Theory I: Database Foundations

Jan-Georg Smaus (Georg Lausen)

26.7.2011

1. Relational Calculus
   Review
   Relational Calculus
   Semantics

Let's look back: Foundations of Programming Languages and Software Engineering

- In the part "Foundations of Programming Languages and Software Engineering" (Chapter 13), we have learnt a general formalism for generating a language based on specifying some symbols.
  - The symbols are: function symbols, each with an arity and defined by a signature; and variables. Function symbols of arity 0 are called constants.
  - We call the elements of the generated language $T(\Sigma, X)$ terms.
- For databases, we will now also learn a general formalism for generating a language based on specifying some symbols. Although the terminology differs a bit, the formalism works in a very similar way:
  - As before, we have constants and variables.
    We do not have (general) function symbols any more.
    In addition, we have quantifiers, connectives and relation symbols.
  - We call the elements of the generated language will be called formulas.
- $\Rightarrow$ First-order logic

Syntax of the relational calculus

Consider a database schema $\mathcal{R}$ defining some relation names and constants (domain elements).
Formulas (R-formulas) are built out of constants, variables, relation names, connectives $\neg, \wedge, \vee$, quantifiers $\forall, \exists$ und auxiliary symbols '(', ')', ',':

Atomic formulas

- Let R be a relation name of arity k. Let $a_1, \ldots, a_k$ be constants or variables.

  $R(a_1, \ldots, a_k)$ is an (atomic) R-formula (over $\mathcal{R}$).
- Let selection condition α be given as $X\theta Y$, or $X\theta a$, or $a\theta X$, where X, Y are variables, a is a constant and $\theta \in \{=, \neq, \leq, <, \geq, >\}$ is a comparison operator.

  α is an (atomic) R-formula (over $\mathcal{R}$).

. . .

Syntax of the relational calculus

Composite formulas

- Let F be an R-formula $(\ldots)$.

  $\neg F$ is an R-formula.
- Let F be an R-formula containing a variable X, however not containing an expression of the form $\exists X$, resp. $\forall X$. X is called free in F.

  $\exists X\ F$ is called a ($\exists$-quantified) R-formula. X is called bound in $\exists X\ F$.
  $\forall X\ F$ is called a ($\forall$-quantified) R-formula. X is called bound in $\forall X\ F$.

  F is the scope of the $\exists$-, resp. $\forall$-quantifier.

. . .

Syntax of the relational calculus

Composite formulas (2)

- Let F and G be R-formulas and let $\mathcal{V}_F$, resp. $\mathcal{V}_G$ the set of variables contained in F, resp. G, where variables occuring in $\mathcal{V}_F \cap \mathcal{V}_G$ are free in F and free in G.

  The conjunction $(F \wedge G)$ is an R-formula.
  The disjunction $(F \vee G)$ is an R-formula.

Queries

- A relational calculus query Q over a database schema $\mathcal{R}$ is given as

$$\{(a_1, \ldots, a_n) \mid F\},$$

  where F an R-formula over $\mathcal{R}$ and $a_1, \ldots, a_n$ variables and constants.
- The set of variables among the a_i equals the set of free variables in F.
- To state a format of the result we can write

$$\{(a_1 : A_1, \ldots, a_n : A_n) \mid F\}$$

  where the A_i are domains / attributes.

Variable assignments

- Let the set of variables of F be $\mathcal{V}_F$. A variable assignment ν of F is a function over $\mathcal{V}_F$:

$$\nu : \mathcal{V}_F \to dom.$$

- We extend ν by identity for constants; for any constant a it holds that $\nu(a) = a$ (blurring of syntax and semantics!).

Truth

Consider a database schema $\mathcal{R}$ defining some relation names and constants, and an instance $\mathcal{I}$, and a variable assignment ν.

- $R(a_1, \ldots, a_k)$ is true under ν wrt. $\mathcal{I}$ iff $(\nu(a_1), \ldots, \nu(a_k)) \in \mathcal{I}(R)$.
- $X\theta Y$, $X\theta a$, or $a\theta X$, resp., are true under ν wrt. $\mathcal{I}$ iff $\nu(X)\theta\nu(Y)$, $\nu(X)\theta a$, or $a\theta\nu(X)$, resp., hold in "the usual mathematical sense".
- $\neg F$ is true under ν wrt. $\mathcal{I}$ iff F is not true under ν wrt. $\mathcal{I}$.
- $\exists X\ F$ is true under ν wrt. $\mathcal{I}$ iff there exists a domain element c such that for

$$\nu'(Y) = \begin{cases} \nu(Y) & \text{for } Y \neq X \\ c & \text{for } Y = X \end{cases}$$

  we have that F is true under ν' wrt. $\mathcal{I}$
- $(F \wedge G)$ is true under ν wrt. $\mathcal{I}$ iff F is true under ν wrt. $\mathcal{I}$ and G is true under ν wrt. $\mathcal{I}$.

This semantics is standard in logic, see Chapter 13 (propositional logic) of this course!

Semantics of a query

- Let $Q = \{(a_1, \ldots, a_n) \mid F\}$ be a query, where $a_1, \ldots, a_n$ are variables and constants and F is an R-formula.
  The answer to Q wrt. instance $\mathcal{I}$, $Q(\mathcal{I})$ is defined as follows:

$$Q(\mathcal{I}) = \{(\nu(a_1), \ldots, \nu(a_n)) \mid \quad \nu \text{ is a variable assignment on } \mathcal{V}_F \text{ such that } F \text{ is true under } \nu \text{ wrt. } \mathcal{I}\}.$$

- We may write $Q(\mathcal{I}, dom)$ to emphasise the domain used for the variable assignments.

Example

Consider schemata $R(A, B), S(C, D)$ with instances r, s. Let $Q = \{(X : A, Y : B, V : C, W : D) \mid R(X, Y) \wedge S(V, W) \wedge Y > V\}$ be a query.

$$r = \begin{array}{cc} A & B \\ \hline 1 & 2 \\ 2 & 2 \\ 2 & 1 \end{array} \qquad s = \begin{array}{cc} C & D \\ \hline 1 & 1 \\ 1 & 2 \\ 3 & 1 \end{array} \qquad \stackrel{Q}{\Longrightarrow}$$

Examples on Comparison with Relational Algebra

Consider schemata $R(A, B)$ and $S(B, C)$ with instance $\mathcal{I} = \{r, s\}$. For each expression of the relational algebra below, let us define a query Q such that $Q(\mathcal{I})$ corresponds to that expression.

- $\pi[A]\sigma[B = 5]R \equiv$
- $\pi[A]R \equiv$
- $\sigma[A = B]R \equiv$
- $R \bowtie S \equiv$

Domain independence

- Let $Q := \{(a_1, \ldots, a_n) \mid F\}$. Let $\mathcal{I}$ an instance to $\mathcal{R}$ and *adom* the set which contains all constants in Q and all constants mentioned in $\mathcal{I}$. *adom* is called active domain Q; *adom* is finite.
- Q is called domain independent, if for any set $D \supset adom$ it holds that:

$$Q(\mathcal{I}, adom) = Q(\mathcal{I}, D).$$

Example: queries that are not domain independent

- $R(A)$ a schema, Q a query given as

$$\{X \mid \neg R(X)\},$$

  where $\mathcal{I}(R) = \{1\}$.
- $R(A, B)$ and $S(B, C)$ schemata. Q a query given as

$$\{(X, Z) \mid \exists Y(R(X, Y) \vee S(Y, Z))\},$$

  where $\mathcal{I}(R) = \{(1, 1)\}$, resp. $\mathcal{I}(S) = \emptyset$.

Safety

If R-formula F is safe, then F domain independent.

- F does not contain $\forall$.
- If $F_1 \vee F_2$ is a subformula of F, then F_1 and F_2 have to contain the same free variables.
- A subformula G of F is called maximally conjunctive, if F does not contain a subformula of the form $H \wedge G$ or $G \wedge H$.

  Let $F_1 \wedge \cdots \wedge F_m, m \geq 1$, be a maximally conjunctive subformula of F. All free variables X have to be bounded in the following sense ($1 \leq j \leq m$):
  - If X is free in F_j, where F_j neither a comparison nor negated, then X bounded.
  - If F_j of the form $X = a$ or $a = X$ and a a constant, then X bounded.
  - If F_j of the form $X = Y$ or $Y = X$ and Y bounded, then X bounded.

Examples

- $\{(X, Y) \mid X = Y \vee R(X, Y)\}$ is not safe.
- $\{(X, Y) \mid X = Y \wedge R(X, Y)\}$ is safe.
- $\{(X, Y, Z) \mid R(X, Y, Z) \wedge \neg(S(X, Y) \vee T(Y, Z))\}$ is not safe, however is safe when written equivalently as

$$R(X, Y, Z) \wedge \neg S(X, Y) \wedge \neg T(Y, Z)$$