# BDW - Calculs relationnels

Fabien Duchateau

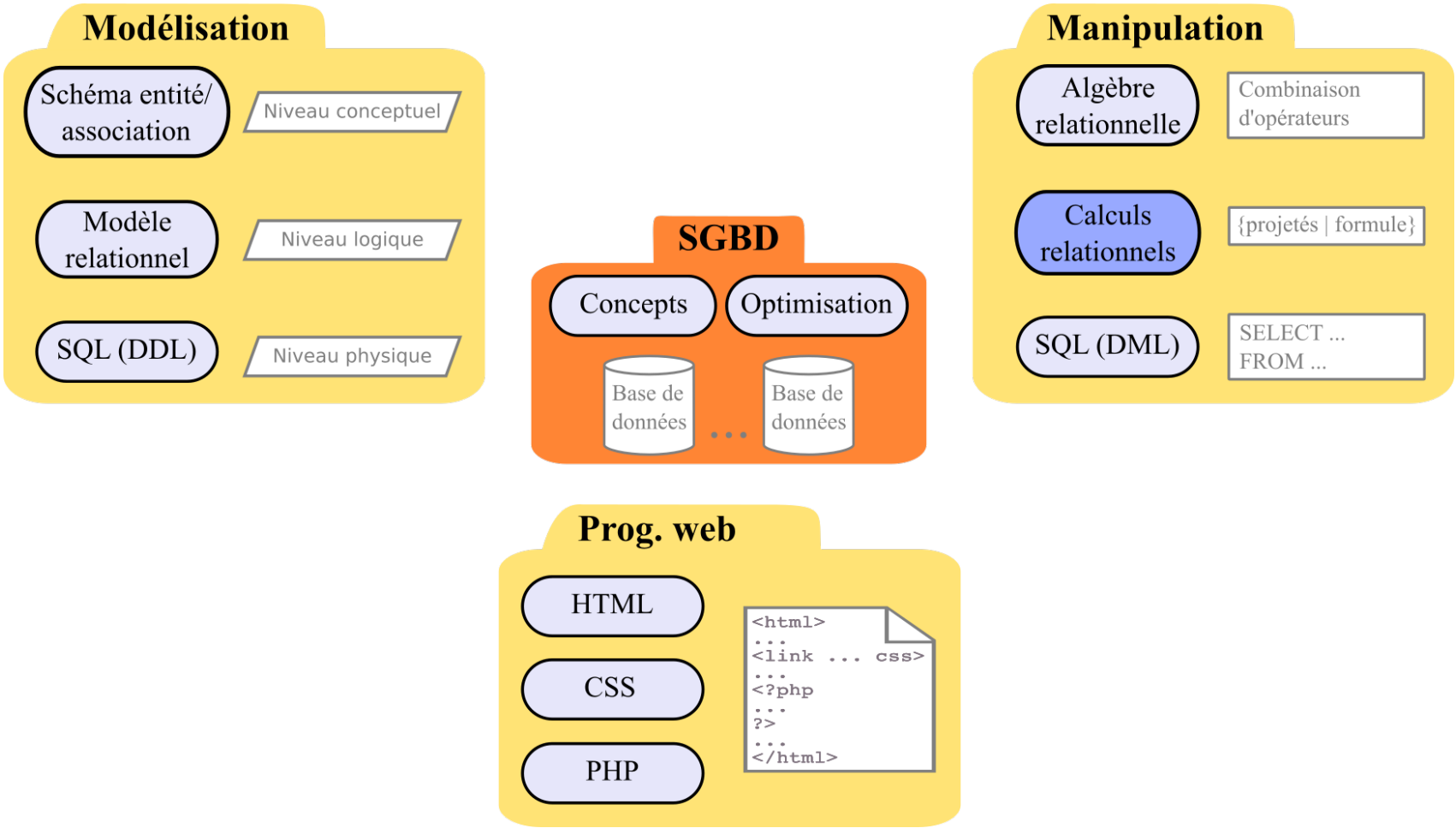
*fabien.duchateau [at] univ-lyon1.fr*

Université Claude Bernard Lyon 1

2023 - 2024


https://perso.liris.cnrs.fr/fabien.duchateau/BDW/

# Positionnement dans BDW




---

Ces diapositives utilisent le genre féminin (e.g., chercheuse, développeuses) plutôt que l'écriture inclusive (moins accessible, moins concise, et pas totalement inclusive)

# Langages de manipulation de données

Pourquoi des langages de manipulation "théoriques" ?

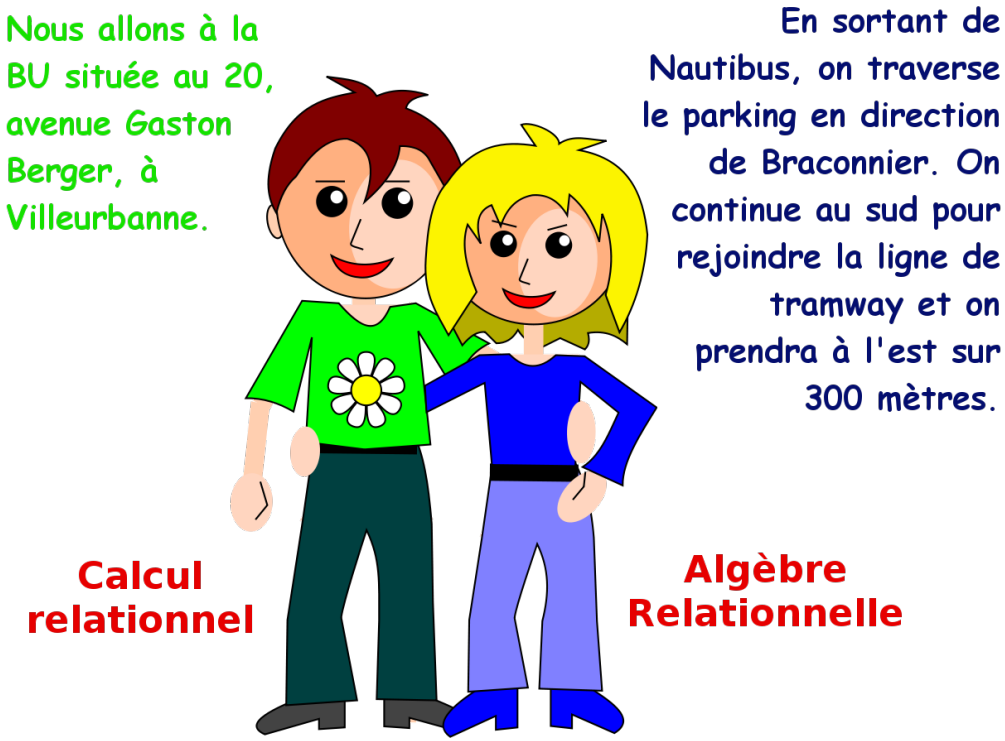
- Étudier le modèle relationnel
- Prouver des propriétés
- Obtenir une mise en œuvre efficace
- Servir de base à des langages plus conviviaux pour les utilisatrices

Deux langages à la puissance d'expression équivalente :

- Algèbre relationnelle (AR)
- Calculs relationnels (CR)

# Langages de manipulation de données (2)

- Calcul relationnel : décrit le résultat à obtenir (langage déclaratif)
- Algèbre relationnelle : exprime comment obtenir le résultat (langage procédural)




http://openclipart.org

# Plan

# Intérêts des calculs relationnels (CR)

- Contrairement à l'AR, une requête CR fait abstraction des mécanismes utilisés par le SGBD pour obtenir le résultat

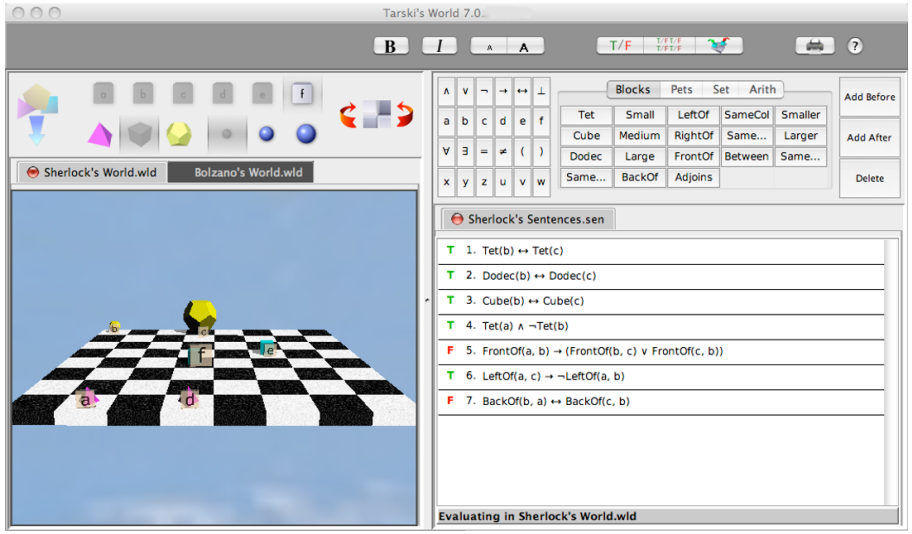
- Des non-informaticien.ne.s peuvent donc écrire des requêtes en décrivant ce qu'elles veulent obtenir

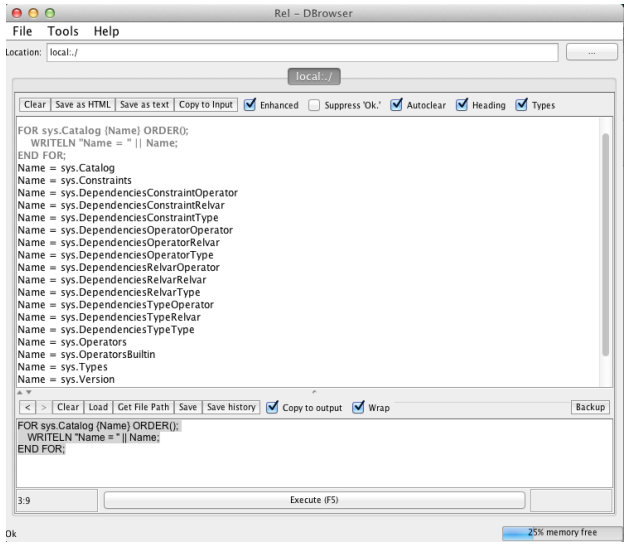
---

Kuhns, John. Logical aspects of question-answering by computer. Software Engineering : COINS (1969)

# Outils utilisant le CR

- QUEL (Ingres)
- D (spécifications) et D4
- Dataphor
- Rel
- Tarsky' world




http://reldb.org/
https://en.wikipedia.org/wiki/Dataphor
https://en.wikipedia.org/wiki/Tarski's_World
https://en.wikipedia.org/wiki/QUEL_query_languages
https://en.wikipedia.org/wiki/D_%28data_language_specification%29

# Principe du CR

Deux calculs :

- Calcul relationnel à variable tuple (CRVT), dans lequel les variables manipulées représentent des n-uplets
- Calcul relationnel à variable domaine (CRVD), dans lequel les variables manipulées représentent des attributs

---

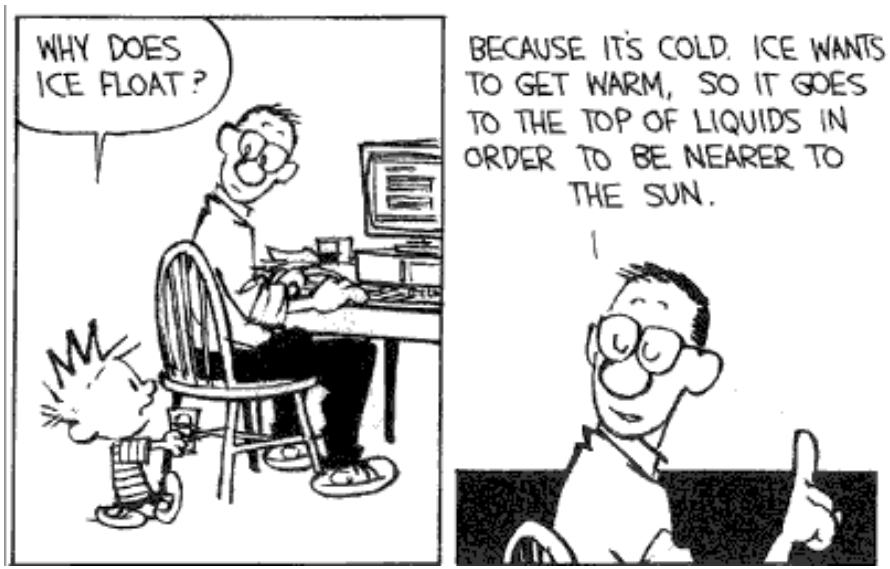
Codd, Edgar. A relational model of data for large shared data banks. Communications of the ACM (1970)

Lacroix, M., Pirotte, A. Domain-oriented relational languages. Proceedings of VLDB (1977)

https://en.wikipedia.org/wiki/Relational_calculus

# Principe du CR (2)

- CR basé sur la **logique du premier ordre**
- Caractériser le résultat d'une requête qui rend vraie une formule



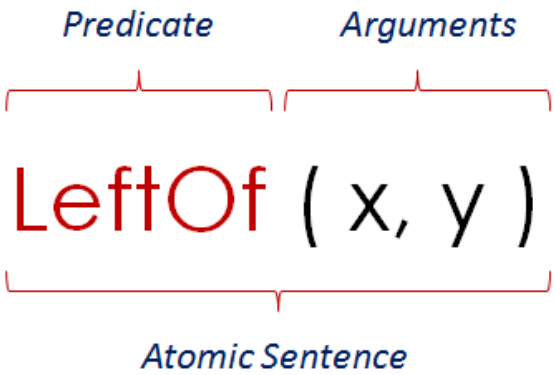

Calvin et Hobbes (Bill Watterson)

# Plan

# Logique du premier ordre

Logique du premier ordre, ou calcul des prédicats :

- Des variables
- Des formules et des prédicats
- Des connecteurs logiques $\vee$, $\wedge$, $\neg$, $\rightarrow$
- Deux quantificateurs $\exists$, $\forall$




http://fr.wikipedia.org/wiki/Calcul_des_pr%C3%A9dicats

# Définitions

- Variables ($x$, $y$, $z$, ...)
- Constantes ($a$, $b$, $c$, ..., $1$, $2$, ..., 'toto', 'titi', ...)
- Domaine = ensemble des valeurs possibles d'une variable
- Symboles de fonctions, avec leur arité ($f/2$, $g/4$, $+/2$, ...)

*joindre*$(x, y)$ est une fonction d'arité 2, avec $x$ et $y$ des variables

# Atome

- Soient $x$ et $y$ des variables et $c$ une constante
- Soit $\Theta$ un opérateur parmi $=$, $\neq$, $<$, $>$, $\leq$, $\geq$
- $x \Theta y$ et $x \Theta c$ sont des atomes

$x < 2$ et $x = b$ sont des atomes

# Formule

Une formule se définit récursivement comme :

- Un atome
- $A \wedge B$ avec $A$ et $B$ des formules
- $A \vee B$ avec $A$ et $B$ des formules
- $A \rightarrow B$ avec $A$ et $B$ des formules
- $\neg A$ avec $A$ une formule
- $\forall x A(x)$ avec $A$ une formule
- $\exists x A(x)$ avec $A$ une formule

$(y < 10) \wedge (y \times 3 \geq 15)$ est une formule

# Formule (2)

Évaluation d'une formule :

- On évalue les arguments (valeurs dans le domaine)
- Résultat booléen (vrai ou faux) calculé en fonction de ces valeurs

Priorité des opérateurs (en débutant par le plus prioritaire) :
$\{\neg\}$, $\{\wedge, \vee\}$, $\{\rightarrow\}$, $\{\forall, \exists\}$

Soit la formule $f(y) := (y < 10) \wedge (y \times 3 \geq 15)$, f(7) est vraie et f(2) est fausse

# Formule (3)

- $A \wedge B$ s'évalue à vrai si $A$ s'évalue à *vrai et* $B$ s'évalue à *vrai*
- $A \vee B$ s'évalue à vrai si $A$ s'évalue à *vrai ou* si $B$ s'évalue à *vrai*
- $A \rightarrow B$ s'évalue à vrai si $A$ s'évalue à *faux ou* si $B$ s'évalue à *vrai*
- $\neg A$ s'évalue à vrai si $A$ s'évalue à faux
- $\forall x A(x)$ ("pour tout $x$") s'évalue à *vrai* si pour toutes les valeurs possibles de $x$, $A$ s'évalue à *vrai*
- $\exists x A(x)$ ("il existe $x$ tel que") s'évalue à *vrai* si on peut trouver une valeur pour $x$ telle que $A$ s'évalue à *vrai*

# Prédicat

- Convention (pour ce cours) : si une variable est introduite par un quantificateur, elle ne peut apparaître que dans la formule sous ce quantificateur
- Variable libre = variable **non** introduite par un quantificateur
- Variable liée = variable introduite par un quantificateur

$A(x) \wedge (\forall y\ B(y)) \wedge C(z)$
Les variables $x$ et $z$ sont libres, et la variable liée $y$ peut apparaître uniquement dans la formule $B$

# Prédicat (2)

Un prédicat est une formule qui possède (au moins) une variable libre

On peut chercher les valeurs des variables libres d'un prédicat qui évaluent à vrai ce prédicat

On peut donc imaginer représenter les relations d'une base de données par des prédicats

# Exemple : du français à la formule

Relation :

ÉLÈVE (idE, *nomE*, *moyenneLycée*, *effectifLycée*)

Prédicat : Élève(i, n, m, e)

**Existe t-il un lycée avec un effectif supérieur à 500 ?**

$Eleve(i, n, m, e) \wedge e > 500$

# En résumé

Logique du premier ordre :

- Atomes (opérateur de comparaison entre variables et/ou constantes)
- Formule (un atome ou une combinaison de formules via des connecteurs logiques)
- Prédicat (formule avec au moins une variable libre)

**Une relation de BD peut être représentée comme un prédicat**

# Plan

# Requête en CR

Il est possible d'utiliser une formule pour spécifier une requête dans une base de données. Mais nos prédicats retournent un booléen !

**Syntaxe d'une requête CR :**

$$\{v_1, \ldots, v_n \mid F\}$$

où $F$ est une formule, et $v_1, \ldots, v_n$ sont des variables libres représentant les attributs projetés (i.e., du résultat)

Deux types de calcul relationnel :

- à variable tuple (CRVT)
- à variable domaine (CRVD)

# Requête en CRVT

En **CR à variable tuple**, les variables manipulées dans les prédicats représentent des **n-uplets**

Pour chaque relation $R(a_1, \ldots, a_n)$ de la base :

- La relation $R$ est traduite par un prédicat unaire $R/1$
- Les arguments de $R/1$ sont des n-uplets de la forme $x^{(a_1,\ldots,a_n)}$
- Étant donnée une instance de $R$ :
  - l'instance est un ensemble de n-uplets $\{t_1, \ldots, t_k\}$, *i.e.* chaque $t_i$ est de la forme $(v_1^i, \ldots, v_n^i)$
  - $R(x)$ s'évalue à vrai si la valeur de $x$ est l'un des n-uplets $t_i$

https://en.wikipedia.org/wiki/Tuple_relational_calculus

# Requête en CRVT (2)

Soit le prédicat $R(x)$, on accède aux attributs avec la notation pointée (e.g., x.att$_1$, x.att$_n$)

- $R(a)$ indique que l'on définit une variable $a$ pour représenter les n-uplets de la relation $R$
- $a$ peut "prendre" les valeurs *(INSA, Lyon, 36000)* ou *(UJF, Grenoble, 10000)*
- *a.nomU* est l'attribut *nomU* des instances représentées par la variable $a$

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table UNIVERSITÉ

# Requête en CRVT (3)

**Syntaxe d'une requête CRVT :**

$$\{x_1.a_1, \ldots, x_1.a_n, \ldots, x_k.a_1, \ldots, x_k.a_p \mid F\}$$

où $F$ est une formule dont les variables libres $x_1, \ldots, x_k$ représentent les n-uplets des relations $R_1, \ldots, R_k$

La formule $F$ est composée d'une combinaison :

- D'atomes $x_i.a_g \Theta x_j.a_h$ ou $x_i.a_g \Theta c$ (avec $c$ une constante)
- De formules connectées $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$, $\neg F_1$ (avec $F_1$ et $F_2$ des formules)
- De formules introduites par un quantificateur $\exists x_i(F_1)$, $\forall x_i(F_1)$ (avec $F_1$ et $F_2$ des formules)

# Exemples de requête en CRVT

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table UNIVERSITÉ

## Le nom et la ville des universités

- *u* est un n-uplet de la relation UNIVERSITÉ : **Université(u)**
- *u.nomU* et *u.ville* représentent les attributs *nomU* et *ville* : **{u.nomU, u.ville | Université(u)}**

# Exemples de requête en CRVT (2)

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table UNIVERSITÉ

| nomU |
|---|
| UJF |

**Le nom des universités avec un effectif inférieur à 12000**

- $u$ est un n-uplet de la relation UNIVERSITÉ : **Université(u)**
- la contrainte sur l'effectif est : **u.effectif** $<$ **12000**
- requête : $\{$**u.nomU** $\mid$ **Université(u)** $\wedge$ **u.effectif** $<$ **12000** $\}$

# Exemples de requête en CRVT (3)

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table UNIVERSITÉ

| ville | département |
|---|---|
| Paris | 75 |
| Lyon | 69 |
| Grenoble | 38 |

Table VILLEDEPT

## Le nom des universités et leur département

- $u$ est un n-uplet de la relation UNIVERSITÉ : **Université(u)**
- $v$ est un n-uplet de la relation VILLEDEPT et contrainte de jointure : **VilleDept(v)** $\wedge$ **u.ville = v.ville**
- requête : { **u.nomU, v.département** | **Université(u)** $\wedge$ **VilleDept(v)** $\wedge$ **u.ville = v.ville** }

| nomU | département |
|---|---|
| INSA | 69 |
| UCB | 69 |
| UJF | 38 |

# Exemples de requête en CRVT (4)

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table UNIVERSITÉ

| ville | département |
|---|---|
| Paris | 75 |
| Lyon | 69 |
| Grenoble | 38 |

Table VILLEDEPT

## Le nom des universités du département 38

$$\{ \text{r1.nomU} \mid \text{UNIVERSITÉ}(\text{r1}) \wedge \exists\ \text{r2}\ (\text{VILLEDEPT}(\text{r2}) \wedge \text{r1.ville} = \text{r2.ville} \wedge \text{r2.département} = 38) \}$$

| nomU |
|---|
| UJF |

# Autre notation pour le CRVT

En CRVT, on utilise parfois la position dans les n-uplets au lieu des attributs :

- $t^{(n)}$ au lieu de $t^{(a_1,\ldots,a_n)}$
- $t.n$ au lieu de $t.a_n$

On utilise parfois des $[\ ]$ au lieu du .

- $t[a_i]$ ou $t[i]$ au lieu de $t.a_i$

# Requête en CRVD

En **CR à variable domaine**, les variables manipulées dans les prédicats représentent des **domaines de définition d'attributs**

Pour chaque relation $R(a_1, \ldots, a_n)$ de la base :

- La relation $R$ est traduite par un prédicat n-aire $R/n$
- Les arguments de $R/n$ sont de la forme $R(a_1 : v_1, \ldots, a_n : v_n)$
  - les variables $v_1, \ldots, v_n$ représentent les domaines des attributs $a_1, \ldots, a_n$
  - $R(a_1 : v_1, \ldots, a_n : v_n)$ s'évalue à vrai s'il existe un n-uplet dont les attributs $a_1, \ldots, a_n$ satisfont les variables $v_1, \ldots, v_n$

https://en.wikipedia.org/wiki/Domain_relational_calculus

# Requête en CRVD (2)

Accès direct aux attributs !

- $R(nomU : n, ville : v, effectif : e)$ indique que l'on définit une variable $n$ pour représenter le domaine de l'attribut *nomU*, une variable $v$ pour le domaine de l'attribut *ville*, etc.
- $n$ peut "prendre" les valeurs *INSA* ou *UJF*

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table Université

# Exemples de requête en CRVD

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table UNIVERSITÉ

## Le nom et la ville des universités

- *n* est le domaine de l'attribut *nomU* : **Université(nomU : n)**
- *v* est le domaine de l'attribut *ville* : **Université(nomU : n, ville : v)**
- requête : **{n, v | Université(nomU : n, ville : v)}**

# Exemples de requête en CRVD (2)

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table UNIVERSITÉ

## Le nom des universités avec un effectif inférieur à 12000

- *n* et *e* sont les domaines des attributs *nomU* et *effectif* : **Université(nomU : n, effectif : e)**
- la contrainte sur l'*effectif* est : $\mathbf{e < 12000}$
- requête : $\{\mathbf{n} \mid \exists\ \mathrm{e}\ (\mathbf{Université(nomU : n, effectif : e)} \wedge \mathbf{e<12000})\}$

| nomU |
|---|
| UJF |

# Exemples de requête en CRVD (3)

| nomU | ville | effectif |
|---|---|---|
| INSA | Lyon | 36000 |
| UCB | Lyon | 15000 |
| UJF | Grenoble | 10000 |
| UJM | Saint-Étienne | 21000 |

Table UNIVERSITÉ

| ville | département |
|---|---|
| Paris | 75 |
| Lyon | 69 |
| Grenoble | 38 |

Table VILLEDEPT

## Le nom des universités et leur département

$\{ n, d \mid \exists\, v_1$ UNIVERSITÉ(nomU : n, ville : $v_1$)
$\wedge\ \exists\, v_2$ VILLEDEPT(ville : $v_2$, département : d)
$\wedge\ v_1 = v_2 \}$

| nomU | département |
|---|---|
| INSA | 69 |
| UCB | 69 |
| UJF | 38 |

# En résumé

Calculs relationnels :

- Expriment ce que l'on veut obtenir
  - CRVT : les variables sont des n-uplets
  - CRVD : les variables sont des domaines d'attributs
- Basés sur la logique du premier ordre
- Requêtes = variables libres (attributs projetés) et une formule $\{v_1, \ldots, v_n \mid F\}$

# Plan

# Requêtes "sûres"

Soient des relations $R$ et $S$, et une constante $c$ :

$$\{t \mid \neg(R(t))\}$$

$$\{t \mid R(t) \wedge \exists s(s.att \neq c)\}$$

$$\{t \mid S(t') \wedge \forall t'.att(R(t) \wedge t.att = t'.att)\}$$

Ces requêtes ne sont pas sûres :

- On ne sait pas où chercher les n-uplets pour la variable $t$ (première requête) ou pour la variable $s$ (seconde requête)
- $R(t)$ dépend du domaine de $t'.att$, qui est potentiellement infini (troisième requête)

# Requêtes "sûres" (2)

Les domaines dans lesquels les variables prennent leur valeur sont potentiellement infinis (e.g., les entiers, les réels)

La caractérisation de requêtes sûres est complexe (les SGBDs obligent l'utilisatrice à spécifier le domaine des variables, ce qui permet d'éviter les requêtes non sûres)

Pour obtenir une requête sûre, il faut réduire son contexte au domaine actif

# Domaine actif

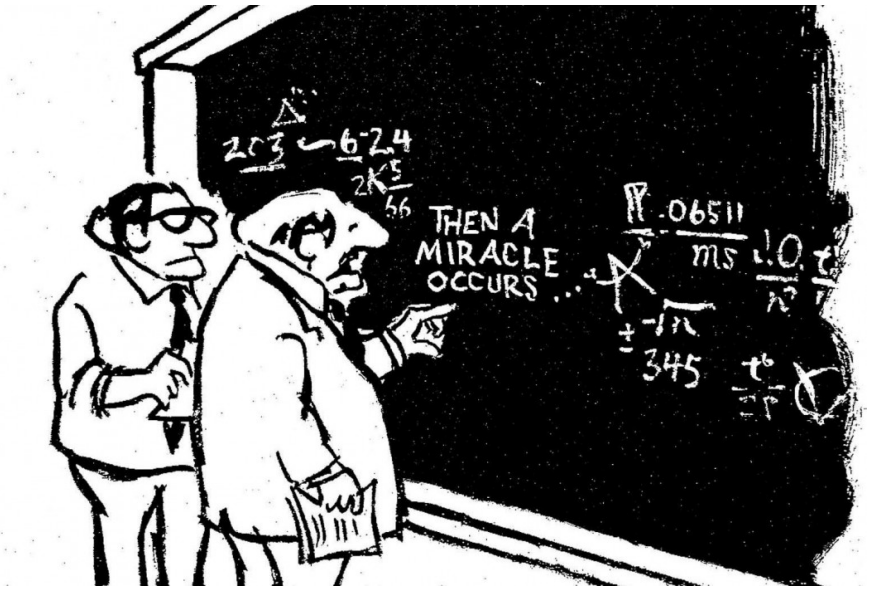
Pour une requête $\mathcal{Q}$ et une instance de base de données $\mathcal{I}$, le domaine actif $Dom(\mathcal{Q}, \mathcal{I})$ se définit comme l'ensemble des constantes qui apparaissent dans $\mathcal{Q}$ et dans $\mathcal{I}$

Une requête sûre possède une formule :

- Dont les réponses contiennent uniquement des valeurs de $Dom(\mathcal{Q}, \mathcal{I})$
- Dont les expressions $\exists t(R(t))$ sont vraies pour des n-uplets $t$ inclus dans $Dom(\mathcal{Q}, \mathcal{I})$
- Idem pour $\forall t(R(t))$

# Équivalence AR/CR

Équivalence AR/CR : toute requête exprimée en AR peut être traduite en CR. Inversement, toute requête sûre exprimée en CR peut être traduite en AR.




Codd, E. F. Relational completeness of data base sublanguages. Data Base Systems (1972)

# Traduction de requêtes AR/CR

Traduction des opérations fréquentes :

- $\Pi_{att_1,\ldots,att_n}(R) \equiv \{t.att_1, \ldots, t.att_n \mid R(t)\}$
- $\sigma_C(R) \equiv \{t \mid R(t) \wedge C'(t)\}$ avec $C'$ une formule obtenue en remplaçant les attributs $att$ de $C$ par $t.att$
- $R \cup S \equiv \{t \mid R(t) \wedge S(t)\}$
- $R \setminus S \equiv \{t \mid R(t) \wedge \neg(S(t))\}$
- $R \times S \equiv \{t, t' \mid R(t) \wedge S(t')\}$

# En résumé

Les langages théoriques AR et CR :

- Ont la même puissance d'expression
- Servent de support aux langages comme SQL

**Prochain cours : SQL comme langage d'interrogation**