Structured Query Language

Logique & requêtes SQL

Alexis NEDELEC

LISYC EA 3883 UBO-ENIB-ENSIETA
Centre Européen de Réalité Virtuelle
Ecole Nationale d'Ingénieurs de Brest

Un peu de logique ($\neg, \forall, \exists, \wedge, \vee$)

Formules équivalentes

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

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)”

"Récupérer les bars qui servent toutes les bières"

Reformulation de la question

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

Traduction en calcul relationnel

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

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

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

“Récupérer les bars qui servent toutes les bières”

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

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

Traduction SQL, en vérifiant (clause 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))
```

“Récupérer les bars qui servent toutes les bières”

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) )
```

“Récupérer les bars qui servent toutes les bières”

Autre formulation “naturelle”

Un bar est sélectionné s’il n’existe aucune bière n’ayant été servie dans ce bar (double negation).

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=b i.id_biere) )
```

“Récupérer les bars qui servent toutes les bières”

Autre formulation “naturelle”

si la différence entre l’ensemble des bieres et l’ensemble des bieres servies dans un bar est nulle alors le bar sert toutes les bieres

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) );
```

“Récupérer les bars qui servent toutes les bières”

Autre formulation moins “naturelle”

trouver les bars qui servent un nombre de bieres différentes qui soit égal au nombre total de bieres 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);
```

“bars qui servent les m̂ bières qu’au ‘Bar du Coin”

Reformulation de la question

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 (s) au “Bar du Coin”
alors elle sera servie (s) 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

$$\{ba \mid bars(ba), \quad \forall bi \quad services(bi, bar = 'BarduCoin') \Rightarrow \exists s \; services(bi, ba)\}$$

“bars qui servent les m̂ bières qu’au ‘Bar du Coin”

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

$$\forall bi \quad \neg services(bi, bar = \ 'BarduCoin') \vee \exists s \ services(bi, ba)$$

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

$$\neg(\exists bi \quad services(bi, bar = \ 'BarduCoin') \wedge \neg \exists s \ services(bi, ba))$$

Reformulation en calcul relationnel

$$\{ba \mid bars(ba), \neg(\exists s1 \ services(bi, bar = \ 'BarduCoin') \wedge \neg \exists s2 \ services(bi, ba) \wedge s1.id\_biere = s2.id\_biere \wedge s2.id\_bar = ba.id\_bar)$$

Calcul Relationnel

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));
```