CSE544

Monday
April 26, 2004

1

Announcements

- Project Milestone
  - Due today
- Next paper: *On the Unusual Effectiveness of Logic in Computer Science*
  - Need to read only up to section 3
  - Review due on Wednesday
  - It's very short; review should be similar :)

2

XML Storage

Shanmugasundaram's paper:

- Shred XML data à relations
  - Easy: use the DTD
- Translate XML queries à SQL queries
  - Largely ignored in the paper
- Tagging
  - SQL tuple streams à XML
  - How do we do that ?

3

XML Storage

Other ways:

- Schema independent shredding
- BLOBs
- Use an object storage system

4

OO Databases

- Started late 80's
  - The OO Manifesto
- Main idea:
  - Toss the relational model !
  - Use the OO model – e.g. C++ classes

5

OO Databases

Two interpretations:

- Make a programming language persistent (ObjectStore)
  - No query language
  - Niche market
  - ObjectStore is still around, renamed to Exelon, stores XML objects now
- Build a new database from scratch (O_2)
  - Elegant extension of SQL
  - Later adopted by ODMG in the OQL language

6

ODL / OQL

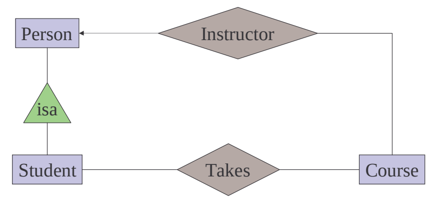
```
interface Person
  (extent People key ssn)
{  attribute string  ssn;
   attribute string  dept;
   attribute string  name;}
```

```
interface Course
  (extent Crs key cid)
{  attribute string cid;
   attribute string cname;
   relationship Person instructor;
   relationship Set<Student> stds
       inverse takes;}
```

```
interface Student extends Person
  (extent Students)
{  attribute string major;
   relationship Set<Course> takes inverse stds;}
```

7

Same in E/R



8

ODL / OQL

```
select C.name
from  Crs C
where C.instructor.dept = 'EE'
```

Follow pointers instead of joins

```
select S.name
from  Students S, S.takes C
where C.instructor.dept = 'EE'
```

Is this more efficient than joins?

9

Object-Relational (OR) Databases

- Take an incremental approach
- Keep the relational model, but allow attributes of complex types
  - Inheritance
  - Pointers
  - Methods (a security nightmare)
- All major commercial databases today are OR
- Trend: XML datatype

10

Theory

- Recall: relational databases invented by a theoretician (Codd)
- Fundamental principle: separate the WHAT from the HOW - *data independence*
- WHAT: First Order Logic (FO)
- HOW: Relational algebra (RA)

11

FO Syntax

Given:

- A vocabulary: $R_1, \ldots, R_k$
- An arity, $ar(R_i)$, for each $i=1,\ldots,k$
- An infinite supply of variables $x_1, x_2, x_3, \ldots$
- Constants: $c_1, c_2, c_3, \ldots$

12

FO Syntax

- Terms (t) and FO formulas (φ) are:

$$t ::= x \mid c$$

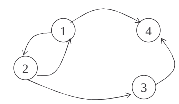
$$\varphi ::= R(t_1, ..., t_{ar(R)}) \mid t_i = t_j \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi' \mid \neg\varphi' \mid \forall x.\varphi \mid \exists x.\varphi$$

13

FO Examples

Most interesting case:

Vocabulary = one binary relation R (encodes a graph)



R=

1	2
2	1
2	3
1	4
3	4

14

FO Sentences

- Does there exists a loop in the graph ?

$\varphi \equiv \exists x.R(x,x)$

- Are there paths of length >2 ?

$\varphi \equiv \exists x.\exists y.\exists z.\exists u.(R(x,y) \wedge R(y,z) \wedge R(z,u))$

- Is there a "sink" node ?

$\varphi \equiv \exists x.\forall y.R(x,y)$

15

FO Queries

- Find all nodes connected by a path of length 2:

$\varphi(x,y) \equiv \exists u.(R(x,u) \wedge R(u,y))$

- Find all nodes without outgoing edges:

$\varphi(x) \equiv \exists u.(R(u,x) \wedge \forall y.\neg R(x,y))$

These are *open formulas*

16

In Class

- Retrieve all nodes with at least two children
- A node x is more important than y if every child of y is also a child of x. Retrieve all 'most important nodes' in the graph

17

FO in Databases

FO	**Databases**
Vocabulary: $R_1, ..., R_n$	Database schema: $R_1, ..., R_n$
Model: $\mathbf{D} = (D, R_1^D, \ldots, R_k^D)$	Database instance: $\mathbf{D} = (D, R_1^D, \ldots, R_k^D)$
Sentences are *true* or *false*	Formulas compute *queries*

18

FO Semantics

- In FO we express WHAT we want
- Sometimes it's even unclear HOW to get it
- See accompanying slides on FO semantics
  - They explain HOW to get it, but it's impractical

19

Relational Algebra

- An algebra over relations
- Five operators:
  $\cup$, -, $\times$, σ, Π
- Meaning:

$R_1 \cup R_2$ = set union
$R_1 - R_2$ = set difference
$R_1 \times R_2$ = cartesian product
$\sigma_c(R)$ = subset of tuples satisfying condition c
$\Pi_a(R)$ = projection on the attributes in a

20

FO $\rightarrow$ RA

$\varphi(x) \equiv R(x,x) \quad \rightarrow \quad \Pi_1(\sigma_{1=2}(R))$

$\varphi(x,y) \equiv \exists z.\exists u.(R(x,z) \wedge R(z,u) \wedge R(u,y)) \quad \rightarrow \quad \Pi_{16}(\sigma_{2=3 \wedge 4=5}(R \times R \times R))$

$\Pi_{16}((R \text{ join}_{2=1} R) \text{ join}_{4=1} R))$

$\varphi(x) \equiv \forall y.R(x,y) \quad \rightarrow \quad ?$

WHAT $\rightarrow$ HOW

21

FO v.s. RA

Theorem. Every query in RA can be expressed in FO

Proof

This shows how to go from HOW to WHAT
not very interesting

What about the converse ?

22

The Drinkers/Beers Example

- Vocabulary:

Likes(drinker,beer), Serves(bar,beer), Frequents(drinker,bar)

- Find all drinkers that frequent some bar that serve some beer that they like:

$\varphi(d) \equiv \exists ba. \exists be.(F(d,ba) \wedge L(d,be))$

23

Lots of Fun Examples (in class)

- Find drinkers that frequent some bar that serves only beer they like
- Find drinkers that frequent only bars that serve some beer they like
- Find drinkers that frequent only bars that serve only beer they like

24

Unsafe FO Queries

- Find all nodes that are not in the graph:

$$q(x) = \neg\exists y.R(x, y) \wedge \neg\exists z.R(z, x)$$

what's wrong ?

Unsafe FO Queries

- Find all nodes that are connected to "everything":

$$q(x) = \forall y.R(x, y)$$

what's wrong ?

Unsafe FO Queries

- Find all pairs of employees or offices:

$$q(x, y) = Emp(x) \vee Office(y)$$

what's wrong ?

- We don't want such queries !

Safe Queries

A model **D** = (D, R_1^D, …, R_k^D)

- In FO:
  - both D and R_1^D, …, R_k^D may be infinite
- In databases:
  - D may infinite (**int**, **string,** etc)
  - R_1^D, …, R_k^D are always finite
  - We call this a *finite model*

Safe Queries

- φ is a *finite* query if for every finite model **D**, φ(**D**) is finite
- φ is *safe*, or *domain independent*, if for every two models **D**, **D'** having the same relations:

  D = (D, R_1^D, …, R_k^D), **D'** = (D', R_1^D, …, R_k^D)

  we have φ(**D**) = φ(**D'**)
- If φ is safe then it is also finite (why ?)
- Note: book has different but equivalent definition

Safe Queries

- **Definition**. Given **D** = (D, R_1^D, …, R_k^D), the *active domain* is D_a = the set of all constants in R_1^D, …, R_k^D
- **Example**. Given a graph **D** = (D, R)
  $D_a = \{ x \mid \exists y.R(x,y) \vee \exists z.R(z,x)\}$
- **Property**. If a query is safe, it suffices to range quantifiers only over the active domain (why ?)
- Hence we can *compute* safe queries

Safe Queries

- The **safe relational calculus** consists only of safe queries. However:
- **Theorem** It is undecidable if a given a FO query is safe.
- Need to write only safe queries, but how do we know how which queries are safe ?
- Work around: write them in an obviously safe way
  - Range restricted queries - formally defined in [AHU]

31

FO v.s. RA

Theorem. Every safe query in FO can be expressed in RA

Proof

From WHAT to HOW
this is really interesting and motivated the relational model

32

Limited Expressive Power

- Vocabulary: binary relation R
- The following queries cannot be expressed in FO:
- Transitive closure:
  - $\forall x.\forall y.$ there exists $x_1, ..., x_n$ s.t. $R(x,x_1) \wedge R(x_1,x_2) \wedge ... \wedge R(x_{n-1},x_n) \wedge R(x_n,y)$
- Parity: the number of edges in R is even

33