Modélisation de données

Survol de la théorie relationnelle

TMR_01
v240b

2022-01-06

UNIVERSITÉ DE SHERBROOKE

Christina.Khnaisser@USherbrooke.ca
Luc.Lavoie@USherbrooke.ca

Plan

- Une intuition
- Préambule
- Modèles et représentations
- Types, valeurs, variables et relations
- Théorie relationnelle
- Opérations
- Contraintes
- De la théorie aux langages
- De l'absence
- Références
- Les colles du prof

Une intuition

- Comment représenter les données de la façon adéquate ?
- Que veut-on dire par « adéquat » ?
- Que nous disent les anciens ?

Que nous disent les anciens ?

- Les fonctionnaires d'Hammourabi (XVIII^{e} siècle avant JC)
- Ératosthène (III^{e} siècle avant JC)
- Les ingénieurs romains (I^{er} siècle après JC)

Que veut-on dire par adéquat ?

- cohérent – *pas de contradiction*
- valide – *conforme au modèle*
- efficace – *conforme aux attentes*

- évolutif – *adaptabilité aux « changements »*
- efficient – *« bonne » utilisation des ressources*
- complet – *couverture « suffisante » du problème*

- cohérent et complet ? réfutable ?

Quelle est donc cette intuition ?

matricule	nom	adresse
15113150	Paul	ᐳᐅᖏᓂᑐᖅ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Préambule

- Qu'est-ce qu'une relation
- Quelques mots pour le dire
- Vérification des relations
- Au-delà du binaire
- Une intuition géniale de Codd
  - la logique et les relations au service de l'information

Préambule – Qu'est-ce donc qu'une relation ?

- La réponse mathématique :
  - Un sous-ensemble d'un produit cartésien d'ensembles.

- Soit les ensembles
  - Nom = {Paul, Éliane, Mohamed, Sergeï, Diego}
  - Ville = {ᐅᕕᖏᓂᖅ, Blanc-Sablon, Tadoussac, Chandler, Adélaïde, San Diego}
- Un exemple de valeur de relation
  - v1 = {(Paul, ᐅᕕᖏᓂᖅ), (Éliane, Blanc-Sablon), (Mohamed, Blanc-Sablon), (Sergeï, Chandler), (Sergeï, San Diego)}
- Une valeur de relation est aussi appelée, dans certains contextes, *constante* (de relation) ou *littéral* (de relation).

Préambule – Qu'est-ce donc qu'un type de relation ?

- Un type est un ensemble fini de valeurs.
- Un type de relation est donc un identifiant qui réfère à un ensemble de valeurs d'une même relation.

- Un exemple de type de relation
  - TVN = $Nom \times Ville$;

Préambule – Qu'est-ce donc qu'une variable relation ?

- Une variable est un identifiant qui réfère à une valeur.
- Une variable de relation est donc un identifiant qui réfère à une valeur de relation.

- Un exemple de variable de relation :
  - Visite : TVN ;

- À sa déclaration, la variable réfère à l'ensemble vide {}.
- L'opération d'affectation permet d'en changer la référence :
  - Visite := v1 ;

Préambule – Qu'est-ce donc qu'un invariant ?

- Un invariant est une condition qui doit être vraie en tout temps.
- La déclaration d'une variable est toujours associée à un invariant qui prescrit que la valeur associée à la variable doit faire partie (être un élément) du type associé à la variable.
- Par exemple,
  - attendu les déclarations
    - TVN : $Nom \times Ville$;
    - Visite : TVN;
  - alors les invariants implicites sont
    - Visite $\subseteq$ TVN ;
    - Visite $\subseteq$ $Nom \times Ville$;

Préambule – La vérification des affectations est capitale !

- Soit les ensembles
  - Nom = {Paul, Éliane, Mohamed, Sergeï, Diego}
  - Ville = {ᐳᕃᖏᓂᑲᖅ, Blanc-Sablon, Tadoussac, Chandler, Adélaïde, San Diego}
  - v2 = {(Paul, ᐳᕃᖏᓂᑲᖅ), (Éliane, Blanc-Sablon), (*Tadoussac*, Blanc-Sablon), (Sergeï, Chandler), (Sergeï, San*s* Diego)}
- La valeur v2 ne fait pas partie du type *TVN* :
  - v2 $\notin$ *Nom* $\times$ *Ville*
  - v2 $\notin$ *TVN*
- La variable Visite : *TVN* ne peut donc pas référer à v2.
  - L'affectation « Visite := v2 » induit donc une incohérence.

Préambule – Vocabulaire

- L'expression « variable de relation » est souvent abrégée en *relvar*.
- Par analogie, il y a aussi *reltype* et *relconst*.
- Un identifiant est aussi appelé, dans certains contextes, *identificateur*.

Préambule – Notation

- Par souci d'expressivité, on permet souvent la déclaration d'une *relvar* sous une forme compacte, par exemple
  - RELVAR Visite {*Nom*, *Ville*}

Préambule – Remarque

- Les ensembles sont des cas particuliers de relations en ce sens qu'il y a une bijection triviale entre une relvar définie sur un seul ensemble et cet ensemble lui-même.
- Par exemple
  - entre
    - la variable RELVAR V {Ville}
  - et l'ensemble
    - Ville

Préambule – Relation n-aire

- **Visite** est une variable de relation binaire, qu'advient-il des relations de degré supérieur ?

- Soit les ensembles
  - Nom = {Paul, Éliane, Mohamed, Sergeï, Diego}
  - Ville = {ᐳᕕᕐᓂᑐᖅ, Blanc-Sablon, Tadoussac, Chandler, Adélaïde, San Diego}
  - Matricule = {15113150, 15112354, 15113870, 15110132}

- RELVAR Etudiant (Matricule, Nom, Ville)
  - **Etudiant ⊆ Matricule × Nom × Ville**
  - Etudiant = { (15113150, Paul, ᐳᕕᕐᓂᑐᖅ),
    (15112354, Éliane, Blanc-Sablon),
    (15113870, Mohamed, Blanc-Sablon),
    (15110132, Sergeï, Chandler),
    (15110132, Sergeï, San Diego)}

Préambule – Prédicat et propositions

- Une relation représente un prédicat...
  qui représente une partie de la réalité.

- Un prédicat est une fonction avec une ou plusieurs variables.

- En associant des valeurs aux variables du prédicat,
  on obtient une proposition représentée par un tuple.

- Une proposition est un énoncé
  à propos d'une partie de la réalité.

Préambule – Fondement de la théorie relationnelle de Codd

- Une relation comprend tous les tuples dont la proposition est vraie relativement à la partie de la réalité modélisée.

Préambule – Les relations et la logique

- Quels sont les prédicats représentés par les relations Visite et Etudiant ?
- Peut-on les déduire à partir des relations ?
- Peut-on déduire les relations à partir des prédicats ?
- Y a-t-il un lien entre prédicat et «information» ?
- Comment définir «information» ?

Préambule – Que choisir et pourquoi ?

- Qu'arrive-t-il si un type est utilisé plus d'une fois dans le produit cartésien définissant un *reltype* ?
- Est-ce une bonne idée de dépendre de l'ordre des valeurs pour les associer au bon type ?

Quel modèle pour la théorie relationnelle ?

- Une matrice ?
- Un graphe ?
- Une table ?
- Le modèle (info-)relationnel !

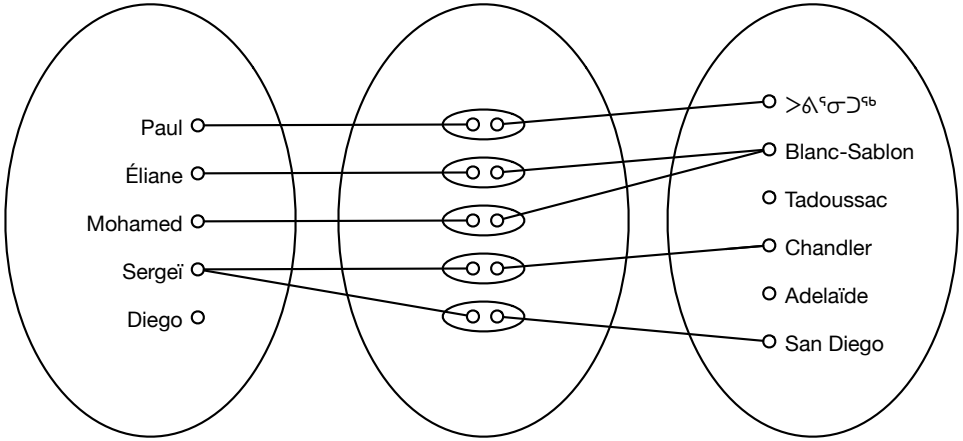
La matrice ?

- Nom = {Paul, Éliane, Mohamed, Sergeï, Diego}
- Ville = {>ᕿᖏᑐᖅ, Blanc-Sablon, Tadoussac, Chandler, Adélaïde, San Diego}
- Visite = {(Paul, >ᕿᖏᑐᖅ), (Éliane, Blanc-Sablon), (Mohamed, Blanc-Sablon), (Sergeï, Chandler), (Sergeï, San Diego)}

	>ᕿᖏᑐᖅ	Blanc-Sablon	Tadoussac	Chandler	Adlaïde	San Diego
Paul	1	0	0	0	0	0
Éliane	0	1	0	0	0	0
Mohamed	0	1	0	0	0	0
Sergeï	0	0	0	1	0	1
Diego	0	0	0	0	0	0

Le graphe ?

- Nom = {Paul, Éliane, Mohamed, Sergeï, Diego}
- Ville = {>ᐅᖏᕐᓂᖅ, Blanc-Sablon, Tadoussac, Chandler, Adélaïde, San Diego}
- Visite = {(Paul, >ᐅᖏᕐᓂᖅ), (Éliane, Blanc-Sablon), (Mohamed, Blanc-Sablon), (Sergeï, Chandler), (Sergeï, San Diego)}



La table ?

- Nom = {Paul, Éliane, Mohamed, Sergeï, Diego}
- Ville = {ᗒᗅᕐᓂᑐᖅ, Blanc-Sablon, Tadoussac, Chandler, Adélaïde, San Diego}
- Visite = {(Paul, ᗒᗅᕐᓂᑐᖅ), (Éliane, Blanc-Sablon), (Mohamed, Blanc-Sablon), (Sergeï, Chandler), (Sergeï, San Diego)}

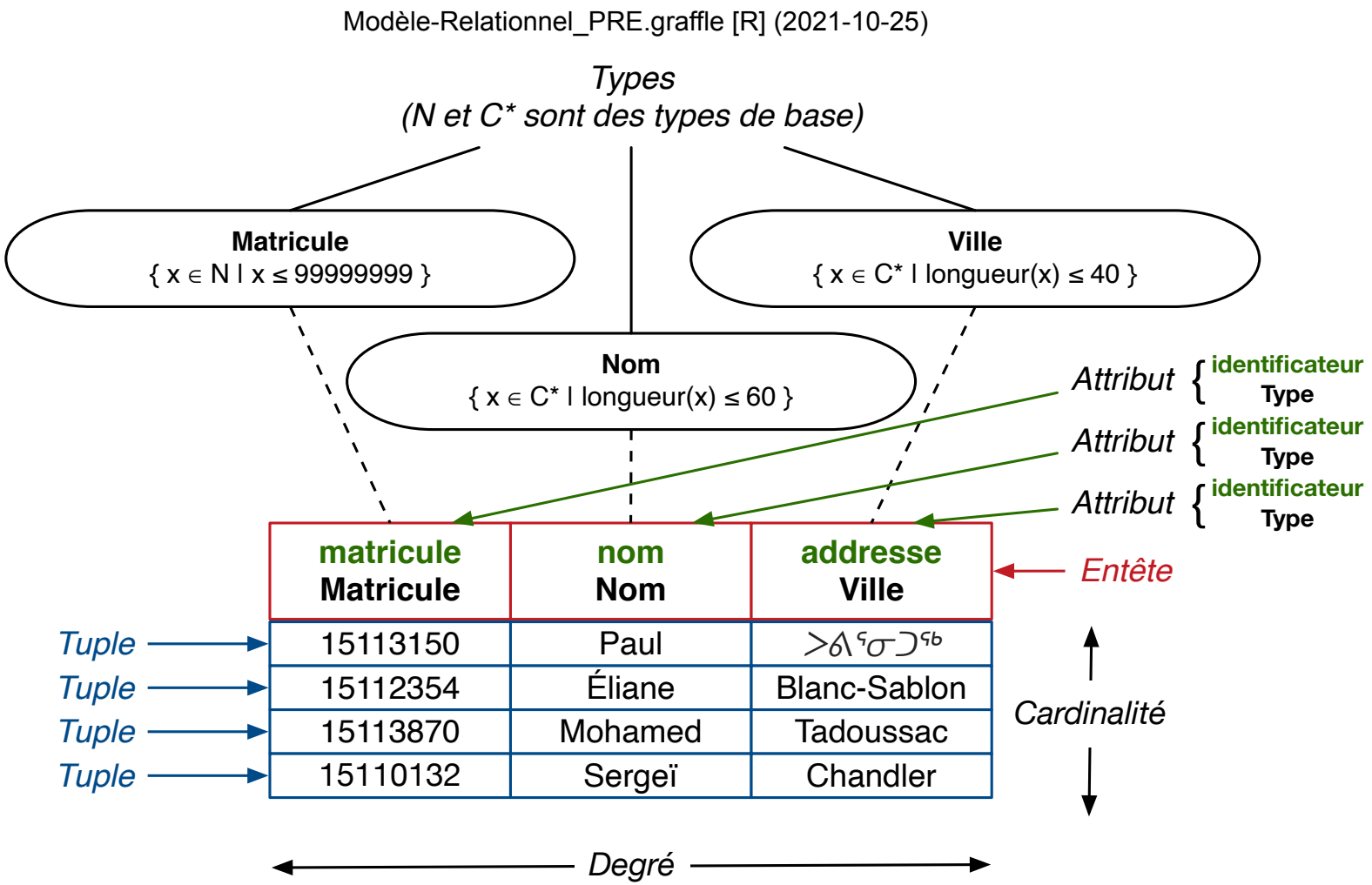
Nom	Ville
Paul	ᗒᗅᕐᓂᑐᖅ
Éliane	Blanc-Sablon
Mohamed	Blanc-Sablon
Sergeï	Chandler
Sergeï	San Diego

Lequel choisir et pourquoi ?

- Chaque représentation est-elle « complète » ?
  - Qu'entend-on par complet ?

- Chaque représentation facilite *certaines* opérations
  - Lesquelles ?

Notre choix : le modèle info-relationnel illustré par la relation *Etudiant*



Types, valeurs, variables et relations

- Type, type de base, sous-type et relation (bis)
- Référence, dénotation et variable
- Logique et relations (bis)

TVVR – Rappels (1)

- Un type de base est un ensemble *fini* de valeurs *propres*.

- Un sous-type est un sous-ensemble d'un type de base déterminé par une contrainte (qui en restreint les valeurs).

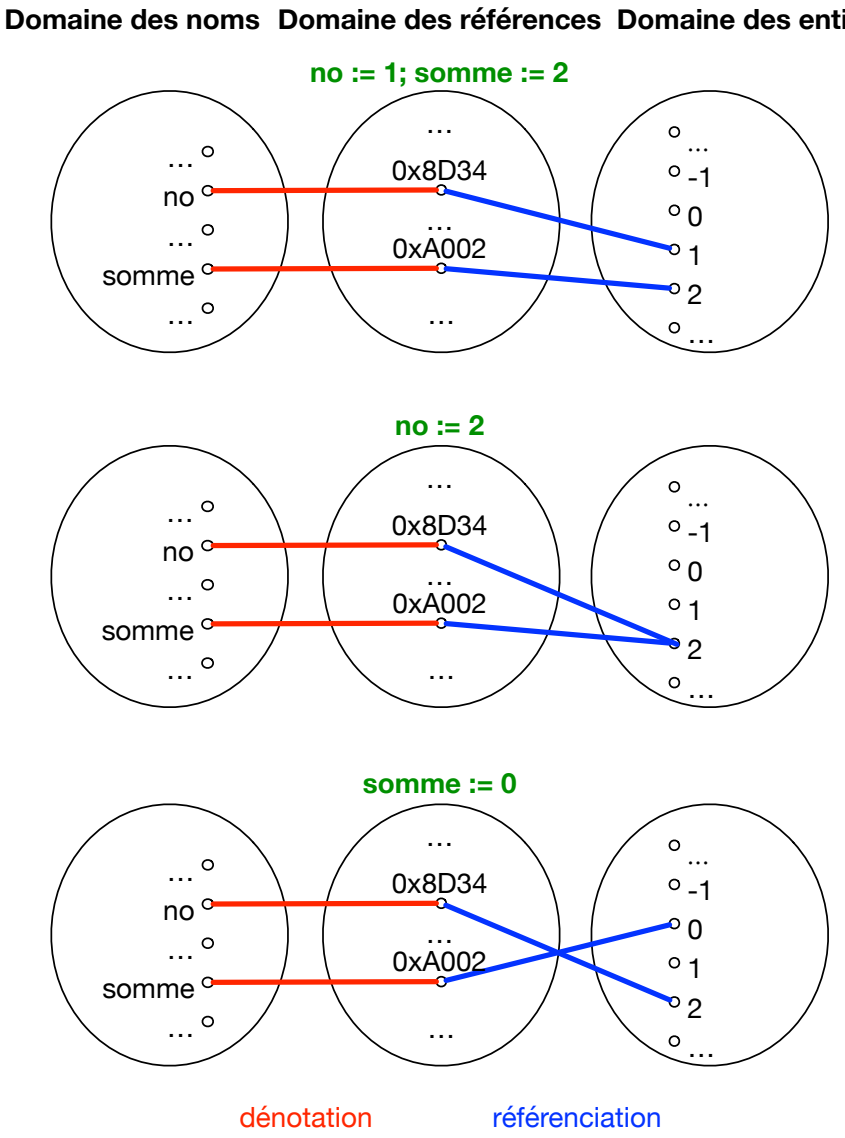
- Une relation de degré n est un sous-ensemble du produit cartésien de n types :

$$D_1 \times D_2 \times \ldots \times D_n$$

TVVR – Rappels (2)

- Une référenciation est une fonction associant une référence (adresse) à une valeur.
- Une dénotation est une fonction associant un identifiant (nom) à une référence.
- Une variable est une dénotation.
- Une variable est dite typée si, par construction, elle réfère toujours à une valeur du type qui lui est associé (par une définition, souvent appelée déclaration en informatique).

TVVR – Variable et affectation



TVVR – Les relations et la logique

relation	prédicat modélisé par un ensemble de tuples représentée par un tableau (une table).
tuple	proposition modélisée par un ensemble d'attribut représenté par une ligne (un enregistrement).
attribut	variable typée définie par un prédicat représentée par une cellule (un champ) dans un tableau.
contrainte	expression logique.
relvar	variable référant une (valeur de) relation.
modèle logique	un ensemble de définitions de relvar et un ensemble de définitions de contraintes.
base de données	un ensemble de relvars conformes à un modèle logique (donc aux définitions des relvars et des contraintes).

Note : dans le but de faciliter la gestion de la complexité, on permet généralement qu'une base de données puisse être définie à l'aide de plusieurs schémas ; chaque identifiant défini par un schéma est alors qualifié (préfixé) par le nom du schéma.

TVVR – Information et relvar, relvar et prédicat, tuple et proposition

- À chaque relvar on associe un prédicat qui en est « l'interprétation » informationnelle.
- En substituant les valeurs des attributs d'un tuple aux variables correspondantes du prédicat de la relvar, on obtient une proposition.
- À chaque tuple est alors associée une proposition qui en est « l'interprétation » informationnelle.

TVVR – Remarques

- L'interprétation informationnelle d'un tuple est aussi appelée, dans certains contextes, « fait ».
- L'association d'une valeur à un attribut passe par la donnée... et c'est très important !
  - C'est la donnée qui permet de rendre l'attribut « manipulable » informatiquement et, corolairement, le tuple, la relvar, le prédicat.
  - À son tour, le prédicat rendu ainsi manipulable permet la représentation et la transmission de l'information.
  - ... et bien d'autres choses encore !

TVVR – De la théorie des types

- La définition du type présentée ici est minimale. La plupart des auteurs s'entendent pour associer au type un ensemble d'opérateurs.
- La théorie fondamentale des types a été établie dans l'article suivant :
  - Luca Cardelli, Peter Wegner (1985) : *On understanding types, data abstraction, and polymorphism.* ACM Computing Surveys 17, 4; pp. 471–523. DOI:http://dx.doi.org/10.1145/6041.6042
- Elle est prolongée, aux fins d'interopérabilité avec le modèle relationnel dans :
  - Chris J. Date (2016) : *Type inheritance and relational theory: subtypes, supertypes, and substitutability*. O'Reilly Media, Sebastopol, CA. ISBN:978-1-4919-5999-2

Fondements

- Attributs
- Tuples
- Relations
- Opérations
- Contraintes

Fondements – Attributs

- 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 écrit-on l'attribut a.

Fondements – Tuples

- Soit a_i des identifiants distincts et D_j des types, un tuple t est défini comme suit :
  - $t \triangleq (\{a_1{:}D_1, a_2{:}D_2, \ldots, a_n{:}D_n\} ; \{(a_1,v_1), (a_2,v_2), \ldots (a_n,v_n)\})$
  - avec $\forall\, i : 1 \leq i \leq deg(t) \Longrightarrow val(t, a_i) \in def(t, a_i)$
- où
  - $def(t) = \{a_1{:}D_1, a_2{:}D_2, \ldots, a_n{:}D_n\}$ entête de t
  - $def(t, a_i) = D_i$ type de a_i de t
  - $val(t) = \{(a_1,v_1), (a_2,v_2), \ldots (a_n,v_n)\}$ valeur de t
  - $val(t, a_i) = v_i$ valeur de a_i de t
  - $deg(t) = n$ degré de t
  - $id(t) = \{a_1, a_2, \ldots, a_n\}$ les identifiants d'attributs de t

Fondements – Relations

- Soit a_i des identifiants distincts, D_j des types et t_k des tuples, une relation R est définie comme suit :
  - $R \triangleq (\{a_1{:}D_1, a_2{:}D_2, \ldots, a_n{:}D_n\} ; \{t_1, t_2, \ldots, t_m\})$
  - avec $\forall\, i : 1 \leq i \leq card(R) \Longrightarrow def(R){=}def(t_i)$
- Où
  - $def(R) = \{a_1{:}D_1, a_2{:}D_2, \ldots, a_n{:}D_n\}$ entête de R
  - $def(R, a_i) = D_i$ type de a_i de R
  - $val(R) = \{t_1, t_2, \ldots, t_m\}$ valeur de R
  - $deg(R) = n$ degré de R
  - $card(R) = m$ cardinalité de R
  - $id(R) = \{a_1, a_2, \ldots, a_n\}$ identifiants d'attributs de R

Fondements – Illustration

Grâce aux contraintes sur les tuples et les relations, la représentation tabulaire initiale est donc bien fondée.

Quatre tuples (ayant le même entête)

matricule : Matricule	nom : Nom	adresse : Ville	
matricule : 15113150	nom : Paul	adresse : ᐊᓕᒍᖅ	t1
matricule : Matricule	nom : Nom	adresse : Ville	
matricule : 15112354	nom : Éliane	adresse : Blanc-Sablon	t2
matricule : Matricule	nom : Nom	adresse : Ville	
matricule : 15113870	nom : Mohamed	adresse : Tadoussac	t3
matricule : Matricule	nom : Nom	adresse : Ville	
matricule : 15110132	nom : Sergeï	adresse : Chandler	t4

Une relation comprenant quatre tuples

matricule : Matricule	nom : Nom	adresse : Ville	
matricule : Matricule	nom : Nom	adresse : Ville	
matricule : 15113150	nom : Paul	adresse : ᐊᓕᒍᖅ	t1
matricule : Matricule	nom : Nom	adresse : Ville	
matricule : 15112354	nom : Éliane	adresse : Blanc-Sablon	t2
matricule : Matricule	nom : Nom	adresse : Ville	
matricule : 15113870	nom : Mohamed	adresse : Tadoussac	t3
matricule : Matricule	nom : Nom	adresse : Ville	
matricule : 15110132	nom : Sergeï	adresse : Chandler	t4

La représentation compacte usuelle de cette même relation

matricule : Matricule	nom : Nom	adresse : Ville
15113150	Paul	ᐊᓕᒍᖅ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Fondements – raccourcis et notations

- Notations équivalentes à val(t, a_i)
  - t.a_i
  - a_i(t)
  - t(a_i)
  - t[a_i]
  - a_i from t
- Nous utiliserons fréquemment
  - t.a_i

Opérations

- Premier essai
- Renommage
- Deuxième essai
- Et encore plus !

Fondements – Opérations naturelles

Restriction
R σ cond

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

Projection
R π {A, C}

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

Jointure naturelle
R ⋈ S

A	B
a1	b1
a2	b1
a3	b3
a4	b4

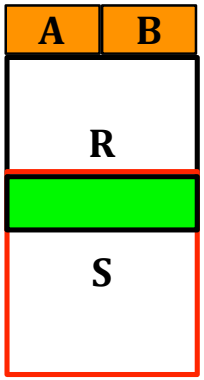
⋈

B	C
b1	c1
b2	c2
b3	c3
b3	c4

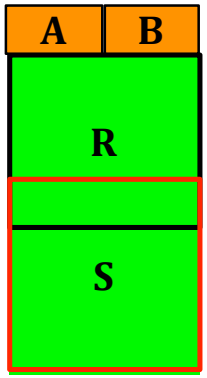
=

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

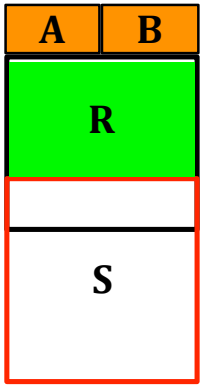
Intersection
R ∩ S



Union
R ∪ S



Différence
R - S



Note : Le symbole de projection π est souvent omis, à l'instar de la multiplication dans les polynômes.

Fondements – Opération de renommage

- La présence de l'entête dans chacun des tuples et chacune des relations permet de définir une opération structurelle, le renommage.

Renommage
R ρ A:C

A	B
a1	b1
a2	b2
a3	b3

ρ A:C =

C	B
a1	b1
a2	b2
a3	b3

- L'entête d'une relation est conservé dans le catalogue du SGBDR.
- Le catalogue est la description des modèles relationnels du SGBDR sous la forme d'une BD dont le modèle relationnel est lui-même dans le catalogue, comme tous les autres modèles relationnels de toutes les autres BD du SGBDR.

Fondements – Règles d'application

- Les trois opérations ensemblistes ne sont bien définies que si les entêtes des opérandes sont identiques.
- L'opération de projection n'est bien définie que si tous les identifiants d'attributs sont définis dans l'entête de la relation.
- 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.
- L'opération de renommage n'est bien définie que si l'identifiant à changer est défini dans l'entête de la relation et que le nouvel identifiant ne l'est pas.

Fondements – Opération d'intersection

- On constate que $\mathbf{R \cap S = R - (R\text{-}S) = R \bowtie S}$
- L'opération d'intersection n'est donc pas « nécessaire ».
- Elle est toutefois «utile» en ce sens qu'elle favorise la lisibilité, voire l'optimisation.

- Nous retirons l'intersection des opérations de base, mais y ajouterons le renommage.

- Opérations des base :
  - relationnelles : produit, restriction, jointure, renommage
  - ensemblistes : union, différence

Fondements – Opérations de base

Restriction
R σ cond

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

Projection
R π {A, C}

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

Jointure (naturelle)
R ⋈ S

A	B
a1	b1
a2	b1
a3	b3
a4	b4

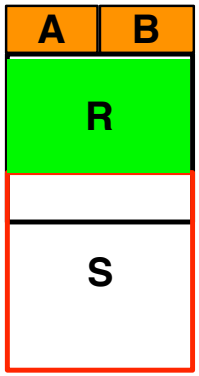
⋈

B	C
b1	c1
b2	c2
b3	c3
b3	c4

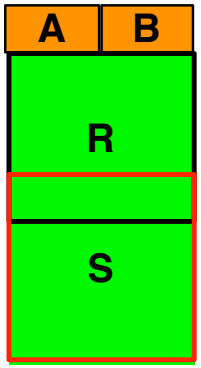
=

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

Différence
R – S



Union
R ∪ S



Renommage
R ρ A:C

A	B
a1	b1
a2	b2
a3	b3

ρ A:C =

C	B
a1	b1
a2	b2
a3	b3

Fondements – Produit cartésien ou jointure ?

- Constatons d'abord que l'un se calcule à l'aide de l'autre :
  - $R \times S = R \bowtie S$
    pourvu que R n'ait aucun attribut commun avec S
    ce dont on peut s'assurer par renommage ;
  - $R \bowtie S = (R \times S)\ \sigma\ (R.a_1 = S.a_1 \wedge \ldots \wedge R.a_n = S.a_n)$
    pourvu qu'on ait la notion pointée à disposition.

Préférez-vous le chocolat ou le cacao ?

Fondements – Opérateurs : ensemble minimal ?

- En pratique, deux ensembles de six opérions de base sont proposés, selon les auteurs :
  - Codd $\pi\ \sigma\ \times\ \cup\ -\ \rho$
  - Date $\pi\ \sigma\ \bowtie\ \cup\ -\ \rho$
- Nous utiliserons toutefois tantôt l'un, tantôt l'autre, car ils proposent un bon équilibre entre minimalisme et expressivité.

- Note : il existe un ensemble minimal complet ne comprenant que deux opérations, voir l'algèbre $\mathcal{A}$ proposée dans [Date 2007]..

Fondements – Quelques opérations construites

Extension
(augmentation)
$R\ \xi\ C{:}f = (R \bowtie F)\ \rho\ f'{:}C$

A	B
a1	b1
a2	b1
a3	b3
a4	b4

$\xi\ C{:}f =$

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

Note : Le symbole de l'extension varie beaucoup d'un auteur à l'autre.

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

Produit
$R \times S = R \bowtie S$
(jointure sans attributs partagés)

A
x
y

$\times$

B
a
b
c

$=$

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

Semi-différence
(not matching, anti-join…)
$R \triangleright S = R - (R \ltimes S)$

A	B
a1	b1
a2	b1
a3	b3
a4	b4

B	C
b1	c1
b2	c2
b3	c3
b3	c4

$=$

A	B
a4	b4

Fondements – Autres opérations construites

- Se reporter au module BD012 qui comprend une suite d'exemples de requêtes relationnelles utilisant les opérations de base et quelques opérations construites.

Contraintes

- Approche classique
  - Contraintes générales
  - Contraintes spécifiques
  - Vocabulaire
- Approche typée (fondée sur la théorie des types)
  - Types et contraintes
  - Application aux entités scalaires
  - Application aux entités non scalaires
  - Application à la base de données elle-même
- Retour sur la nature d'une base de données
  - Une définition… très simple
- Évaluation des contraintes
  - Moment
  - Suspension
- Et les invariants ?

Approche classique

- Contraintes générales
- Contraintes spécifiques
  - Portée
  - Rôle
- Vocabulaires

Approche classique – Contraintes

- Contrainte (générale)
  - Une contrainte est une expression logique applicable aux objets d'une base de données ; l'expression qui doit être maintenue vraie tout au long de l'existence de la base de données.
- Contrainte (spécifique)
  - Un cas particulier de contrainte générale.
  - On désigne souvent une contrainte spécifique selon
    - sa portée par rapport à un objet particulier de la base de données : contrainte d'attribut, contrainte de relation, contrainte de base de données ;
    - son rôle : clé candidate, clé référentielle.

Approche classique – Contraintes spécifiques (portée)

- Contrainte de type
  - restreindre l'ensemble des valeurs d'un type (pour en faire un sous-type, par exemple).
- Contrainte d'attribut
  - limiter la valeur des attributs d'un tuple entre eux.
- Contrainte de relation
  - limiter globalement les valeurs de la relation (donc celles des tuples de la relation entre eux) ;
  - exemple : clés candidates.
- Contrainte d'intégrité référentielle
  - limiter globalement les valeurs d'attributs d'une relation relativement à une clé candidate d'une autre ;
  - exemple : clés référentielles (étrangères).

Approche classique – Contraintes spécifiques (rôle)

- Clé (d'une relation R)
  - sous-ensemble d'attributs déterminant un tuple unique au sein de la relation.
    - Soit X une clé de R :
      $\#R = \#(R\ \pi\ \{X\}) \equiv card(R) = card(R\ \pi\ \{X\})$

- Clé référentielle (d'une relation R sur une relation S)
  - sous-ensembles d'attributs dont les valeurs sont restreintes aux seules valeurs effectives de la clé d'une (autre) relation
    - Soit X une clé référentielle de R sur S :
      $(R\ \pi\ \{X\}) \subseteq (S\ \pi\ \{X\}) \wedge (\#S = \#(S\ \pi\ \{X\}))$

Approche classique – Vocabulaire

- Clé irréductible
  - clé dont on ne peut retirer aucun attribut sans qu'elle cesse d'être une clé.
- Clé candidate
  - synonyme : clé irréductible.
- Surclé
  - tout ensemble d'attributs contenant une clé.

Approche classique – Contraintes, une suite ?

- Nous reprendrons l'étude des clés (et des dépendances fonctionnelles) quand nous aurons mieux maitrisé la théorie relationnelle.

Approche typée

- Types et contraintes
- Application aux entités scalaires
- Application aux entités non scalaires
- Application à une chronologie simple

Approche typée – Types et contraintes

Rappel

- Types
  - type de base :
    - ensemble fini de valeurs propres ;
    - aussi appelé domaine, «type racine» ou (*root type*).
  - sous-type :
    - sous-ensemble d'un type déterminé par une contrainte ;
    - plusieurs auteurs considèrent qu'il doit s'agir d'un sous-ensemble propre.

Approche typée – Types scalaires et non scalaires

- entité :
  une valeur, un type, une variable, etc.
- type scalaire :
  un type dont les valeurs sont atomiques
- type non scalaire :
  un type dont les valeurs ne sont pas atomiques
  (dotées d'une structure dont les parties sont dénotables)

Approche typée – Exemples de types scalaires

- Entier
  - type Cardinal ::= Entier contrainte (valeur >= 0) ;
  - type Positif ::= Entier contrainte (valeur > 0) ;
  - type Positif_pair ::= Cardinal contrainte (valeur mod 2 = 0) ;

- Texte
  - type Nom ::= Texte contrainte (longueur(valeur) <= 60)) ;
  - type Matricule ::= Texte contrainte (valeur ~ ‘ [0-9]{8}’) ;

Approche typée – Exemples de types non scalaires

- type Personne ::=
  tuple {mat:Matricule, nom:Nom, ville:Ville} ;
- type Tandem ::=
  tuple {e1:Matricule, e2:Matricule}
  contrainte (e1<e2) ;
- type Registre ::=
  relation Personne ;
- type Registre_naissance ::=
  relation {mat:Matricule, a:Entier, m:Entier, j:Entier}
  contrainte (a>1608 et 1<=m<=12 et 1<=j<= 31) ;

Approche typée – Application à une chronologie simple (1)

- -- Types de tuple
- type Mesure ::=
  tuple {x : Cardinal ; debut : Estampille ; fin : Estampille}
  contrainte (debut <= fin) ;

- -- Types de relation
- type Chronologie ::=
  relation {debut : Estampille}
  contrainte (#Chronologie = # (Chronologie $\boldsymbol{\pi}$ debut)) ; -- clé candidate sur {debut}
- type Table_de_Mesure ::=
  relation Mesure
  contrainte (#Mesure = # (Mesure $\boldsymbol{\pi}$ debut)) ; -- clé candidate sur {debut}

- -- Variables de relation
- relvar chrono : Chronologie ;
- relvar m : Table_de_Mesure
  contrainte (m π {debut}) $\subseteq$ (chrono π {debut}) $\wedge$ (#chrono = #(chrono π {debut})) ;
  -- clé référencielle sur {debut} vers chrono

Approche typée – Application à une chronologie simple (2)

- On constate :
  - que les clés candidates (PRIMARY KEY et UNIQUE) sont des cas particuliers de contraintes ;
  - que les clés référentielles (FOREIGN KEY) également ;
  - qu'elles sont parfaitement exprimables sans raccourcis syntaxiques.

- En conclusion :
  - il n'est pas utile d'inclure ces cas particuliers dans la théorie ni le modèle (du moins pas le modèle de base).
  - les raccourcis syntaxiques, lorsque bien définis et bien choisis, peuvent jouter de l'expressivité au langage.

Évaluation des contraintes

- Moment de l'évaluation
- Suspension

Évaluation des contraintes – Moment de l'évaluation

- À quel moment faut-il évaluer une contrainte ?
- À chaque fois qu'une variable (attribut, tuple, relvar...) est changée ?

- Nous aurons à examiner cette question lors de notre revue de SQL.

Évaluation des contraintes – Suspension de l'évaluation

- Doit-on permettre la suspension (et corolairement, la reprise) de l'évaluation des contraintes ?

- Nous aurons à examiner cette question lors de notre revue de SQL.

Retour sur les bases de données

- Une idée simple
- Fondements (bis)
- Exemple
- Conclusion

Retour sur les bases de données – une idée simple

- Et si une base de données n'était après tout qu'*une* variable d'un type non scalaire, «base» ?
- Nous aurions donc finalement trois types non scalaires :
  - tuple
  - relation
  - base
- Toutes les contraintes trouvent ainsi leur place naturelle à l'intérieur de la théorie des types. Nous n'avons plus besoin de cas particulier pour la «base de données».

Fondements – Base «de données»

- Soit v_i des identifiants distincts, D_j des types de relation et r_k des (valeurs de) relations, une base (de données) B est définie comme suit :
  - $B \triangleq (\{v_1:D_1, v_2:D_2,..., v_n:D_n\} ; \{r_1, r_2, ..., r_m\})$
  - avec $\forall\, i : 1 \leq i \leq card(B) \Longrightarrow def(B, v_i)=def(r_i)$
- Où
  - $def(B) = \{v_1:D_1, v_2:D_2,..., v_n:D_n\}$ entête de B
  - $def(B, a_i) = D_i$ type de a_i de B
  - $val(B) = \{r_1, r_2, ..., r_m\}$ valeur de B
  - $deg(B) = n$ degré de B
  - $id(B) = \{v_1, v_2, ..., v_n\}$ ensemble des identifiants de variables de relation de B

Retour sur les bases de données – Exemple

- type BDMC ::=
  base {chrono : Chronologie ; m : Table_de_Mesure}
  contrainte (m π {debut}) $\subseteq$ (chrono π {debut}) $\wedge$
  (#chrono = #(chrono π {debut}))
  -- clé référentielle {debut} vers chrono ;
- basevar b : BDMC ;

Ainsi on ne déclarerait plus *des* relvars (et des contraintes flottantes), mais *une* basevar.

Retour sur les bases de données – conclusion

- Une base de données, ce n'est donc qu'une «variable de base de données».

- La théorie, c'est plus simple !
- La théorie, c'est pratique (nous le verrons très bientôt).

Retour sur les invariants

Retour sur les invariants – aux origines

- En mathématiques, un invariant est une propriété d'un objet qui reste inchangée après l'application d'une opération.

- En informatique (logique axiomatique de Floyd-Hoare), un invariant est une assertion dont la valeur demeure inchangée par l'exécution d'une instruction, quel que soit l'état légitime qui prévaut au moment de l'exécution. Par exemple, l'invariant d'une boucle :

$$\frac{\{I \wedge B\}\ S\ \{I\}}{\{I\}\ \textit{tantque}\ B\ \textit{faire}\ S\ \textit{fin}\ \{I \wedge \neg\ B\}}$$

Retour sur les invariants – Une conclusion

- Les contraintes ne sont finalement que des invariants associés aux types et appliqués aux variables !

De la théorie aux langages

Plusieurs modèles ont été proposés sur la base d'une même théorie

Au quotidien toutefois, nous utilisons des langages.

- Modèle de Codd I
- Modèle de Codd II
- Modèle de Date
- Modèle d'Ullman
- Modèles SQL
- ...
- Et la liste n'est pas complète !
- ...
- Que choisir ?

De la théorie aux langages – En passant par les modèles !

- Il y a une théorie relationnelle.
- Il y a plusieurs modèles relationnels, par exemple
  - Modèle de Codd I
  - Modèle de Codd II
  - Modèle de Date
  - Modèle d'Ullman
  - Modèles SQL (ANSI:1992, ISO:1999, ISO:2006, ISO:2011, ISO:2016...)
- Pour chacun de ces modèles, il y a plusieurs langages (et même, plusieurs dialectes pour certains de ces langages)

De la théorie aux langages – Que choisir ?

- Pour l'exposé des principes relationnels, nous utiliserons toujours le modèle de Date.
- Pour la programmation SQL, nous présenterons des techniques permettant d'être aussi proche que possible du modèle de Date, en indiquant les écarts possibles en fonction du modèle SQL ISO 9075:2016.

Données absentes

Jusqu'à présent nous avons assumé que toutes les données étaient disponibles.

Dans la pratique, c'est rarement le cas.

- Pourquoi une donnée serait-elle absente ?
- Un modèle simple
- Solutions AVEC annulabilité
- Solutions SANS annulabilité
- Impact sur la logique utilisée par le modèle
- Et, non, le problème n'est pas résolu !

Données absentes
Pourquoi une donnée serait-elle absente ?

- Discussion en classe.

Données absentes – Un modèle simple (proposé par Codd)

- **NA**
  - L'information est sans objet (n'est **pas «applicable»**).
  - Dans ce cas, l'utilisation de l'annulabilité est à remettre en question ; une bonne modélisation permet généralement d'éviter d'y avoir recours.
- **IN**
  - L'information est **inconnue**.
  - Dans ce cas, l'annulabilité pourrait être légitime ; la question est de savoir
    - comment la représenter pour que cela pose le moins de problèmes possible ;
    - s'il est nécessaire de conserver la raison pour laquelle elle n'est pas connue.
- **X**
  - L'information n'est **pas accessible**.
    - **De façon transitoire (donc à court terme)** :
      dans ce cas, le gestionnaire transactionnel rend inutile l'utilisation de l'annulabilité.
    - **De façon durable (donc à moyen ou long terme)** :
      équivalent à **IN**.

Données absentes – Impact sur la logique des modèles relationnels

- On en conclut que
  - le cas X, qui se réduit en IN ou NA, peut être ignoré ;
  - le cas NA, *pourrait* être ignoré, si la modélisation est (toujours) adéquate.
- Que faire du cas IN ?
  - Plusieurs théoriciens, dont Date, ont décidé de ne pas intégrer ce cas au modèle relationnel et de le traiter aussi par la modélisation, préservant ainsi la logique classique, donc bi-valuée (faux, vrai).
  - Les membres du Comité de normalisation du langage SQL ont décidé d'intégrer ce cas à leur modèle et, conséquemment, d'utiliser une non classique logique tri-valuée (faux, vrai, inconnu).

Données absentes – Solutions SANS annulabilité (Date et cie)

- Principes
  - Séparer les propositions complètes des incomplètes.
  - Conserver les causes d'absence séparément.

- Pour un inventaire des techniques de modélisation
  - http://www.dcs.warwick.ac.uk/~hugh/TTM/Missing-info-without-nulls.pdf

Données absentes – Solutions AVEC annulabilité (SQL et cie)

- Trois étapes à faire dans l'ordre :
  - si possible, corriger cette lacune à la source (dans la réalité) ;
  - si possible, modifier le modèle conceptuel pour en tenir compte ;
  - sinon, introduire le concept d'annulabilité dans la théorie relationnelle, ce qui induit le recours
    - à l'un des deux artifices suivants :
      - un marqueur NUL (une propriété ajoutée à tous les attributs) ou
      - une valeur NULLE (ajoutée à tous les types de base).
    - **et** à une logique à 3 valeurs
      - afin, notamment, de pouvoir définir l'égalité (opération essentielle aux opérations d'affectation, de restriction, de jointure, d'union, etc.)
  - remarquer que si la cause de l'annulabilité doit être conservée, il faut utiliser la solution de Date de toute façon !

Données absentes – Exemple de logique à 3 valeurs

V : Vrai
F : Faux
I : Inconnu

A	B	$A \vee B$	$A \wedge B$	$\neg A$
V	V	V	V	F
V	I	V	I	F
V	F	V	F	F
I	V	V	I	I
I	I	I	I	I
I	F	I	F	I
F	V	V	F	V
F	I	I	F	V
F	F	F	F	V

Données absentes
La solution SQL :
P3 ou B3 ?

CHECK
- **satisfait ssi**
  - **true ou unknown**
- **comme P3 !**

WHERE
- **satisfait ssi**
  - **true**
- **comme B3 !**

Et la cohérence ?

OR	true	unknown	false
true	true	true	true
unknown	true	unknown	unknown
false	true	unknown	false

AND	true	unknown	false
true	true	unknown	false
unknown	unknown	unknown	false
false	false	false	false

P	NOT P
true	false
unknown	unknown
false	true

IS	true	unknown	false
true	true	false	false
unknown	false	true	false
false	false	false	true

Données absentes – Remarques

- Codd, dans son deuxième modèle, a proposé d'introduire deux marqueurs (IN et NA), ce qui induit une logique à 4 valeurs.
- D'autres chercheurs ont proposé d'autres systèmes logiques à 4 et même 5 valeurs.
- Humblement, nous nous en tiendrons, le plus souvent, à la logique classique dans le cadre du présent cours, d'autant que, le plus souvent, en pratique, il faut conserver la cause de l'annulabilité.

Données absentes
Problème réglé ?

Et non,
le problème n'est pas résolu !

Nous y reviendrons donc dans le module BD028

De: oracle-acct_ww@oracle.com
Objet: Nom d'utilisateur de votre compte Oracle
Date: 2 octobre 2014 19:44
À: luc.lavoie@usherbrooke.ca

ORACLE

Cher/Chère NULL !,

Vous avez demandé à recevoir par email le nom d'utilisateur de votre compte Oracle.

Votre nom d'utilisateur est : **luc.lavoie@usherbrooke.ca**

Merci !

L'équipe de gestion des comptes Oracle

Mettez votre compte à jour :

- Abonnez-vous aux communications dédiées aux thèmes qui vous intéressent.
- Devenez membre des communautés Oracle.
- Pour modifier votre adresse email, votre mot de passe ou toute autre information de votre compte, cliquez sur le lien Compte en haut des pages Oracle.com.

Obtenir de l'aide

- Des questions ? Aide (page Account Help)
- Se connecter
  - Envoyer une demande d'aide
  - profilehelp_ww@oracle.com

Hardware and Software
ORACLE
Engineered to Work Together

Aide (page Account Help) | Ne pas envoyer d'email | Mentions légales | Conditions d'utilisation | Confidentialité

Références (1)

- Une théorie (mathématique) est un ensemble d'affirmations dont certaines sont des axiomes et les autres des théorèmes démontrables à partir de ces axiomes et au moyen de règles d'inférence (exprimée à l'aide de la) logique.
  - http://fr.wikipedia.org/wiki/Théorie
- Un modèle est une représentation conforme à une théorie.
  - http://fr.wikipedia.org/wiki/Modèle
- Un langage (formel) est un formalisme permettant de décrire des propositions sémantiquement interprétables en termes d'un modèle.
  - http://fr.wikipedia.org/wiki/Langage_formel
- Une théorie peut être à l'origine de plusieurs modèles, un modèle de plusieurs langages, un langage de plusieurs dialectes.
- Pour en savoir plus sur le calcul des prédicats :
  - http://fr.wikipedia.org/wiki/Calcul_des_prédicats

Références (2)

- Théorie relationnelle
  - E.F. Codd (1990) :
    The Relational Model for Database Management: Version 2.
    Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc.
  - C.J. Date, H. Darwen (2007) :
    Databases, types and the relational model: the third manifesto.
    Reading, Mass.: Addison-Wesley.
  - F. de Sainte Marie (2013) :
    Bases de données relationnelles et normalisation : de la première à la sixième forme normale.
    https://fsmrel.developpez.com/basesrelationnelles/normalisation/
  - H. Darwen (2006) :
    How To Handle Missing Information Without Using NULL.
    http://www.dcs.warwick.ac.uk/~hugh/TTM/Missing-info-without-nulls.pdf

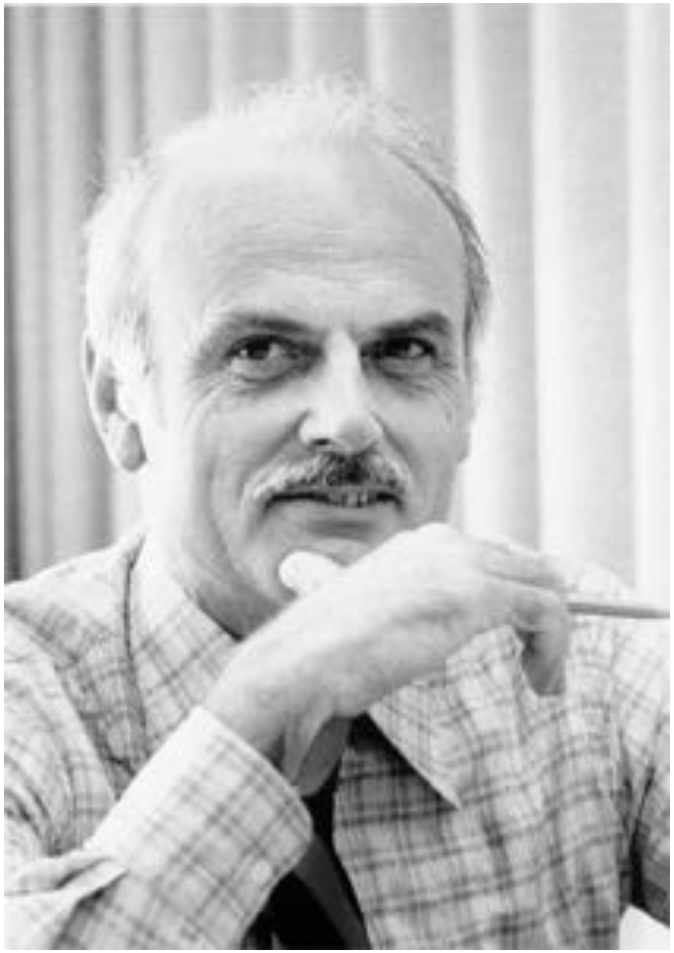
Références (3)

- Théorie des Types
  - Luca Cardelli, Peter Wegner (1985) : *On understanding types, data abstraction, and polymorphism.* ACM Computing Surveys 17, 4; pp. 471–523. DOI:http://dx.doi.org/10.1145/6041.6042
  - Chris J. Date (2016) : *Type inheritance and relational theory: subtypes, supertypes, and substitutability*. O'Reilly Media, Sebastopol, CA. ISBN:978-1-4919-5999-2
- Manuels classiques
  - [C. J. Date 2004], chapitre 3.
  - [Elmasri and Navathe 2004], chapitre 4.
  - [Elmasri and Navathe 2011], chapitre 3.
  - [Elmasri and Navathe 2016], chapitre 8.
  - [Ullman and Widom 2008], chapitre 3.

Les colles du prof

- Quelles différences existe-t-il entre
  - un type de base et sous-type ?
  - un tuple et une relation ?
  - une relation et une variable de relation ?
  - un modèle logique et une base de données ?
  - une théorie et un modèle ?
  - un modèle et un langage ?
  - une clé candidate et une superclé ?
- Quelles sont les opérations de base proprement relationnelles ?
- En quoi se distinguent-ils des opérations ensemblistes ?

Edgar Frank Codd et Christopher J. Date



https://en.wikipedia.org/wiki/Edgar_F._Codd

Photo of Chris Date by Douglas Robertson, Edinburgh

https://en.wikipedia.org/wiki/Christopher_J._Date