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 :)

XML Storage

Shanmugasundaram’s paper:

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

XML Storage

Other ways:

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

OO Databases

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

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 (O₂)
  – Elegant extension of SQL
  – Later adopted by ODMG in the OQL language
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; }

Same in E/R

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

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)

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: $c_1, c_2, c_3, \ldots$
FO Syntax

- Terms (t) and FO formulas (ϕ) are:
  
  \[
  t ::= x \mid c \\
  ϕ ::= R(t_1, ..., t_a) \mid t_i = t_j \\
  \mid ϕ \land ϕ' \mid ϕ \lor ϕ' \mid ¬ϕ \\
  \mid ∀x.ϕ \mid ∃x.ϕ
  \]

FO Examples

Most interesting case:
Vocabulary = one binary relation R (encodes a graph)

FO Sentences

- Does there exist a loop in the graph?
  \[
  ϕ ≡ ∃x.R(x,x)
  \]

- Are there paths of length >2?
  \[
  ϕ ≡ ∃x.∃y.∃z.(R(x,y) \land R(y,z) \land R(z,x))
  \]

- Is there a "sink" node?
  \[
  ϕ ≡ ∃x.∀y.R(x,y)
  \]

FO Queries

- Find all nodes connected by a path of length 2:
  \[
  ϕ(x,y) ≡ ∃u.(R(x,u) \land R(u,y))
  \]

- Find all nodes without outgoing edges:
  \[
  ϕ(x) ≡ ∃u.(R(u,x) \land ∀y.¬R(x,y))
  \]

These are open formulas

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

FO in Databases

<table>
<thead>
<tr>
<th>FO</th>
<th>Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vocabulary: R₁, ..., Rₙ</td>
<td>Database schema: R₁, ..., Rₙ</td>
</tr>
<tr>
<td>Model: D = (D, R₁, ..., Rₙ)</td>
<td>Database instance: D = (D, R₁, ..., Rₙ)</td>
</tr>
<tr>
<td>Sentences are true or false</td>
<td>Formulas compute queries</td>
</tr>
</tbody>
</table>
**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

**Relational Algebra**

- An algebra over relations
- Five operators: \( \cup, -, \times, \sigma, \Pi \)
- Meaning:
  - \( R_1 \cup R_2 = \text{set union} \)
  - \( R_1 - R_2 = \text{set difference} \)
  - \( R_1 \times R_2 = \text{cartesian product} \)
  - \( \sigma_c(R) = \text{subset of tuples satisfying condition } c \)
  - \( \Pi_a(R) = \text{projection on the attributes in } a \)

**FO \rightarrow RA**

\[
q(x) = R(x,x) \quad \rightarrow \quad \Pi_1(\sigma_1(R))
\]

\[
q(x,y) = \exists a \exists b (R(x,a) \land R(y,b)) \quad \rightarrow \quad \Pi_{16}(\sigma_{2,3} \land \sigma_{1,4} (R \times R)) \land \Pi_{16} (R \text{ join } 1 = 2 \text{ join } 4 = 1 R)
\]

\[
q(x) \equiv \forall y R(x,y) \quad \rightarrow \quad \text{WHAT} \quad \rightarrow \quad \text{HOW}
\]

**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:
  \[
  q(d) = \exists b \exists c (F(d, ba) \land L(d, be))
  \]

**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?

**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
Unsafe FO Queries

• Find all nodes that are not in the graph:

\[ q(x) = \neg \exists y.R(x, y) \land \neg \exists z.R(z, x) \]

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:

\[ q(x, y) = \text{Emp}(x) \lor \text{Office}(y) \]

what’s wrong ?

• We don’t want such queries !

Safe Queries

A model \( D = (D, R_1^D, \ldots, R_k^D) \)

• In FO:
  – both \( D \) and \( R_1^D, \ldots, R_k^D \) may be infinite

• In databases:
  – \( D \) may infinite (int, string, etc)
  – \( R_1^D, \ldots, R_k^D \) are always finite
  – We call this a finite model

Safe Queries

• \( \varphi \) is a finite query if for every finite model \( D \), \( \varphi(D) \) is finite

• \( \varphi \) is safe, or domain independent, if for every two models \( D, D' \) having the same relations:

\[ D = (D, R_1^D, \ldots, R_k^D), \quad D' = (D', R_1^{D'}, \ldots, R_k^{D'}) \]

we have \( \varphi(D) = \varphi(D') \)

• If \( \varphi \) is safe then it is also finite (why ?)

• Note: book has different but equivalent definition

Safe Queries

• Definition. Given \( D = (D, R_1^D, \ldots, R_k^D) \), the active domain

is \( \text{D}_a = \{ x \mid \exists y.R(x, y) \lor \exists z.R(z, x) \} \)

• Example. Given a graph \( D = (D, R) \)

\( \text{D}_a = \{ x \mid \exists y.R(x, y) \lor \exists z.R(z, x) \} \)

• 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 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]

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

Limited Expressive Power

- Vocabulary: binary relation R
- The following queries cannot be expressed in FO:

- Transitive closure:
  - $\forall x, y. \text{there exists } s_1, \ldots, s_k \text{ s.t.}$
  - $R(x, s_1) \land R(s_1, s_2) \land \ldots \land R(s_{n-1}, s_n) \land R(s_n, y)$

- Parity: the number of edges in R is even