ECS 165B: Database System Implementation
Lecture 7

UC Davis

April 13, 2011

Announcements

Several reminders:

1. DavisDB Part 1 due Sunday at 11:59pm
2. Don't forget about the writeup (`writeup.txt`) when submitting your code
3. Style counts! Keep your code clean, simple, readable ...

We'll have "code review" meetings next week (stay tuned for email)

DavisDB Part 2 (Index Manager) will be handed out Monday; due Sunday, 5/1 at 11:59pm

Agenda

- Last time - A taste of database theory, Part 1: relational algebra, relational calculus, and first-order logic
- Today - A taste of database theory, Part 2: containment and equivalence of conjunctive queries
- Reading: none

Recall from last time: relational calculus (RC)

- Database query language based on first-order logic
- **Syntax:** expressions of the form

  $$\{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$$

  where $\varphi(x_1, \ldots, x_n)$ is a first-order formula with free variables $x_1, \ldots, x_n$.
- **Semantics:** return all tuples $(a_1, \ldots, a_n)$ such that $\varphi(a_1, \ldots, a_n)$ is true in the database.

Example: relational calculus queries

Database with three relations: **Class**(classId, className, roomNo); **Student**(studentId, studentName); and **Takes**(studentId, classId).

- "Find all students taking a class meeting in Wellman 1"

$$\{(x) \mid \exists s \exists c \exists n\ \textbf{Student}(s, x) \wedge \textbf{Takes}(s, c) \\ \wedge \textbf{Class}(c, n, \text{"Wellman 1"})\}$$

- "Find all pairs of students not taking a class together"

$$\{(x, y) \mid \exists s \exists s'\ \textbf{Student}(s, x) \wedge \textbf{Student}(s', y) \wedge \\ \neg \exists c (\textbf{Takes}(s, c) \wedge \textbf{Takes}(s', c))\}$$

Review: main results from last time (1)

The first result from last time concerned expressiveness of SQL, relational algebra (RA), and relational calculus (RC).

Theorem

Relational algebra and (safe) relational calculus are expressively equivalent to each other and to the domain-independent fragment of first-order logic (FO).

Thus, even though relational algebra and relational calculus are syntactically very different, they are fundamentally two sides of the same coin.

Review: main results from last time (2)

The connection between RA and RC can be exploited, e.g., to show something about the fundamental problem of checking query equivalence:

Theorem

Given two relational algebra (or relational calculus) queries, it is undecidable to determine whether they are equivalent, i.e., agree on all database instances.

Episode IV: a New Hope

- Today we'll look at an important fragment of these query languages where equivalence is decidable: so-called conjunctive queries
- We'll also see some advanced query optimizations based on these results, useful for removing redundancy from queries

What is a conjunctive query?

In SQL: query that uses only `select-from-where`; no inequalities in `where` clause; no `union` or `difference`

```
select R.A, S.C
from R, S
where R.B = S.B
```

In RA: query that uses only $\pi, \sigma, \times$ (no $\cup, -$): e.g.,

$$\pi_{1,4}(\sigma_{2=3}(R \times S))$$

In RC: query whose formula is conjunctive (no $\vee, \neg$ or $\forall$): e.g.,

$$\{(x, z) \mid \exists z\ R(x, y) \wedge S(y, z)\}$$

The three definitions of conjunctive queries agree

Theorem

The conjunctive fragments of SQL, RA, and RC defined on the previous slide are all expressively equivalent. Also, one can easily convert a conjunctive query from SQL to RA, RA to RC, and RC to SQL.

What we'll look at: query containment and equivalence

Equivalence. Given queries Q, Q', is Q equivalent to Q'?
(Answers to Q and Q' are the same on all databases.)

This fundamental problem underlies advanced query optimizations, and has many other applications in databases. A related problem also of fundamental interest:

Containment. Given queries Q, Q', is Q contained in Q'?
(Answers to Q are always a subset of the answers to Q'.)

If we can check containment, then we can also check equivalence!

What we're going to see today

1. Containment (and equivalence) of conjunctive queries is decidable; complexity is NP-complete

2. Can minimize conjunctive queries, to eliminate redundancy

Why are conjunctive queries "easy"?

Key insight: the body of a conjunctive query

$$\{(x, y, z) \mid \exists u \; R(x, y) \wedge R(x, z) \wedge S(y, u, z)\}$$

can be viewed as a database!

$$R : \begin{array}{|cc|} \hline x & y \\ \hline x & z \\ \hline \end{array} \qquad S : \begin{array}{|ccc|} \hline y & u & z \\ \hline \end{array}$$

This is called the "canonical database" for the query. Here we've "frozen" the variables in the query, viewing them as ordinary (constant) values.

Using the canonical database

- We're given conjunctive queries

$$Q = \{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$$
$$Q' = \{(y_1, \ldots, y_m) \mid \psi(y_1, \ldots, y_m)\}$$

  We want to check whether Q is contained in Q'.

- It turns out that you can do the following:
  1. Take the canonical database for Q
  2. Evaluate query Q' on it
  3. See if $(x_1, \ldots, x_n)$ is in the answer!

  $(x_1, \ldots, x_n)$ will be in the answer iff Q is contained in Q'

Examples: checking containment (on board)

1.

$$Q = \{(u, v) \mid R(u, v)\}$$
$$Q' = \{(x, y) \mid \exists z\ R(x, y) \wedge R(x, z)\}$$

2.

$$Q = \{(u) \mid \exists v \exists w\ R(u, v) \wedge S(v, w)\}$$
$$Q' = \{(x) \mid \exists y \exists z\ R(x, y) \wedge R(x, z)\}$$

3.

$$Q = \{(u, v, w) \mid R(u, v) \wedge S(v, w)\}$$
$$Q' = \{(x, y, z) \mid R(x, y) \wedge S(x, z)\}$$

Complexity of checking query containment

Theorem

Checking containment of conjunctive queries is NP-complete

Proof.

(Sketch) Reduction from 3-coloring problem □

Note, queries in practice are usually small, so here is one place where NP-completeness isn't necessarily so bad.

An advanced optimization: query minimization

- We already saw an example of a query containing "redundancy:"

$$\{(x, y) \mid R(x, y) \wedge R(x, z)\}$$

- You probably wouldn't write such an inefficient query; but, your program might!
  - "Middleware" layers very common these days; use complicated automatically-generated queries.
- Can we systematically eliminate such redundancies?
- Yes! Using query minimization

How minimization works

Input: a conjunctive query $Q = \{(\bar{x}) \mid R_1(\bar{x}), \ldots, R_n(\bar{x})\}$

1. for i from 1 to n $\{$
2. let Q' be Q with $R_i(\bar{x})$ removed
3. check if Q' is contained in Q
4. if so, remove R_i from Q and continue
5. else, leave Q alone and continue
6. $\}$

Characterizing the result of minimization

Theorem

If Q, Q' are equivalent conjunctive queries, then minimizing Q and minimizing Q' will produce the same query (up to isomorphism)

"Up to isomorphism" just means up to renaming of variables

Summary

- We looked at an important fragment of SQL/RA/RC called conjunctive queries
- We saw that the fundamental problems of containment and equivalence are decidable (and NP-complete)
- We used this to derive a minimization procedure (eliminate redundancies from conjunctive queries)

Historical note: these results were first shown in a paper by Chandra and Merlin (1977) that helped get the nascent field of database theory off the ground