course:

Database Systems (A7B36DBS)

lecture 7:

Query formalisms for relational model – relational calculus

Doc. RNDr. Irena Holubova, Ph.D.

Acknowledgement:
The slides were kindly lent by Doc. RNDr. Tomas Skopal, Ph.D.,
Department of Software Engineering, Charles University in Prague

Today's lecture outline

- relational calculus
  - domain relational calculus
  - tuple relational calculus
  - safe formulas

Map of the Lecture

Database design

Formulation of the task

Conceptual modelling

ER, UML

Transformation to **relational** model

or XML, object, object-relational…

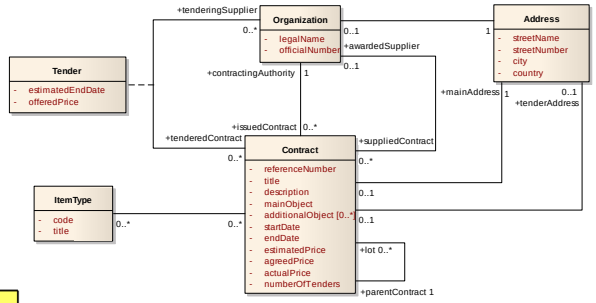
functional dependencies, normal forms, decomposition, synthesis

SQL: Data Definition

relational algebra, relational calculus

SQL: Data Manipulation

Example: Information system for contracts, organizations and tenders



Name	Producer	Price	Availability
Mouse	Dell	5,80	100000
Windows	Microsoft	12,50	NULL
Printer	Toshiba	325,00	15000
Phone	Nokia	135,00	32000
Laptop	Dell	500,00	9000
Bing	Microsoft	0,00	NULL

key

foreign key
(key in table Producer)

Name	Address	Debt
Dell	USA	YES
Microsoft	USA	NO
Samsung	Korea	NO
Nokia	Norway	YES
E.ON	Germany	NO
Toshiba	Japan	NO

key

```
CREATE TABLE Contract (
   id INTEGER,
   startDate DATETIME,
   endDate DATETIME,
   ...);
```

physical storage, implementation of indices

```
SELECT id, name
FROM Contract, Organization
WHERE Organization.id = ... AND
      startDate > ...;
```

Relational calculus

- application of first-order calculus (predicate logic) for database querying
- extension by **„database predicate"** testing a membership of an element in a relation, defined at two levels of granularity
  - domain calculus (DRC) – variables are attributes
  - tuple calculus (TRC) – variables are tuples (whole elements of relation)
- query result in DRC/TRC is relation (and the appropriate schema)

Relational calculus

- the "language"
  1. **terms** – variables and constants
  2. **predicate symbols**
     - standard binary predicates {<, >, =, ≥, ≤, ≠}
     - „database predicates" (extending the first-order calculus)
  3. **formulas**
     - atomic – $R(t_1, t_2, \ldots)$, where R is predicate symbol and t_i is term
     - complex – expressions that combine atomic or other complex formulas using logic predicates $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
  4. **quantifiers** $\exists$ (existential), $\forall$ (universal)

Relational calculus

A	B	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
1	1	1	1	1	1
1	0	0	1	0	0
0	1	0	1	1	0
0	0	0	0	1	1

Domain relational calculus (DRC)

- **variables** stand for attributes (= their values)
- database predicate: R(x, y, …)
  - R stands for the name of table the predicate is applied on
  - predicate that for interpreted *x*, *y*, … returns ***true*** if there is an element in R (table row) with the same values
    - i.e., row membership test
  - predicate scheme (input parameters) is the same as the scheme of relation R, i.e., each parameter *x*, *y*,… is substituted by a (interpreted) variable or constant

Domain relational calculus (DRC)

- database predicate
  - **full notation:** variables and constants in the predicate have an attribute name assigned, determining the value testing, e.g.:
    CINEMA(*NAME_CINEMA* : x , *FILM* : y)
  - **short notation:** if the list of variables and constants does not contain the attribute names, it is assumed they belong to the attributes given by the relation schema R, e.g.:
    CINEMA(x, y)
    - where (*NAME_CINEMA, FILM*) is the schema of CINEMA
    - in the following we consider this short notation

Domain relational calculus (DRC)

- the result of query given in DRC is a set of all interpretations of variables and constants in the form of ordered *n*-tuple, for which the formula of the query is ***true***

{(t_1, t_2, …) | query formula that includes t_1, t_2, … }

  - t_i is either a constant or a **free variable**, i.e., a variable that is not quantified inside the formula
  - the schema of the result relation is defined directly by the names of the free variables

{(x, y) | CINEMA(x, y)}

returns relation consisting of all CINEMA elements

{(x) | CINEMA(x, 'Titanic')}

returns names of cinemas that screen the film Titanic

Domain relational calculus (DRC)

- quantifiers allow to declare **bound variables** that are interpreted within the database predicates
  - formula $\exists x\ R(t_1, t_2, \ldots, x, \ldots)$ is evaluated as ***true*** if there **exists** domain interpretation of **x** such that *n*-tuple (t_1, t_2, …, **x**, …) is a member of **R**
  - formula $\forall x\ R(t_1, t_2, \ldots, x, \ldots)$ is evaluated as true if **all** domain interpretations of **x** leading to *n*-tuples (t_1, t_2, …, x, …) are members of **R**

{(film) | ∃name_cinema CINEMA(name_cinema, film) }

returns names of all films screened **at least** in one cinema

Domain relational calculus (DRC)

- it is important to determine which domain is used for interpretation of variables (both bound and free variables)
  1. domain can be unspecified (i.e., interpretation is not limited) – the domain is the whole **universum**
  2. domain is an attribute type (i.e., integer, string, …) – **domain** interpretation
  3. domain is a set of values of a given attribute that exist in the relation on which the interpretation is applied – **actual domain** interpretation

Domain relational calculus (DRC)

{(film) | ∀name_cinema CINEMA(name_cinema, film) }

could be evaluated differently based on the way variable **name_cinema** is interpreted

- if the **universum** is used, the query result is an empty set, because the relation CINEMA is surely not infinite, also it is type-restricted
  - e.g., values in **NAME_CINEMA** will not include 'horse', 125, 'quertyuiop'
- if the **domain** (attribute type) is used, the query answer will be also empty – still infinite relation CINEMA assumed, containing all strings, e.g., 'horse', 'qwertyuiop', …
- if the **actual domain** is used, the query result consists of names of films screened in all cinemas (that are contained in the relation **CINEMA**)

Domain relational calculus (DRC)

- if we implicitly consider interpretation based on the actual domain, we call such limited DRC as **DRC with limited interpretation**

- **another simplification:** because schemas often consist of many attributes, we can use simplifying notation of quantification
  - an expression $R(t_1, \ldots, t_i, t_{i+2}, \ldots)$,
    i.e., t_{i+1} is missing,
    is understood as $\exists t_{i+1}\ R(t_1, \ldots, t_i, \mathbf{t_{i+1}}, t_{i+2}, \ldots)$
    - the binding of variables then must be clear from the context, or strict attribute assignment must be declared
      in the following we will assume that the names of variables denote the respective columns

{(name) | CINEMA(name)} is the same as **{(name) | ∃film CINEMA(name, film)}**

Examples – DRC

FILM(NAME_FILM, NAME_ACTOR) ACTOR(NAME_ACTOR, YEAR_BIRTH)

In what films all the actors appeared?

$\{(film) \mid FILM(film) \wedge \forall actor\ (ACTOR(actor) \Rightarrow FILM(film, actor))\}$

Which actor is the youngest?

$\{(actor,year) \mid ACTOR(actor,year) \wedge \forall actor2\ \forall year2\ (ACTOR(actor2,year2) \wedge actor \neq actor2) \Rightarrow year2 < year\}$

or

$\{(actor,year) \mid ACTOR(actor,year) \wedge \forall actor2\ (ACTOR(actor2) \Rightarrow \neg\exists year2(ACTOR(actor2,year2) \wedge actor \neq actor2 \wedge year2 > year))\}$

Which pairs of actors appeared at least in one film?

$\{(actor1, actor2) \mid ACTOR(actor1) \wedge ACTOR(actor2) \wedge actor1 \neq actor2 \wedge \exists film\ (FILM(film, actor1) \wedge FILM(film, actor2))\}$

Evaluation of DRC query

Which actor is the youngest?

$\{(a,y) \mid ACTOR(a,y) \wedge \forall a2(ACTOR(a2) \Rightarrow \neg\exists y2\,(ACTOR(a2,y2) \wedge a \neq a2 \wedge y2 > y))\}$

universal quantifier = chain of conjunctions

existential quantifier = chain of disjunctions

```
$result = ∅
for each (a,y) do
    if (ACTOR(a,y) and
        (for each a2 do
            if (not ACTOR(a2) or not (for each y2 do
                if (ACTOR(a2,y2) ∧ a ≠ a2 ∧ y2 > y) = true then return true
            end for
            return false)) = false then return false

        end for
        return true) ) = true then Add (a,y) into $result
end for
```

Tuple relational calculus (TRC)

- almost the same as DRC
- the difference is **variables** are whole elements of relations (i.e., **rows of tables**), i.e., predicate ***R***(***t***) is interpreted as true if (a row) ***t*** belongs to ***R***
  - the resulting schema is defined by concatenation of schemas of the free variables (*n*-tuples)

- to access the attributes within a tuple ***t***, a "**dot notation**" is used
  - **{t | CINEMA(t) ∧ t.FILM = 'Titanic'}** returns cinemas which screen film Titanic

- the result schema could be **projected using [...]** only on a subset of attributes
  - **{t[NAME_CINEMA] | CINEMA(t)}**

Examples – TRC

FILM(NAME_FILM, NAME_ACTOR) ACTOR(NAME_ACTOR, YEAR_BIRTH)

Get the pairs of actors of the same age acting in the same film.

$$\{a1, a2 \mid ACTOR(a1) \wedge ACTOR(a2) \wedge a1 \neq a2 \wedge a1.YEAR\_BIRTH = a2.YEAR\_BIRTH \wedge \exists f1, f2\ (FILM(f1) \wedge FILM(f2) \wedge f1.NAME\_FILM = f2.NAME\_FILM \wedge f1.NAME\_ACTOR = a1.NAME\_ACTOR \wedge f2.NAME\_ACTOR = a2.NAME\_ACTOR)\}$$

Which films were casted by all the actors?

$$\{fi[NAME\_FILM] \mid FILM(fi) \wedge \forall actor(ACTOR(actor) \Rightarrow \exists f(FILM(f) \wedge f.NAME\_ACTOR = actor.NAME\_ACTOR \wedge f.NAME\_FILM = fi.NAME\_FILM))\}$$

Safe formulas in DRC

- unbound interpretation of variables (domain-dependent formulas, resp.) could lead to infinite query results
  - negation: $\{x \mid \neg R(x)\}$
    - e.g. $\{j \mid \neg \text{Employee(Name: j)}\}$
  - disjunction: $\{x, y \mid R(\ldots, x, \ldots) \vee S(\ldots, y, \ldots)\}$
    - e.g. $\{i, j \mid \text{Employee(Name: i)} \vee \text{Student(Name: j)}\}$
  - universal quantifiers lead to an empty set
    $\{x \mid \forall y\ R(x, \ldots, y)\}$, and generally $\{x \mid \forall y\ \varphi(x, \ldots, y)\}$, where φ does not include disjunctions (implications, resp.)
- the solution is to limit the set of DRC formulas – set of safe formulas

Safe formulas in DRC

- to simply avoid infinite quantification ad-hoc,
  it is good to constrain the quantifiers so that the
  interpretation of bound variables is limited to a finite set
  - using $\exists \mathbf{x}\ (\mathbf{R(x)} \wedge \varphi\mathbf{(x)})$ instead of $\exists \mathbf{x}\ (\varphi\mathbf{(x)})$
  - using $\forall \mathbf{x}\ (\mathbf{R(x)} \Rightarrow \varphi\mathbf{(x)})$ instead of $\forall \mathbf{x}\ (\varphi\mathbf{(x)})$
    - by this convention the evaluation is implemented as

```
for each x in R        // finite enumeration
   instead of
for each x             // infinite enumeration
```

- **free variables** in $\varphi(x)$ can be limited as well – by conjunction
  - $\mathbf{R(x)} \wedge \varphi\mathbf{(x)}$

Examples – safe formulas

$\{x,y \mid x = y\}$ — **not safe** (x, y not limited)

$\{x,y \mid x = y \vee R(x,y)\}$ — **not safe**

(the disjunction elements share both free variables, but the first maximal conjunction (x=y) contains equation of not limited variables)

$\{x,y \mid x = y \wedge R(x,y)\}$ — **is safe**

$\{x,y,z \mid R(x,y) \wedge \neg(P(x,y) \vee \neg Q(y,z))\}$ — **not safe**

(*z* is not limited in the conjunction + the disjunction elements do not share the same variables)

$\{x,y,z \mid R(x,y) \wedge \neg P(x,y) \wedge Q(y,z)\}$ — **equivalent formula to the previous one – now safe**

Relational calculus – properties

- "even more declarative" than relational algebra
  (where the structure of nested operations hints the evaluation)
  - just specification of what the result should satisfy
- both DRC and TRC are relational complete
  - moreover, could be extended to be stronger
- besides the different language constructs, the three formalisms can be used for differently "coarse" access to data
  - operations of relational algebra work with entire relations (tables)
  - database predicates of TRC work with relation elements (rows)
  - database predicates of DRC work with attributes (attributes)

Examples – comparison of RA, DRC, TRC

FILM(NAME_FILM, NAME_ACTOR) ACTOR(NAME_ACTOR, YEAR_BIRTH)

Which films were casted by all the actors?

RA:

$FILM \div ACTOR[NAME\_ACTOR]$

DRC:

$\{(film) \mid FILM(film) \wedge \forall actor\ (ACTOR(actor) \Rightarrow FILM(film, actor))\}$

TRC:

$$\{film[NAME\_FILM] \mid FILM(film) \wedge \forall actor(ACTOR(actor) \Rightarrow \exists f(FILM(f) \wedge f.NAME\_ACTOR = actor.NAME\_ACTOR \wedge f.NAME\_FILM = film.NAME\_FILM))\}$$