Cours de bases de données, , http://sql.bdpedia.fr

Quelques notions de logique

Philippe Rigaux

le cnam

Notions élémentaires de logique formelle

Cette session présente les notions essentielles de logique formelle à connaître :

- Le calcul propositionnel : valeurs de vérité et formules
- Les prédicats
- Collections et quantificateurs
- Importance pour les bases de données et SQL

Ces diapositives correspondent au support en ligne disponible sur le site `http://sql.bdpedia.fr/`

Calcul propositionnel

Proposition : énoncé auquel on peut affecter deux valeurs de vérité, Vrai ou Faux.

Formule : expression basée sur des propositions et leur combinaison par des **connecteurs logiques**.

- la conjonction, notée $\wedge$
- la disjonction, notée $\vee$
- la négation, notée $\neg$

Exemple : $(p \wedge q) \vee r$

Valeur de vérité d'une formule

Induite à partir de valeurs de vérité des propositions, et des règles suivantes :

p	q	$p \wedge q$	$p \vee q$	$\neg p$
Vrai	Vrai	Vrai	Vrai	Faux
Vrai	Faux	Faux	Vrai	Faux
Faux	Vrai	Faux	Vrai	Vrai
Faux	Faux	Faux	Faux	Vrai

Exemple : $(p \wedge q) \vee r$ est Vrai pour les valeurs V, V, F de p, q, r

Equivalences

Un informaticien averti sait manier avec agilité les équivalences. Quelques exemples.

- $\neg(\neg F)$ est équivalente à F
- $F \vee (F_1 \wedge F_2)$ est équivalente à $(F \vee F_1) \wedge (F \vee F_2)$ (distribution)
- $F \wedge (F_1 \vee F_2)$ est équivalente à $(F \wedge F_1) \vee (F \wedge F_2)$ (distribution)
- $\neg(F_1 \wedge F_2)$ est équivalente à $(\neg F_1) \vee (\neg F_2)$ (loi DeMorgan)
- $\neg(F_1 \vee F_2)$ est équivalente à $(\neg F_1) \wedge (\neg F_2)$ (loi DeMorgan)

Donc $p \vee \neg(p \vee \neg q)$ est une **tautologie**.

Prédicats

Extension puissante des propositions : **construire des énoncés sur des "objets"**.
Le prédicat $Compose(X, Y)$ permet de construire des énoncés de la forme :

- $Compose$('Mozart', 'Don Giovanni')
- $Compose$('Debussy', 'La mer')
- Etc.

Ce sont des nuplets (ou des atômes, ou des faits).

- Il en existe une infinité
- Un **contexte** (ou "interprétation") définit ceux qui sont vrais / faux.

Remarque: Base de données ?

C'est un contexte donnant un ensemble fini de faits vrais.
Tous les autres sont considérés comme faux.

Nuplets ouverts et fermés

Un nuplet énoncé avec des constantes est un nuplet **fermé**.

$$Compose(\text{'Mozart'}, \text{'Don Giovanni'})$$

Un nuplet énoncé avec au moins une variable est un nuplet **ouvert**.

$$Compose(X, \text{'Don Giovanni'})$$

Un nuplet ouvert désigne une infinité de faits possibles.

Remarque: Interêt ?

En général on s'intéresse aux valeurs de X pour lesquelles les faits sont vrais. **On a effectué une requête**.

Collections et quantificateurs

Les nuplets libres expriment des contraintes sur un fait.

On peut exprimer des contraintes sur des collections de fait avec les quantificateurs.

- $\exists x P(x)$ est vraie s'il existe **au moins** aune affectation de X pour laquelle $P(x)$ est vraie.
- $\forall x P(x)$ est vraie si $P(x)$ est vraie pour **toutes** les valeurs de x.

Variables libres et liées : un variable quantifiée est libre ;
sinon elle est liée.

SQL = formules logique

Requête SQL = une formule avec des variables libres.

Résultat d'une requête = les valeurs des variables libres qui satisfont la formule.

La formule *Compose*(X, 'Don Giovanni') s'écrit en SQL

```
select compositeur
  from Compose
  where oeuvre='Don Giovanni'
```

SQL c'est une syntaxe pour écrire des formules.

Remarque: Déclarativité

On ne dit pas comment on calcule !

Exemples

Deux relations/prédicats :

- `Expert (id_expert, nom)`
- `Manuscrit (id_manuscrit, auteur, titre, id_expert, commentaire)`

$Manuscrit(\_, \text{'Proust'}, t, x, \_) \wedge Expert(x, n)$

```
select titre, nom from Expert, Manuscrit
where Expert.id_expert = Manuscrit.id_expert
and titre = 'Proust'
```

$Expert(x, n, \_) \wedge \exists Manuscrit(\_, \text{'Proust'}, \_, x, \_)$

```
select nom from Expert as e
where exists (select *  from Manuscrit as m
              where e.id_expert = m.id_expert
              and titre = 'Proust')
```

À retenir

SQL est un langage **déclaratif** qui exprime par une formule logique les propriétés du résultat à construire.

Avantages prouvés et éprouvés depuis les années 1970.

- signification précise, non ambigue
- algorithmes efficaces
- langage robuste, universellement connu et adopté, **normalisé**
- **déclarativité** : SQL ne donne aucune indication sur la manière dont le système doit trouver le résultat.