Lecture 7: Foundations of Query Languages

Tuesday, January 23, 2001

1

A History of DB Theory: In the Beginning...

- Up to 1970, a "database" was a file of records
  - COBOL/CODASYL
  - Network model, with low level navigational interface
- Codd proposed the relational model in 1970
  - Database = a first order structure
- This was a great vision; it took 10 years for the community to adopt it
- Today: relational databases heralded as major success of theory

2

The Golden Years

- The 80s: rich research work on foundations
- Relational model and algebra:
  - Theory of functional dependecies
  - Transaction processing
- Study other data models:
  - Complex objects, object oriented
- Study other query languages:
  - Query complexity ➔ descriptive complexity
- Study other applications:
  - Distributed query processing, semijoin reduction
  - Partial information

3

- But practical database interested only in:
  - *one* particular language (SQL)
  - one particular application (OLTP queries) and one particular architecture (client-server)
- Transaction processing = useful
- Functional dependencies = somewhat
- The rest = great but useless

4

Database Theory in the Web Age

- Sudden interest in changing everything
  - Web data is not relational: what is it ?
    - The XML-Schema has a few hundreds pages; how to understand it ?
  - New query languages are not relational algebra: what are they ?
    - W3C is designing a new XML query language; how to proceed ?
  - New architectures that are not client-server:
    - Distributed data, incomplete information, etc.

5

Our Goal

- Talk about fundamental concepts in the theory of the relational model and relational query languages:
- Use AHV's book liberally

6

First Order Logic

- Given:
  - a vocabulary, $R_1, \ldots, R_k$
  - An arity, $ar(R_i)$, for each i=1,…,k
  - an infinite supply of variables $x_1, x_2, x_3, \ldots$
- FO formulas, φ , are:

$$R_i(x_1,...,x_{ar(R_i)}), \quad x_i = x_j$$

$$\varphi \wedge \varphi', \ \varphi \vee \varphi', \neg\varphi$$

$$\exists x.\varphi, \ \forall x.\varphi$$

Sometimes we also allow constants

Examples of FO Formulas

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

$\exists y.(R(x,y) \wedge \neg R(y,x))$ x is a free variable

$\forall x.\forall y.(R(x,y) \Rightarrow \exists z.(R(x,z) \wedge R(z,y)))$

"a $\Rightarrow$ b" abbreviates as usual "¬a V b"

Bound and free variables defined the usual way

Models for FO

- Given a vocabulary $R_1, \ldots, R_k$
- A model is $D = (D, R_1, \ldots, R_k)$
  - D = a set, called *domain*, or *universe*
  - $R_i \subseteq D \times D \times \ldots \times D$, ($ar(R_i)$ times) i = 1,...,k
- The model is *finite* if $R_1, \ldots, R_k$ is finite
- E.g. D = int, while $R_1,\ldots,R_k$ are finite sets

Remarks

- Vocabulary $R_1, \ldots, R_k$ = database schema
- Model = database instance
- Abuse of notation: R_i and R_i
- Abuse of notation: D and D
- We are interested in *finite models,* but we will consider infinite models too, for a while

Meaning (Semantics) of FO formulas

- Given:
  - A formula φ , with free variables $x_1, \ldots, x_n$ (we write $\varphi(x_1,...,x_n)$)
  - A model $(D, R_1, \ldots, R_k)$
- We say that φ is true on $a_1, \ldots, a_n \in D$:
  - In notation: $D \models \phi(a_1, \ldots, a_n)$
  - Defined inductively (next)

Meaning of FO formulas

$D \models R_i(a_1,...,a_n)$, if $(a_1,...,a_n) \in R_i$

$D \models (x_i = x_j)$, if $a_i = a_j$

$D \models \varphi \wedge \varphi'[a_1...,a_n]$, if $D \models \varphi[a_1...,a_n]$ and $D \models \varphi'[a_1...,a_n]$

(similarly for OR and NOT)

$D \models \exists x.\varphi[a_1,...,a_k]$ if there exists $b \in D$ s.t. $D \models \varphi[a_1,...,a_k,b]$

$D \models \forall x.\varphi[a_1,...,a_k]$ if for all $b \in D$, $D \models \varphi[a_1,...,a_k,b]$

FO Formulas as Queries

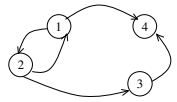
- Given:
  - A FO formula $\varphi(x_1,...,x_n)$
  - A (finite) model $D = (D, R_1, ..., R_k)$
- The *answer* of evaluating φ on D is:

$$\varphi(D) = \{(a_1,...,a_n) \mid D \models \varphi[a_1,...,a_n]\}$$

- Hence: an FO formula defines a *function* mapping a database to a relation

Examples of Formulas = Queries

Vocabulary: single relation R

D = 

R=

1	2
2	1
2	3
1	4
3	4

Graphs are the most "common" models

Examples of Formulas = Queries

$q_1 = \varphi(1, x)$ Notice: uses a constant, 1
Looks for successors of 1
Answer: $q_1(D) = \{2, 4\}$

$q_2 = \exists z.(\varphi(x, z) \wedge \varphi(z, y))$

Looks for pairs (x,y) connected by paths of length 2
Answer: $q_2(D) = \{(1,1), (2,2), (1,3), (2,4)\}$

$q_3 = \forall y.(R(x, y) \Rightarrow \exists z.(R(y, z))$

Answer: $q_3(D)=\{1\}$

Boolean Queries

A *boolean* query is one without free variables
Its answer is true or false

$$q_4 = \forall x.\forall y.((\exists u.R(x, u) \vee R(u, x)) \wedge \exists v.R(y, v) \vee R(v, y)) \Rightarrow R(x, y))$$

says that x and y are nodes in the graph

Tests for a clique

More Examples

- Vocabulary (= schema):
  - Employee(name, office, mgr), Manager(name, office)
- Queries:
  - Find offices: $q_1(x) = \exists y.\exists z.(E(y, x, z))$
  - Find offices with at least two employees:
    $q_2(x) = \exists y_1.\exists z_1.\exists y_2.\exists z_2.(E(y_1, x, z_1) \wedge E(y_2, x, z_2) \wedge \neg(y_1 = y_2))$
  - Find managers that share office with all their employees:
    $q_3(x) = \exists y.(M(x, y) \wedge \forall u.\forall v.(E(u, v, x) \Rightarrow y = v))$

Properties of Queries

- Decidable
- Generic
- Domain-independent

- They make more sense if we think of queries in general, not just FO queries
- Define next general queries

Queries

- A ***query,*** q, is a function from models to relations, s.t. for every model (D, R_1, ..., R_k):
  - $q(D, R_1, ..., R_k) = R$, s.t. $R \subseteq D^n$
- Here n is called the *arity* of q; when n=0, q is called a *boolean* query

19

Property 1: Decidable Queries

- q is *decidable* if there exists a Turing Machine that, for some encoding of D, given R_1, ..., R_k on its input tape, computes $q(D, R_1, ..., R_k)$

20

Property 2: Domain Independence

- In English
  - q only depends on R_1, ..., R_k, not on D !
  - Intuition: a database consists only of R_1, ..., R_k, not on D.
- Formally: a query q is *domain independent* if
  - for any model (D, R_1, ..., R_k)
  - for any set D' s.t. $R_1 \subseteq (D')^{ar(R1)}$, ..., $R_k \subseteq (D')^{ar(Rk)}$
  - the following holds
    - $q(D , R_1, ..., R_k) = q(D', R_1, ..., R_k)$

21

Property 2: Domain Independence

Examples:

- Queries that are domain independent:
  - "Find pairs of nodes connected by a path of length 2"
  - "Find the manager of Smith"
  - "Find the largest salary in the database"
- Queries that are not domain independent:
  - "Find all nodes that are not in the graph"
  - "Find the average salary"

22

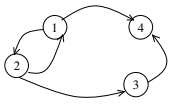
Property 3: Genericity

- In English:
  - q does not depend on the particular encoding of the database
- Formally:
  - for every $h:(D,R_1, ...,R_k) \rightarrow (D',R'_1, ...,R'_k)$
  - s.t. h=injective, $h(D) = D'$, $h(R_1)=R'_1$,..., $h(R_k)=R'_k$
  - It follows: $h(q(D ,R_1, ...,R_k)) = q(D',R'_1, ...,R'_k)$

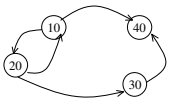
23

Property 3: Genericity

Example:

D = 

q(D)={1,3}

D'= 

q(D')= ??

24

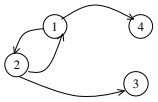
Property 3: Genericity

Examples:

- Queries that are generic:
  - "Find pairs of nodes connected by a path of length 2"
  - "Find all employees having the same office as their manager"
  - "Find all nodes that are not in the graph"
- Queries that are not generic:
  - "Find the manager of Smith"
    - we often relax the definition to allow this to be generic
    - *C-genericity*, for a set of constants C
  - "Find the largest salary in the database"

Property 3: Genericity

More example:

D = 

q(D)={4}

This query cannot be generic (why ?)

Back to FO Queries

1. All FO queries are computable
2. NOT All FO queries are domain independent
   - Why ? Next...
3. All FO queries are generic
   - In particular query on previous slide not expressible in FO

FO Queries and Domain Independence

- Find all nodes that are not in the graph:

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

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

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

- Find all pairs of employees or offices:

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

- We don't want such queries !

FO Queries and Domain Independence

- Domain independent FO queries are also called *safe* queries
- Definition. The ***active domain*** of $(D, R_1, ..., R_k)$ is D_a = the set of all constants in $R_1, ..., R_k$
- E.g. for graphs, $D_a = \{x \mid \exists y.R(x, y) \vee \neg\exists z.R(z, x)\}$
- Very important:
  - If a query is safe, it suffices to range quantifiers only over the active domain (why ?)

FO Queries and Domain Independence

- The bad news:
  - **Theorem** It is undecidable if a given a FO query is safe.
- The good news:
  - no big deal
  - can define a subset of FO queries that we *know* are safe = range restricted queries (rr-query)
  - Any safe query is equivalent to some rr-query

Range-restriction

- Syntactic, rather ad-hoc definition (several exists):
- $R(x, x)$ OK, $\neg R(x, x)$ not OK
- $\exists y.(S(y) \wedge \neg R(y, y))$ OK, $\exists y.(\neg R(y, y))$ not OK
- $\forall y.(S(y) \Rightarrow \neg R(y, y))$ OK, $\forall y.(\neg R(y, y))$ not OK

- If a query q is safe, it is equivalent to a rr-query:

$\neg R(x, y,...)$ to $(x \in D_a \wedge y \in D_a \wedge ... \wedge \neg R(x, y,...))$

$\exists y.\varphi$ to $\exists y.(y \in D_a \wedge \varphi)$ $\forall y.\varphi$ to $\forall y.(y \in D_a \Rightarrow \varphi)$

31

FO = Relational Algebra

- Recall the 5 operators in the relational algebra:
  - U, -, x, σ, Π
- Theorem. A domain independent query is expressible in FO iff it is expressible in the relational algebra

32