Calcul & algèbre relationnelle

Implication et Division relationnelle

Alexis NEDELEC

Centre Européen de Réalité Virtuelle
Ecole Nationale d'Ingénieurs de Brest

Introduction

Modèle relationnel

- Calcul relationnel :
  - modéliser les requêtes sans décrire l'ordre d'éxécution.
- Algèbre relationnelle :
  - modéliser les requêtes que SQL pourra éxécuter.

Formulation logique

- Calcul relationnel : $R = \{s \mid s \in S \wedge p(s)\}$
- Opérateur relationnel : $\sigma_{[p(s)]}(S) = \{s \mid s \in S \wedge p(s)\}$

où :

- s : tuple, enregistrement, élément d'un ensemble.
- $p(s)$: expression, proposition, formule logique.

Introduction

Algèbre relationnelle : 5 opérateurs de base

- **Projection** :
  - $\Pi_{(X)}(R) = \{r(X) \mid r \in R\}$

  où $r(X)$: valeurs du tuple r pour les attributs X de R
- **Restriction** :
  - $\sigma_{[p(r)]}(R) = \{r \mid r \in R \wedge p(r)\}$
- **Produit cartésien**, $T = \times(R, S)$:
  - $T = \{t \mid \forall r \in R, \forall s \in S, \exists t \in T \wedge \Pi_{(R)}(t) = r \wedge \Pi_{(S)}(t) = s\}$
- **Union** :
  - $\cup(R, S) = \{t \mid t \in R \vee t \in S\}$
- **Différence** :
  - $\setminus(R, S) = \{t \mid t \in R \wedge t \notin S\}$

Logique $(\neg, \forall, \exists, \wedge, \vee)$

Equivalences loqigues

$$\begin{array}{lcl} \forall x\ p(x) & \iff & \neg(\exists x\ \neg p(x)) \\ \forall x \exists y\ p(x,y) & \iff & \neg(\exists x\ \neg(\exists y\ p(x,y))) \\ p \Rightarrow q & \iff & \neg p \vee q \end{array}$$

Implication : $p \Rightarrow q$

Condition nécessaire mais non-suffisante : SI p ALORS q

p	q	$\neg p \vee q$
0	0	1
0	1	1
1	0	0
1	1	1

Ex : "Pour avoir le diplôme ENIB (proposition p) il faut avoir passé 5 années d'étude (proposition q)"

Quantificateur universel : "Quel que soit"

"Récupérer les bars qui servent toutes les bières"

Quelque soit la bière ($\forall bi$) de la Base de Données, elle doit être servie ($services(bi, ba)$) dans les bars (ba) en question

Modélisation en calcul relationnel

$$\{ba \mid ba \in bars \\ \wedge(\forall bi \in bieres\ ,\ \exists s \in services(bi, ba)) \\ \wedge\ s(id\_bar) = ba(id\_bar) \wedge s(id\_biere) = bi(id\_biere)\}$$

Equivalence logique : $\forall x \exists y\ p(x, y) \iff \neg(\exists x\ \neg(\exists y\ p(x, y)))$

$$\forall bi \exists ba\ services(bi, ba) \iff \neg(\exists bi\ \neg(\exists ba\ services(bi, ba))) \\ \{ba \mid ba \in bars \\ \wedge(\neg(\exists bi \in bieres\ ,\ \neg(\exists s \in services(bi, ba))) \\ \wedge\ s(id\_bar) = ba(id\_bar) \wedge s(id\_biere) = bi(id\_biere))\}$$

Quantificateur universel : "Quel que soit"

$\{ba \mid ba \in bars$

```
SELECT *
FROM bars ba WHERE
```

$\wedge\ (\neg(\exists bi \in bieres\ ,\ \neg(\exists s \in services(bi, ba)))$

```
NOT EXISTS(SELECT *
           FROM bieres bi
           WHERE NOT EXISTS(SELECT *
                            FROM services s
```

$\wedge\ s(id\_bar) = ba(id\_bar) \wedge s(id\_biere) = bi(id\_biere)\ )\ \}$

```
                            WHERE s.id_bar=ba.id_bar
                             AND s.id_biere=bi.id_biere))
```

Quantificateur universel : "Quel que soit"

Formulation SQL "générique" de la

```
SELECT  re.X
FROM R re WHERE
```

division relationnelle : $T[X] = R[X, Y] \div S[Y]$

```
NOT EXISTS (SELECT *
            FROM S
            WHERE NOT EXISTS (SELECT *
                              FROM R ri
                              WHERE ri.X=re.X
                                AND S.Y=ri.Y
                             )
           );
```

Quantificateur universel : "Quel que soit"

division relationnelle : $T[X] = R[X, Y] \div S[Y]$

```
SELECT re.X
FROM R re
WHERE NOT EXISTS ( (SELECT Y FROM S)
                     EXCEPT
                     (SELECT Y FROM R ri WHERE ri.X=re.X)
                   );
```

Formulation algébrique : $T[X] = R[X, Y] \div S[Y]$

$$T = \div(R, S) = \{t \mid \forall s \in S \ , \ (t, s) \in R\}$$

Division relationnelle : Quel que soit les éléments d'un ensemble donnée ils sont liés aux éléments que je cherche à récupérer.

Implication : "SI ... ALORS"

"bars servant les m̂ bières qu'au 'Bar du Coin' "

Quelque soit la bière ($\forall bi$) servie au 'Bar du Coin' elle doit-être servie dans le bar en question,

SI une bière (bi) est servie (s1) au 'Bar du Coin'
ALORS elle sera servie (s2) dans le bar (ba) en question.

Implication ($p \Rightarrow q$) :

- proposition p : services(bi,bar='Bar du Coin')
- proposition q : services(bi,ba)

Traduction en calcul relationnel : $\forall x\ r(x)$ (r : implication)

$$\{ba \mid ba \in bars$$
$$\wedge \forall s1(s1 \in services(bi, bar = \text{'Bar du Coin'}) \Rightarrow \exists s2 \in services(bi, ba))\}$$

Implication : "SI ... ALORS"

Equivalence logique : $p \Rightarrow q \iff \neg p \vee q$

$\forall s1(s1 \in services(bi, bar =$'Bar du Coin'$) \Rightarrow \exists s2 \in services(bi, ba))$

$\forall s1(s1 \notin services(bi, bar =$'Bar du Coin'$) \vee \exists s2 \in services(bi, ba))$

Equivalence logique : $\forall x\ p(x) \iff \neg(\exists x\ \neg p(x))$

$\forall s1(s1 \notin services(bi, bar =$'Bar du Coin'$) \vee \exists s2 \in services(bi, ba))$

$\neg(\exists s1 \in services(bi, bar =$'Bar du Coin'$) \wedge \neg(\exists s2 \in services(bi, ba)))$

Reformulation en calcul relationnel

$\{ba \mid ba \in bars$

$\wedge \neg(\exists s1 \in services(bi, bar =$'Bar du Coin'$) \wedge \neg(\exists s2 \in services(bi, ba)))$

$\wedge\ s1(id\_biere) = s2(id\_biere) \wedge s2(id\_bar) = ba(id\_bar)\ \}$

"bars servant les m̂ bières qu'au 'Bar du Coin'"

Traduction en SQL

```
SELECT *
FROM bars ba
WHERE NOT EXISTS(SELECT *
            FROM services s1
            WHERE s1.id_bar IN(SELECT id_bar
                               FROM bars
                               WHERE  bar='Bar du Coin')
            AND NOT EXISTS(SELECT *
                           FROM services s2
                           WHERE s2.id_bar = ba.id_bar
                           AND s1.id_biere = s2.id_biere));
```

Algèbre Relationnelle

Opérateurs de base pour récupérer

- Un ensemble : Π (`SELECT`), $\times$ (`FROM`), σ (`WHERE`)
- Union, Différence entre ensembles : $\cup, \setminus$

Opérateurs dérivés

- Jointure entre ensembles ($\bowtie$) : combinaison de ($\times$ et σ)
- Intersection de deux ensembles ($\cap$) : déduite de ($\cup$ et $\setminus$)
  - $\cap(R,S) = \setminus(R, \setminus(R,S))$
  - $\cap(R,S) = \setminus(S, \setminus(S,R))$
  - $\cap(R,S) = \setminus(\cup(R,S), [(\cup(\setminus(R,S), \setminus(S,R))])$
- Division relationnelle ($\div$) : construite à partir de (Π, $\times$ et $\setminus$)
  - $\div(R,S) = \setminus(\ \Pi_{(X)}(R)\ ,\ \Pi_{(X)}(\ \setminus(\ \times(\Pi_{(X)}(R)\ ,\ S\ ), R\ )\ )$

Division relationnelle

Formulation algébrique

Soient deux relations R, S :

- de schéma : $att(S) \subset att(R)$
- avec $X = \backslash(att(R), att(S))$

$T[X] = \div(R, S) = \backslash(\ \Pi_{(X)}(R)\ ,\ \Pi_{(X)}(\ \backslash(\ \times(\Pi_{(X)}(R)\ ,\ S\ ), R\ )\ )$

Décomposition en suite d'opérations algébriques

- $T_1 = \Pi_{(X)}(R)$: éléments X dans l'ensemble R
- $T_2 = \times(T_1, S)$: lier tous les X de R à tous les S
- $T_3 = \backslash(T_2, R)$: les X liés aux S qui ne seraient pas dans R
- $T_4 = \Pi_{(X)}(T_3)$: seulement les X de T_3
- $T[X] = \backslash(T_1, T_4)$: les X de R liés à tous les S

$\div(R, S)$: "Les bars qui servent toutes les bières"

$\setminus(\Pi_{(id\_bar,id\_biere)}(\times(services, bieres)), \Pi_{(id\_bar,id\_biere)}(services))$

```
CREATE  VIEW  bars_ne_servant_pas_certaines_bieres AS
SELECT s.id_bar, b.id_biere
FROM services s,bieres b
EXCEPT
SELECT id_bar,id_biere
FROM services
```

$\div(R, S) = \setminus(\ \Pi_{id\_bar}(services)\ ,\ \Pi_{id\_bar}(bars\_ne\_...\_bieres)$

```
(SELECT id_bar FROM services)
EXCEPT
(SELECT id_bar FROM bars_ne_servant_pas_certaines_bieres)
```

$\div(R, S)$: "Les bars qui servent toutes les bières"

Traduction SQL, ce que l'on veut obtenir (`SELECT`)

```
SELECT *
FROM bars ba
WHERE id_bar
```

En explicitant la clause `WHERE` : les bars qui servent toute les bières

```
IN ( (SELECT id_bar
      FROM services)
     EXCEPT
     (SELECT id_bar
      FROM bars_ne_servant_pas_certaines_bieres)
   );
```

$\div(R, S)$: formulation "naturelle"

Trouver les bars tel qu'il n'existe pas de bière pour laquelle ...

```
SELECT *
FROM bars ba
WHERE NOT EXISTS( SELECT *
                  FROM bieres bi
```

... il n'existe pas de service associant ce bar et cette bière

```
    WHERE NOT EXISTS( SELECT *
                      FROM services s
                      WHERE ba.id_bar=s.id_bar
                       AND s.id_biere=bi.id_biere
                    )
                )
```

$\div(R, S)$: double négation

"Récupérer les bars qui servent toutes les bières"

Un bar est sélectionné s'il n'existe aucune bière n'ayant été servie dans ce bar.

Même requête SQL

```
SELECT *
FROM bars ba
WHERE NOT EXISTS( SELECT * FROM bieres bi
                  WHERE NOT EXISTS(
                     SELECT *
                     FROM services s
                     WHERE ba.id_bar=s.id_bar
                       AND s.id_biere=bi.id_biere) )
```

$\div(R, S)$: différence d'ensembles

"Récupérer les bars qui servent toutes les bières"

Si la différence entre l'ensemble des bières et l'ensemble des bières servies dans un bar est nulle alors le bar sert toutes les bières

Requête SQL correspondante

```
SELECT bar
FROM bars ba
WHERE NOT EXISTS ( (SELECT id_biere FROM bieres bi)
                   EXCEPT
                   (SELECT id_biere
                    FROM services
                    WHERE id_bar=ba.id_bar) );
```

$\div(R, S)$: groupement et fonction d'agrégats

"Récupérer les bars qui servent toutes les bières"

trouver les bars qui servent un nombre de bières différentes qui soit égal au nombre total de bières de la base de données

Requête SQL correspondante

```
SELECT bar
FROM bars ba NATURAL JOIN  services s
GROUP BY bar
HAVING COUNT(DISTINCT s.id_biere)=(
                                  SELECT COUNT(id_biere)
                                  FROM bieres
                                  );
```

Conclusion

Modèle relationnel

- logique du premier ordre, des prédicats :
  - $p(x)$: propositions logiques
  - $\forall, \exists$: quantificateurs universel, existentiel
  - $\neg, \wedge, \vee$: opérateurs, connecteurs
- calcul relationnel
  - $R = \{s \mid s \in S \wedge p(s)\}$
- algèbre relationnelle
  - cinq opérateurs de base
    - $\Pi, \times, \sigma, \cup, \setminus$
  - opérateurs dérivées
    - $\bowtie$: combinaison de $\times$ et σ
    - $\cap$: déduite de $\cup$ et $\setminus$
    - $\div$: construite à partir de Π, $\times$ et $\setminus$

Bibliographie

Livres

- **Michelle Clouse** : Algèbre relationnelle : guide pratique de conception d'une base de données relationnelle normalisée
- **Laurent Audibert** : Bases de données : de la modélisation au SQL : conception des bases de données, modèle relationnel et algèbre relationnelle, langage SQL, programmation SQL : support de cours, exercices corrigés , Ellipses, 2009
- **Mokrane Bouzeghoub** : Le Modele Relationnel - Algèbre, Langages, Applications, Hermès 1998
- **C. J. Date** : SQL and Relational TheoryLe Modele Relationnel - How to Write Accurate SQL Code (3rd edition), O'Reillye 2015

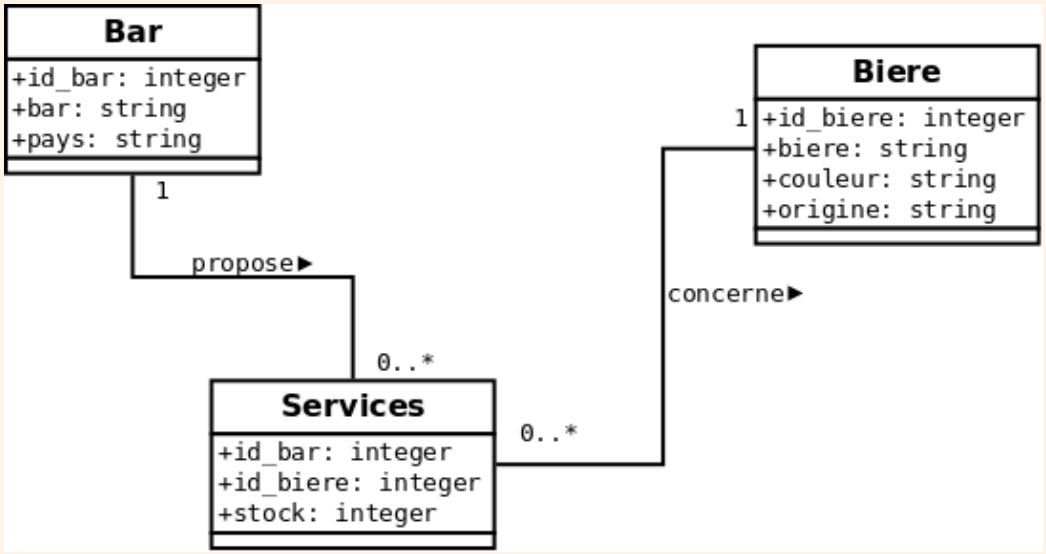
Bibliographie

Adresses “au Net”

- `www.lamsade.dauphine.fr/~manouvri` : Maude Manouvrier (`www.librecours.org`)
- `georges.gardarin.free.fr` : le site de Georges Gardarin
- `www.developpez.com` : entre autre du SQL et des SGBD ...

Annexe

Modèle de données



Modèle de données

Création des tables

```
CREATE TABLE bars (id_bar SERIAL PRIMARY KEY,
                   bar TEXT,
                   pays TEXT);
CREATE TABLE bieres (id_biere SERIAL PRIMARY KEY,
                     biere   VARCHAR(20),
                     couleur VARCHAR(10),
                     origine VARCHAR(20) );
```

Modèle de données

Création des tables

```
CREATE TABLE services (id_bar INTEGER NOT NULL,
                       id_biere INTEGER NOT NULL,
                       stock    SMALLINT,
                       PRIMARY KEY(id_bar,id_biere),
                       FOREIGN KEY(id_bar)
                         REFERENCES bars,
                       FOREIGN KEY(id_biere)
                         REFERENCES bieres);
```

Modèle de données

Insertion dans les tables

```
INSERT INTO bars (bar,pays)
  VALUES ('Bar du Coin','France');

SELECT * FROM bars;
 id_bar |     bar      |  pays
--------+--------------+--------
      1 | Bar du Coin | France
(1 row)
```

Modèle de données

Insertion dans les tables

```
INSERT INTO bieres (biere,couleur,origine)
  VALUES ('Kronenbourg', 'blonde','France');

SELECT * FROM bieres;
 id_biere |    biere     | couleur | origine
----------+--------------+---------+---------
        1 | Kronenbourg | blonde  | France
(1 row)
```

Modèle de données

Noms des séquences liées aux clés primaires

```
SELECT c.relname FROM pg_class c
WHERE c.relkind = 'S'
  AND c.relname LIKE 'bars_id%'
  OR c.relname LIKE 'bieres_id%';
```

Insertion dans les tables

```
INSERT INTO services (id_bar,id_biere,stock)
  VALUES (currval('bars_id_bar_seq'),
          currval('bieres_id_biere_seq'),1000);
```

Trouver les bars qui servent toutes les bières ...