

Avancerede emner i databaser

Kursus på Datalogi-OB, Forår 2003

by Jørgen Villadsen
Davide Martinenghi
Henning Christiansen

Computer Science



Roskilde University

Computer Science, building 42.1
Roskilde University
Universitetsvej 1
P.O. Box 260
DK-4000 Roskilde
Denmark
Phone: +45 4674 2000
Fax: +45 4674 3072
www.dat.ruc.dk

Advanced topics in databases

A Computer Science course, Spring 2003

by Jørgen Villadsen
Davide Martinenghi
Henning Christiansen

Computer Science



Roskilde University

Computer Science, building 42.1
Roskilde University
Universitetsvej 1
P.O. Box 260
DK-4000 Roskilde
Denmark
Phone: +45 4674 2000
Fax: +45 4674 3072
www.dat.ruc.dk

Outline of today's program

- Presentation of teachers
- Purpose, focus, and approach in the course
- Presentation of students: background, interests
- Demonstration: Prolog as database engine
"logic at work"
- Introduction to literature, central paper(s)
- Proposal for a course schedule



Purpose

Continue from std. DB course into areas where RDBMS gives little or no support...

To be able to

- improve present day RDBMS technology
- develop smart frontends, e.g., generate DBs from high-level spec's
- provide methodology for DB designers

Give background for project works/Master's theses

Provide you with basic theory, methods and tools to help you advance DB technology



Focus in this course

Logical properties of databases with special emphasis on *integrity constraints* (ICs):

- Specification
- Maintenance, enforcement in DB
 - checking efficiently before update, update routines, ...
- Help developer to ensure correctness of DB application
- Perhaps also other applications of ICs:
 - Intensional answers, data integration, semantic query optimization
- Perhaps other applications of logic in DBs:
 - View updates, query transformation, summaries...



Approach and work at the course

Databases formulated as first-order logic \approx
clauses in logic prog. language Prolog

- provides a flexibility of spec's and transformations that goes far beyond SQL and RA
- Prolog a possible experimentation tool

Perspectives: Should be transferable to RDBS, not
(only) academic exercises

At the course

1. Prolog as database & database impl. language
2. Central literature (lectures, planned exercises)
3. Practical experiments, presentations, discussions..



Motivating example: A DB of personal info.

Tabled relations for persons, fathers, mothers, etc.
with properties we expect.

In SQL something like:

```
CREATE TABLE PERSON(NAME ... IS KEY, ADDRESS, GENDER)
CREATE TABLE FATHER(NAME-F..., NAME-C...)
CREATE TABLE MOTHER(NAME-F, NAME-C)
CREATE VIEW PARENT AS FATHER UNION MOTHER
```

....



Example, cont'd

Person relation in SQL (sketch of syntax)

```
CREATE TABLE PERSON(NAME, ADDRESS, GENDER)
```

Key constraints on NAME

Intuitive: "Name uniquely determines person"

SQL: NAME CHAR(10) primary key

Logic

$\square n, a, a', g, g' \left(\text{PERSON}(n, a, g) \wedge \text{PERSON}(n, a', g') \right)$

$\square a = a' \wedge g = g'$



Example, cont'd

Person relation in SQL (sketch of syntax)

```
CREATE TABLE PERSON(NAME, ADDRESS, GENDER)
```

Domain constraint on GENDER:

Intuitive: "Gender is either male or female"

```
SQL: CREATE DOMAIN GenderDomain  
      GENDER GenderDomain
```

Logic

$$\exists n,a,g (\text{PERSON}(n,a,g) \wedge (g = \text{"male"} \vee g = \text{"female"}))$$


Example, cont'd

Father relation in SQL (sketch of syntax)

```
CREATE TABLE FATHER(NAME-F, NAME-C)
```

Integrity constraint:

Intuitive: "You can only have one father who must be male; both of you must exist"

SQL: FOREIGN KEY NAME-C references PERSON(NAME),
NAME-C references PERSON(NAME)

Logic:

$$\exists f, c (\text{father}(f, c) \wedge \exists a (\text{person}(f, a, \text{male}) \wedge \exists g (\text{person}(c, a, g)))$$


Example cont's

A relation married(husband, wife)

IC: "you can only marry persons of opposite gender, one at a time, only existing persons and only if you exist"

SQL: Tricky to express/enforce!

Logic:

$$\begin{aligned} & \exists h,w(\text{married}(h,w) \wedge \\ & \quad \exists a(\text{person}(h,a,\text{male})) \\ & \quad \wedge \exists a(\text{person}(w,a,\text{female})) \\ & \quad \wedge \text{not } \exists w'(w \neq w' \wedge \text{married}(h,w')) \\ & \quad \wedge \text{not } \exists h'(h \neq h' \wedge \text{married}(h',w)) \end{aligned}$$


The purpose and application of IC's

Important knowledge about data

- In RDMBS, no explicit IC's, implemented by triggers, (assertions), handwritten code

Ensure that updates preserve IC's

- Difficult to ensure that all this code is correct

User-dialogues that prepare transaction that contains user's intention and preserves IC's

- Handwritten code; maybe tricky to get 100% correct

QUESTION: Could be use logic as a means to do this (and other appl's of ICs) in a systematic way?



Evaluations of ICs at update

This way?

Convert ICs to big SQL query, Q so $Q=\emptyset$ means OK;
Execute update;
Evaluate Q , if empty OK, else undo update;

Too inefficient, quadratic or worse!

Another approach, simplified integrity constraints!

(illustrate informally)



Simplified integrity constraints

Consider update "Freddy and Mary got married today"

IC: "you can only marry persons of opposite gender, one at a time, only existing persons and only if you exist"

We check update preserves IC by checking:

Does Freddy exist?

Does Mary exist?

Is Freddy not married?

Is Mary not married?

Are they of opposite gender?

In demand: A methodology to perform simplification — automatically, perhaps.

Update routine derived from IC

Consider update "Freddy and Mary got married today"

Assume checking reveals

Peter is married to Mary

Freddy does not exist

System asks user

Do you want to create a record for Freddy, or did you mean another guy (perhaps Peter)?

Are you sure you mean Mary? Should we divorce her from Peter or did you mean another girl?

And so on, checking responses and only suggest alternatives that may lead to consistency.....



Intensional answers derived from IC's

Can you give me a list of all men married to three women at 1 jan 2003?

There is no such man, because no man can be married to more than one woman!

Who is married to Freddi?

No one, and I do not know any Freddi?



This course

- We shall give a background as to characterize such systematic usage of ICs
- We use logic formulas and transformations – not that difficult :)
- Goal: Systematic, perhaps automated tools that interface to RDMS (or other ...)
- And use Prolog as "animation tool" which makes things more fun

