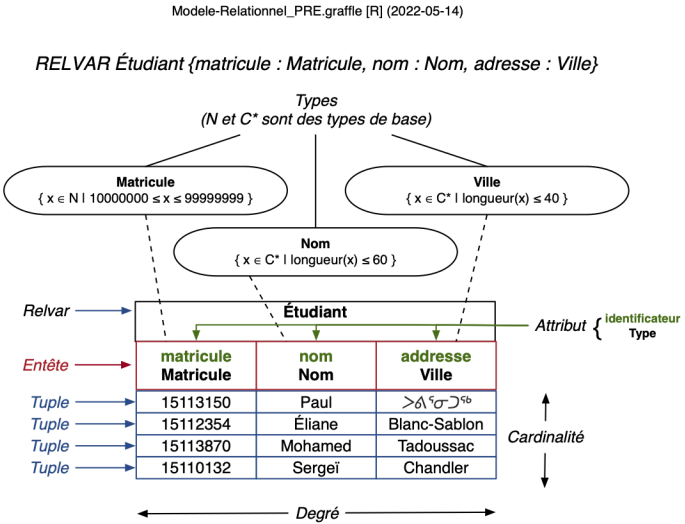
2.1.1. Définition informelle

La représentation usuelle, simple et pratique

Tableau 1. Étudiant

matricule	nom	ville
15113150	Paul	ᐳᕃᕀᓂᑲᖅ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Une représentation plus élaborée et plus précise



Toutefois, aucune de ces représentations ne permet de transformer les données en information puisqu'il y manque le prédicat de définition !

2.1.2. Définition formelle

La théorie relationnelle est construite sur la base de la logique du premier ordre et de la théorie des ensembles.

En pratique, il faut aussi ajouter un modèle de typage.

Finalement, il est également utile d'y ajouter l'arithmétique entière et les langages rationnels afin de rendre compte de certaines propriétés des nombres entiers et des textes considérés comme types primitifs avec les booléens.

2.2. Logique

Nous utiliserons les opérateurs suivants pour les expressions logiques (adaptation libre de https://fr.wikipedia.org/wiki/Portail:Logique, consulté les 2021-10-24 et 2024-04-07) :

Tableau 2. Opérateurs de la logique du premier ordre

Symbole	Spécification
$\neg A$	Négation de A. Non A.
$A \wedge B$	Conjonction. A et B.
$A \vee B$	Disjonction (inclusive). A ou B.
$A \Rightarrow B$	Implication. Si A alors B.
$A \Leftrightarrow B$	Équivalence. A est équivalent à B ; on dit aussi : A si et seulement si B.
$\Gamma \vdash A$	Déduction. De l'ensemble de formules Γ on déduit A.
$M \Vdash A$	Réalisabilité. M réalise A, on dit aussi que M « force » A.
$M \models A$	Modélisation. M est un modèle de A ; on dit aussi A est vraie dans M.
$\vdash A$	Théorème. Notion syntaxique.
$\models A$	Tautologie. Notion sémantique.

Les symboles $\equiv$ et $\equiv$ sont également utilisés pour dénoter l'équivalence $\Leftrightarrow$. Les nuances associées aux trois symboles ne sont traitées dans le présent document.

2.3. Ensemble

Dans le cadre de la théorie relationnelle, les éléments d'un ensemble sont toujours contraints par un type associé à l'ensemble lui-même. Ce type est désigné comme le « type de référence » de l'ensemble et noté *typeref(e)* où *e* est un ensemble.

Cette association peut être explicite (par voie de déclaration) ou déduite grâce au contexte.

2.3.1. Cas général

La structure des ensembles se veut volontairement aussi simple que possible. Ils ne possèdent donc qu'un seul constructeur et une seule propriété.

Le nombre d'opérateurs est toutefois considérable.

Nous n'en présenterons que quelques-uns.

Dénotation des valeurs et propriétés

{}

l'ensemble vide.

{ z }

l'ensemble comprenant une seule valeur dénotée par l'expression z (synonyme : singleton).

{ z_1, z_2, …, z_n }

l'ensemble comprenant les seules valeurs dénotées par les expressions z_i.

Le point-virgule « ; » peut être utilisé indifféremment en lieu et place de la virgule « , ».

 # e

la cardinalité de l'ensemble e, soit le nombre d'éléments qu'il contient..

Opérateurs usuels

$a = b$

égalité de a et b.

$a \neq b$

inégalité de a et b.

$z \in e$

l'appartenance d'un élément z à un ensemble e (aussi noté **$e \ni z$**).

$e_1 \subseteq e_2$

l'inclusion, e_1 est un sous-ensemble de e_2 (aussi noté **$e_2 \supseteq e_1$**).

$e_1 \subset e_2$

l'inclusion stricte, e_1 est un sous-ensemble strict de e_2 (aussi noté **$e_2 \supset e_1$**).

$e_1 - e_2$

la différence de e_2 moins e_1 (non commutatif).

$e_1 \cup e_2$

l'union de e_1 et e_2 (commutatif).

$e_1 \cap e_2$

l'intersection de e_1 et e_2 (commutatif).

$e_1 \times e_2$

le produit cartésien de e_1 par e_2 (non commutatif).

Inclusion de la négation (liste très partielle)

$z \notin e$

la non-appartenance d'un élément z à un ensemble e (aussi noté **$e \not\ni z$**).

$e_1 \not\subset e_2$

e_1 n'est pas un sous-ensemble strict de e_2 (aussi noté **$e_2 \not\supset e_1$**).

Raccourcis

Dans la mesure où tous les ensembles ont un type de référence, l'opérateur « \ » permet de calculer le complément d'un sous-ensemble par rapport à son type de référence.

\ e ≜ typeref(e) – e

où *e* est une expression dont la valeur est un ensemble typé.

Notes

- En Tutorial D, cet opérateur n'est utilisable que dans un contexte de dénotation explicite des valeurs, ainsi `{ALL BUT e1, …, en}` dénote $\{ e_1, \ldots, e_n \}$. On aurait souhaité, la syntaxe plus générale suivante `ALL BUT exp`, où `exp` dénote toute expression ensembliste typée.
- En Discipulus, la syntaxe générale est utilisée et les expressions d'ensemble (en particulier d'ensembles d'identifiants d'attributs) sont dénotables.

2.3.2. Couple

Le couple (paire) est la formation d'une nouvelle entité (ou élément) à partir de deux autres.

Il serait possible de reformuler le couple en termes d'ensemble, son utilisation au titre de construction propre relève donc uniquement de la commodité.

Dénotation des valeurs et propriétés

(x ; y)

pour x et y des expressions dénotant des valeurs appropriées.

Note

- La virgule « , » est souvent utilisée en lieu et place du point-virgule « ; » lorsqu'elle n'entraine pas d'ambigüité (e.a. en présence de nombres rationnels représentés en notation fractionnaire décimale).

Opérateurs usuels

@1(x ; y) = x

l'élément gauche du couple ;

@2(x ; y) = y

l'élément droit du couple.

2.4. Type

Si la théorie relationnelle peut s'accommoder de plusieurs modèles de typage, certaines exigences doivent être respectées.

- Malheureusement, il n'y a pas de consensus relativement à ces exigences (notamment, celles du comité ISO 9075 diffèrent de celles énoncées par Codd et développées par Date et Darwen).
- Nous nous en tiendrons donc ici à un modèle de typage très simple, également acceptable en regard des jeux d'exigences les plus courants.

Une donnée est une valeur associée à deux représentations :

- l'une interne, apte à être traitée par ordinateur ;
- l'autre externe, apte à être utilisée au sein d'un langage formel

Une représentation est une suite de signaux (un signal est un phénomène physique mesurable, donc suffisamment stable pour être mesuré).

Dans le cadre de la théorie minimaliste de typage utilisée dans le présent document, un type est soit un type de base, soit un sous-type.

- Un type de base (appelé aussi type racine ou domaine) est un ensemble fini de valeurs propres (c'est-à-dire que les valeurs d'un type de base n'appartiennent à aucun autre type de base). Une valeur est donc un élément d'un type de base.
- Un sous-type (appelé aussi type dérivé) est un sous-ensemble d'un type, sous-ensemble déterminé par une contrainte explicite (qui restreint les valeurs acceptées dans le sous-ensemble).

La finitude des types suscite encore de nombreux débats.

À telle enseigne que Date lui-même l'a retiré de sa définition depuis 2016, moins par conviction selon nous, mais plutôt pour éviter un débat peu productif.

N'ayant pas sa notoriété ni son influence, nous pouvons nous permettre de la maintenir, et de l'utiliser, dans le présent document.

S'il est vrai que la structure mathématique, et plus particulièrement algébrique du modèle relationnel, ne repose pas sur la finitude des types, son application pratique, elle, repose sur cette hypothèse — du moins, tant que nous ne bénéficierons pas de machine de calcul (ordinateur) dotée d'une mémoire infinie.

2.5. Attribut

Un attribut est un couple formé d'un identifiant a et d'un type D, noté a:D.

Par abus de langage, lorsque le contexte le permet, il est usuel de désigner l'attribut par son seul identifiant ; ainsi peut-on écrire « l'attribut a ».

C'est-à-dire que le contexte permet de déterminer de façon univoque et non ambigüe le type associé à l'identifiant au sein de l'attribut.

La mise en page des expressions formelles des trois prochaines sous-sections (tuple, relation, base) varie d'une sous-section à l'autre. C'est volontaire. Nous y expérimentons trois mécanismes différents mis à disposition par AsciiDoc pour ce faire.

Nous désirons recueillir ainsi les commentaires de nos lecteurs, afin de choisir ce qui convient le mieux. Vos commentaires sont donc bienvenus et attendus !

2.6. Tuple

Un tuple, en tant qu'objet de la théorie relationnelle, est une « composition » d'attributs.

En regard de la théorie des types, il s'agit d'un constructeur (de types non scalaires) applicable à un ensemble d'attributs.

Soit n un entier naturel,
soit n identifiants **distincts** dénotés $a_i \mid 1 \leq i \leq n$,
soit n types (pas nécessairement distincts) dénotés $D_i \mid 1 \leq i \leq n$,
soit n valeurs (pas nécessairement distinctes) dénotées $v_i \mid (a_i \in D_i) \wedge (1 \leq i \leq n)$
alors
un tuple t est défini comme suit :

$$t \triangleq (\{a_i{:}D_i \mid 1 \leq i \leq n\}; \{(a_i,v_i) \mid 1 \leq i \leq n\})$$

ou, de façon équivalente, en extension :

$$t \triangleq (\{a_1{:}D_1, \ldots, a_i{:}D_i, \ldots, a_n{:}D_n\}; \{(a_1,v_1), \ldots, (a_i,v_i), \ldots (a_n,v_n)\})$$

Sont également définis

$deg(t) = n$	degré de t
$id(t) = \{a_i \mid 1 \leq i \leq n\}$	ensemble des identifiants d'attributs de t
$def(t) = \{a_i{:}D_i \mid 1 \leq i \leq n\}$	entête de t
$val(t) = \{(a_i,v_i) \mid 1 \leq i \leq n\}$	valeur de t
$ANTE\ a \in id(t) : def(t, a) = D \mid a{:}D \in def(t)$	type de l'attribut a_i de t
$ANTE\ a \in id(t) : val(t, a) = v \mid (a,v) \in val(t)$	valeur de l'attribut a_i de t

Observation

Pourquoi autoriser $deg(t) = 0$?

- Le cas $deg(t) = 0$ correspond à un tuple remarquable $(\{\}; \{\})$ souvent désigné par l'identifiant t_0.

Exercice

Expliquer pourquoi ce tuple est remarquable… et essentiel !

Autres dénotations fréquentes

Parmi les plus fréquentes :

- dénotation des tuples fondée sur l'ordre d'énumération des attributs (les identifiants d'attributs et leurs types étant déterminés par ailleurs) :
  - $<v_1, v_2, \ldots, v_n>$
- dénotation des attributs au sein d'un tuple, [.Math]#val(t, a) :
  - *t.a*
  - *a(t)*
  - *t(a)*
  - *a FROM t*

Dénotation des constructeurs de types et de valeurs

Il est important, en pratique et du point de vue du génie logiciel, de différencier syntaxiquement le constructeur de types du constructeur de valeurs, ce que Tutorial D ne fait pas de façon assez explicite, nous semble-t-il.

Deux solutions sont généralement utilisées,

- des mots-clés différents, par exemple `TUPLE` et `TUP` ;
- un seul mot-clé et des parenthèses distinctives, par exemple `TUPLE{}` et `TUPLE[]`.

Discipulus a adopté cette dernière solution… pour le moment. Ainsi,

- le type tuple sans attribut est-il dénoté par `TUPLE {}` ;
- et la (seule) valeur (possible) d'un tuple sans attribut par `TUPLE []`.

2.7. Relation

Une relation, en tant qu'objet de la théorie relationnelle, est un ensemble de tuples.

En regard de la théorie des types, il s'agit d'un constructeur (de types non scalaires) applicable à un ensemble de tuples de même type.

Soit a_i des identifiants distincts,
soit D_j des types,
soit t_k des tuples alors
une relation R est définie comme suit :

$$R \triangleq (\{a_1{:}D_1, \ldots, a_i{:}D_i, \ldots, a_n{:}D_n\}; \{t_1, \ldots, t_k, \ldots, t_m\})$$

avec

$$\forall\ 1 \leq k \leq card(R) : def(R) = def(t_k)$$

Sont également définis

$deg(R) = n$	le degré de R
$card(R) = m$	la cardinalité de R
$id(R) = \{a_1, ..., a_i, ..., a_n\}$	l'ensemble des identifiants d'attributs de R
$val(R) = \{t_1, ..., t_k, ..., t_m\}$	la valeur de R
$def(R) = \{a_1{:}D_1, ..., a_i{:}D_i, ..., a_n{:}D_n\}$	l'entête de R
$def(R, a_i) = D_i$	le type de l'attribut a_i de R

Quatre tuples (ayant le même entête)

matricule: Matricule	nom: Nom	adresse: Ville	
matricule: 15113150	nom: Paul	adresse: >ᕕ(ᐛ)ᕗ	t1
matricule: 15112354	nom: Éliane	adresse: Blanc-Sablon	t2
matricule: 15113870	nom: Mohamed	adresse: Tadoussac	t3
matricule: 15110132	nom: Sergeï	adresse: Chandler	t4

Une relation comprenant quatre tuples

matricule: Matricule	nom: Nom	adresse: Ville	
matricule: 15113150	nom: Paul	adresse: >ᕕ(ᐛ)ᕗ	t1
matricule: 15112354	nom: Éliane	adresse: Blanc-Sablon	t2
matricule: 15113870	nom: Mohamed	adresse: Tadoussac	t3
matricule: 15110132	nom: Sergeï	adresse: Chandler	t4

La représentation compacte usuelle de cette même relation

matricule: Matricule	nom: Nom	adresse: Ville
15113150	Paul	>ᕕ(ᐛ)ᕗ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Prédicat et relation

Un prédicat peut être défini de deux façons et, corolairement, une relation aussi :

- par énumération (l'ensemble de tous les énoncés vrais et eux seuls) ;
- par compréhension (la caractérisation nécessaire et suffisante des relations entre les variables).

Observation

Pourquoi autoriser $deg(R) = 0$?

Le cas $deg(R) = 0$ correspond à deux (valeurs de) relations remarquables :

- $(\{\}; \{\})$
- $(\{\}; t_0)$

Elles sont très importantes, comme le zéro et le un pour les entiers !

Exercice

Expliquer pourquoi !

Dénotation des constructeurs de types et de valeurs

Il est important, en pratique et du point de vue du génie logiciel, de différencier syntaxiquement le constructeur de types du constructeur de valeurs, ce que Tutorial D ne fait pas de façon assez explicite, nous semble-t-il.

Attendu la solution retenue par Discipulus (voir section précédente), les (deux seules) valeurs de relations sans attributs sont représentées par

- `RELATION []` et
- `RELATION [TUPLE []]`.

Cela ne rappelle-t-il pas quelque chose aux Bourbakistes parmi vous ?

2.8. Base

Une base, en tant qu'objet de la théorie relationnelle, est une « composition » de relations.

En regard de la théorie des types, il s'agit d'un constructeur (de types non scalaires) applicable à un ensemble de relations.

Soit v_i des identifiants distincts,
soit D_i des types de relation,
soit r_i des (valeurs de) relations alors
une base (banque) B est définie comme suit :

$$B \triangleq (\{v_1{:}D_1, \ldots, v_i{:}D_i, \ldots, v_n{:}D_n\}; \{r_1, \ldots, r_i, \ldots, r_n\})$$

avec

$$\forall\ 1 \leq i \leq card(B) : def(B, v_i) = def(r_i)$$

Sont également définis

$deg(B) = n$	degré de B
$id(B) = \{v_1, \ldots, v_i, \ldots, v_n\}$	ensemble des identifiants de B
$val(B) = \{r_1, \ldots, r_i, \ldots, r_n\}$	valeur de B
$def(B) = \{v_1{:}D_1, \ldots, v_i{:}D_i, \ldots, v_n{:}D_n\}$	entête de B
$def(B, a_i) = D_i$	type de a_i de B

3. Opérateurs relationnels

L'algèbre relationnelle est souvent présentée à l'aide de six opérateurs relationnels primaires :

- **renommage**, $R\ \rho\ a{:}b$, la relation comprenant tous les tuples formés à partir d'un tuple de R dont l'attribut de nom a est remplacé par un attribut de nom b de même valeur, et rien d'autre ;
- **restriction**, $R\ \sigma\ c$, la relation comprenant tous les tuples de R satisfaisant la condition c, et rien d'autre ;
- **projection**, $R\ \pi\ x$, la relation comprenant tous les tuples formés à partir d'un tuple de R dont seuls les attributs dont le nom est parmi x ont été conservés, et rien d'autre ;

- **jointure**, $R \bowtie S$, la relation comprenant tous les tuples formés des attributs d'un tuple de R et de ceux d'un tuple de S dont les attributs de même nom sont de même valeur, et rien d'autre ;
- **union**, $R \cup S$, la relation comprenant tous les tuples de R et tous les tuples de S, et rien d'autre ;
- **différence**, $R - S$, la relation comprenant tous les tuples de R qui ne sont pas dans S, et rien d'autre.

3.1. Opérateurs primaires

Définition relativement à la théorie des ensembles et à la représentation précédente (au modèle, à la structure).

3.1.1. Renommage

R ρ A:C

A	B
a1	b1
a2	b2
a3	b3

ρ A:C =

C	B
a1	b1
a2	b2
a3	b3

Soit R une relation,
soit a et b deux identifiants alors

$R \rho\ a{:}b \equiv$

`R RENAME {a AS b}` $\equiv$

`select` a_1`, …, a as b, …,` a_n `from R` $\equiv$

$ANTE\ a \in id(R) \bigwedge b \notin id(R) :$

$SOIT\ E := def(R,a)\ ;\ Z := def(R) - \{a{:}E\} \cup \{b{:}E\} :$

$(Z\ ;\ \{(Z\ ;\ val(t) - \{(a,val(t,a))\} \cup \{(b,val(t,a))\}) \mid t \in val(R)\})$

Corolaires

- $card(R) = card(R \rho\ a{:}b)$
- $\forall\ t \in val(R) : (Z ; val(t) - \{(a,val(t,a))\} \cup \{(b,val(t,a)) \in R \rho\ a{:}b$
- $a \notin id(R \rho\ a{:}b) \bigwedge b \in id(R \rho\ a{:}b)$
- $R = ((R \rho\ a{:}b) \rho\ b{:}a)$

3.1.2. Projection

$R \pi \{A, C\}$

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

Soit w une expression ayant pour valeur une liste d'identifiants d'attribut $\{w_0, \ldots, w_n\}$ alors

$R\ \pi\ w \equiv$

$R\ \{w_0, \ldots, w_n\} \equiv$

$\texttt{select}\ w_1, \ldots, w_n\ \texttt{from}\ R \equiv$

ANTE $w \subseteq id(R)$:

SOIT $Z := \{a{:}D \mid a \in id(R) \bigwedge a \in w\}$:

$(Z; \{(Z; \{(a,v) \mid (a,v) \in val(t) \bigwedge a \in w\}) \mid t \in val(R)\})$

L'opération de projection (applicable tant aux tuples qu'aux relations) n'est bien définie que si tous les identifiants d'attributs sont définis dans l'entête de la relation.

Note

- En Tutorial D et en SQL, la liste doit être explicite.

3.1.3. Jointure

$R \bowtie S$

A	B
a1	b1
a2	b1
a3	b3
a4	b4

$\bowtie$

B	C
b1	c1
b2	c2
b3	c3
b3	c4

=

A	B	C
a1	b1	c1
a2	b1	c1
a3	b3	c3
a3	b3	c4

Soit R et S deux relations alors

$R \bowtie S \equiv$

`R JOIN S` $\equiv$

`select * from R natural join S` $\equiv$

$$SOIT\ X := id(R) \cap id(S) :$$
$$ANTE\ \forall\ a \in X : a{:}D \in def(R) \bigwedge a{:}D \in def(S) :$$
$$SOIT\ Z := def(R) \cup def(S) :$$
$$(Z ; \{(Z ; val(t1) \cup val(t2)) \mid$$
$$t1 \in val(R) \bigwedge t2 \in val(S) \bigwedge (\forall\ a \in X : (a,v) \in val(t1) \bigwedge (a,v) \in val(t2))\})$$

3.1.4. Restriction

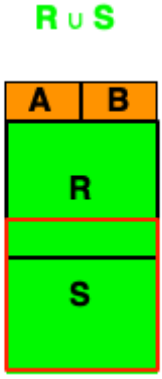
R σ cond

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

Soit R une relation,
soit c, une condition (expression booléenne),
soit $id(c)$, l'ensemble des identifiants d'attribut utilisés par c alors
R σ c $\equiv$
`R WHERE c` $\equiv$
`select * from R where c` $\equiv$
ANTE $id(c) \subseteq id(R)$:
$(def(R); \{t \mid t \in val(R) \bigwedge c(t)\})$

L'opération de restriction n'est bien définie que si tous les identifiants d'attributs de la condition sont définis dans l'entête de la relation.

3.1.5. Union



Soit R et S deux relations alors

$R \cup S \equiv$

`R UNION S` $\equiv$

```
select * from R union select * from S
```
$\equiv$

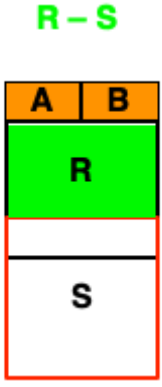
ANTE $def(R) = def(S)$:

$(def(R); val(R) \cup val(S))$

Note

- L'antécédent doit être aménagé en cas de relaxation de la compatibilité.

3.1.6. Différence



Soit R et S deux relations alors

$R - S \equiv$

`R MINUS S` $\equiv$

```
select * from R except select * from S
```
$\equiv$

ANTE $def(R) = def(S)$:

$(def(R); val(R) - val(S))$

Note

- L'antécédent doit être aménagé en cas de relaxation de la compatibilité.

3.1.7. Synthèse des opérateurs relationnels primaires

Restriction

$R \sigma cond$

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

Projection

$R \pi \{A, C\}$

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

Jointure

$R \bowtie S$

A	B
a1	b1
a2	b1
a3	b3
a4	b4

$\bowtie$

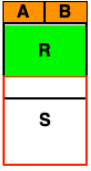
B	C
b1	c1
b2	c2
b3	c3
b3	c4

=

A	B	C
a1	b1	c1
a2	b1	c1
a3	b3	c3
a3	b3	c4

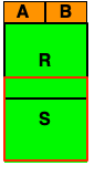
Différence

$R - S$



Union

$R \cup S$



Renommage

$R \rho A:C$

A	B
a1	b1
a2	b2
a3	b3

ρ A:C =

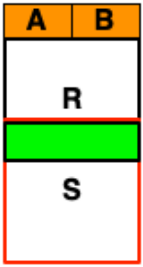
C	B
a1	b1
a2	b2
a3	b3

3.2. Opérateurs usuels

Afin d'augmenter l'expressivité de l'algèbre relationnelle, il est possible de définir d'autres opérateurs en terme des opérateurs relationnels primaires. En voici quelques-uns.

3.2.1. Intersection

$$R \cap S = R - (R-S) = R \bowtie S$$



Soit R et S deux relations :

$R \cap S \equiv$

```
R INTERSECT S ≡
```

```
select * from R intersect select * from S ≡
```

ANTE def(R) = def(S) :

$R \bowtie S$

Notes

- L'antécédent garantit que $R \bowtie S = R$ - $(R$ - $S)$.
- L'antécédent doit être aménagé en cas de relaxation de la compatibilité.

3.2.2. Produit (cartésien)

$R \times S = R \bowtie S$

A
x
y

$\times$

B
a
b
c

$=$

A	B
x	a
x	b
x	c
y	a
y	b
y	c

Soit R et S deux relations alors

$R \times S \equiv$

`R TIMES S` $\equiv$

```
select * from R cross join S
```
$\equiv$

ANTE $id(R) \cap id(S) = \varnothing$:

$R \bowtie S$

Notes

- L'antécédent garantit que $R \bowtie S = R \times S$.
- L'antécédent doit être aménagé en cas de relaxation de la compatibilité.
- Le produit cartésien n'est pas commutatif en SQL.

3.2.3. Semi-jointure (*matching*)

$$R \ltimes S = (R \bowtie S)\ \pi\ R$$

A	B
a1	b1
a2	b1
a3	b3
a4	b4

$\ltimes$

B	C
b1	c1
b2	c2
b3	c3
b3	c4

$=$

A	B
a1	b1
a2	b1
a3	b3

Soit R et S deux relations alors

$R \ltimes S \equiv$

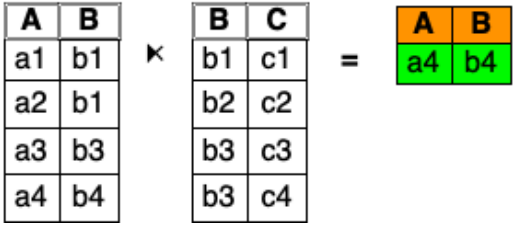
`R MATCHING S` $\equiv$

`select r1, …, rn from R natural join S` $\equiv$

$(R \bowtie S)\ \pi\ id(R)$

3.2.4. Semi-différence (*not matching*)

$R \ltimes S = R - (R \ltimes S)$



Soit R et S deux relations alors

$R \ltimes S \equiv$

`R NOT MATCHING S` $\equiv$

`select * from R except select` r_1`, …,` r_n `from R natural join S` $\equiv$

$R - (R \ltimes S)$

3.2.5. Extension

$$R\ \xi\ \{C:=f(A,B)\} = (R \bowtie F)\ \rho\ f':C$$

A	B
a1	b1
a2	b1
a3	b3
a4	b4

$\xi\ \{C:=f(A,B)\} =$

A	B	C
a1	b1	f(a1,b1)
a2	b1	f(a2,b1)
a3	b3	f(a3,b3)
a4	b4	f(a4,b4)

Définition

Une relation F correspond à une fonction $f(p_1{:}T_1, \ldots, p_n{:}T_1)$ de type T lorsque $def(F) = \{p_1{:}T_1, \ldots, p_n{:}T_1, f{:}T\} \bigwedge F = (F\ \sigma\ f{=}f(p_1, \ldots, p_n))$

3.2.5.1. Solution utilisant les seuls opérateurs relationnels primaires

Soit R une relation,
soit F la relation correspondant à la fonction $f(p_1{:}T_1, \ldots, p_n{:}T_1)$ de type T alors

$R \; \xi \; \{c := f(p_1, \ldots, p_n)\} \equiv$

```
R EXTEND {c := f(p1, …, pn)} ≡
select R.*, f(p1, …, pn) as c from R ≡
```

ANTE

$c \notin id(R) \bigwedge$
$def(R) \supseteq def(F) - \{f{:}T\} \bigwedge$
$R \; \pi \; \{p_1, \ldots, p_n\} \subseteq F \; \pi \; \{p_1, \ldots, p_n\} :$
$R \bowtie (F \; \rho \; f{:}c)$

Notes

- L'antécédent signifie simplement que
  (1) l'identifiant c ne dénote pas déjà un attribut de R,
  (2) chaque paramètre de F correspond à un attribut de R,
  (3) tout tuple de R à son correspondant dans F.
- La construction de relations en lieu et place des fonctions est un élément important dans la démonstration de la complétude de l'algèbre relationnelle.
- Le symbole utilisé pour l'extension varie beaucoup d'un auteur à l'autre. Certains utilisent α. D'autres présentant l'opération comme une extension de la projection et utilisent conséquemment le symbole π.

Exercice

- En pratique, plusieurs affectations peuvent être spécifiées dans la portée de l'extension, elles sont réputées être réalisées indépendamment il est donc possible de les exécuter concurremment.
  - En conséquence, $R \,\xi\, \{r_1 := exp_1 \; ; \; r_2 := exp_2\}$ pourrait ne pas être équivalent à $(R \,\xi\, \{r_1 := exp_1\}) \,\xi\, \{r_2 := exp_2\}$ en particulier si exp_2 fait appel à l'attribut r_1.
- Qu'en est-il en **SQL** ? Par exemple, quel est le résultat de

```
select 5 as x, x as y
from (values (1, 2)) as A (x, y) ;
```

3.2.5.2. Solution utilisant le modèle

Soit R une relation,
soit $f(p_1{:}T_1, \ldots, p_n{:}T_1)$ une fonction de type T alors
$R \xi \{c := f(p_1, \ldots, p_n)\} \equiv$
`R EXTEND {c := f(p1, …, pn)}` ≡
`select *, f(p1, …, pn) as c from R` ≡
ANTE
$c \notin id(R) \bigwedge$
$\{p_1, \ldots, p_n\} \subseteq id(R)$:
SOIT
$Z := def(R) \cup \{c{:}T\}$:
$(Z; \{(Z; val(t) \cup \{(c, f(t.p_1, \ldots, t.p_n))\}) \mid t \in val(R)\})$

3.2.6. Image

Soit R une relation et t un tuple alors

$R \circ t \equiv$

$R(t) \equiv$

`IMAGE_OF t IN R` $\equiv$

$R \bowtie RELATION[t\ \pi\ (id(t) \cap id(R))]$

Soit R une relation et t un tuple,
soit $\{x_1, \ldots, x_n\} := id(t) \cap id(R)$ alors

$R \circ t \equiv$

```
select * from R where R.x1=t.x1 and … and R.xn=t.xn
```

Note

- Si R et t n'ont pas d'atttibuts communs ($n=0$), la condition est vraie !

3.2.7. Image du tuple courant

Définition

Au sein d'une expression relationnelle référant à une relation R, le tuple « courant » d'une relation R a pour valeur le tuple de la relation R effectivement utilisé au moment de l'évaluation de ladite expression. Il est noté R[*] en Discipulus (ou `TUPLE{*} IN R` en TutorialD).

L'image du tuple courant d'une relation R (donc la valeur de relation contenant ce seul tuple courant) est notée *!!R* en Discipulus et en TutorialD. C'est-à-dire :

!!R ≡ R ∘ R[*]

`!!R` ≡ `IMAGE_OF TUPLE{*} IN R`

En SQL, il faut recourir à une dénotation énumérative pour désigner le tuple courant

Soit $\{x_1, \ldots, x_n\} := id(R)$ alors

R[*] ≡ (x₁, …, xₙ)

!!R ≡ (values(x₁, …, xₙ))

3.3. Opérateurs spécialisés

À développer

3.4. Opérateurs de regroupement

À développer

3.5. Opérateurs d'agrégation

Un opérateur d'agrégation (ou fonction d'agrégation) est traditionnellement présenté comme un opérateur dont les paramètres sont des « colonnes ». Bien que cette métaphore puisse guider l'intuition de façon satisfaisante dans les cas simples, elle ne rend pas compte de toutes les possibilités couvertes par ces opérateurs. De plus, elle fait appel à un concept, la « colonne » (qu'on aurait tort d'identifier à l'attribut) qui est n'est pas définie dans le modèle relationnel. Il faut plutôt considérer l'opérateur d'agréation comme une fonction définie sur une relation et *certains* de ces attributs. En couplant de telles fonctions avec les autres opérateurs relationnels, il est possible de couvrir non seulement les fonctions d'agrégation, mais aussi les fonctions de fenêtrage (*window functions*) et la clause *lateral* de SQL.

Nous laissons pour le moment la définition formelle en exercice et renvoyons à [Date2015a] pour la solution.

3.6. Opérateurs de graphes

À développer

Les graphes sont en fats des relations binaires internes. En ce sens tous les opérateurs requis pour les manipuler sont déjà disponibles (et même plus) par l'entremise des opérateurs primaires. Il n'en demeure pas moins qu'il est possible, et souvent très commodes, de définir des opérateurs spéciques tirant parti de l'internalité, de la binarité et de la transitivité cololaire.

3.7. Opérateurs déclassés

- division, DIVIDE BY…
- sommaire, SUMMARIZE…

Produit le 2025-01-09 15:23:11 UTC