Database Theory: Conjunctive Queries & Static Analysis

CS 645
Mar 3, 2006
Life of a database theoretician

• Expressiveness of query languages
  – Any query in L1 can be expressed in L2
  – Query q cannot be expressed in L

• Complexity of languages
  – Bounds on resources required to evaluate any query in language L

• Static analysis of queries (for optimization)
  – Given q in L: is it minimal?
  – Given q1 and q2 in L: are they equivalent?

• Views
Coming lectures

• TODAY:
  – Overview of languages
  – Conjunctive queries (CQs)
  – Properties of CQs
  – Containment/equivalence for CQs

• Next Week
  – Adding recursion
  – Reasoning about views
Query languages

• So far we’ve seen:
  – Relational algebra
  – Relational calculus
  – SQL
Review: relational algebra

- Five operators:
  - Union: $\cup$
  - Difference: -
  - Selection: $\sigma$
  - Projection: $\Pi$
  - Cartesian Product: $\times$

- Derived or auxiliary operators:
  - Intersection, complement
  - Joins (natural, equi-join, theta join)
  - Renaming: $\rho$
Review: relational calculus

English: Name and sid of students who are taking the course "DB"

RA: $\Pi_{\text{name},\text{sid}} (\text{Students } \bowtie \bowtie \sigma_{\text{name} = "DB"} (\text{Course}))$

RC: $\{x_{\text{name}}, x_{\text{sid}} \mid \exists x_{\text{cid}} \exists x_{\text{term}} \text{Students}(x_{\text{sid}}, x_{\text{name}}) \land \text{Takes}(x_{\text{sid}}, x_{\text{cid}}) \land \text{Course}(x_{\text{cid}}, "DB", x_{\text{term}}) \}$
Review: SQL

Basic form:

```
SELECT attributes
FROM relations (possibly multiple, joined)
WHERE conditions (selections)
```
Query language classes

Expressiveness

Recursive Queries

FO queries

Conjunctive Queries

RA

(safe) RC

SFW +

UNION

EXCEPT

single
datalog
rule

Query language classes

Algebra

Logic

SQL
Conjunctive Queries

*abbreviated: CQ*

- A **subset** of FO queries (i.e. less expressive)
- Many queries in practice are conjunctive
- Some optimizers handle only conjunctive queries - break larger queries into many CQs
- CQ’s have “better” theoretical properties than arbitrary queries
Conjunctive Queries

in rule-based (datalog) notation

- **R**: Extensional database (EDB) - stored
- **P**: Intentional database (IDB) - computed

\[
P(x,z) \leftarrow R(x,y) \land R(y,z)
\]

- **Head**
- **Body**
- **Variables**
- **Subgoals**
- Implicit $\exists$
- Conjunction
- “IF”
Conjunctive Queries

Intuitively: when facts in the body are true of stored relations, then we infer the fact in the head

\[ P(x,z) :- R(x,y) \& R(y,z) \]

• More formally:
• Consider all possible substitutions: assignments of the variables in the body
Examples

EDB Relation: ManagedBy(emp,mgr)

\[ A(x) :- \text{ManagedBy(“Smith”,y) & ManagedBy(x,y)} \]

All employees having the same manager as “Smith”
Defining answers to CQ

- A substitution \( v \) is a function from variables into the domain. e.g. \( x \to a, y \to a, z \to b, u \to c \)
- Let \( I \) be an instance, i.e. relations \( I(R1) \ldots I(Rn) \)
- A tuple \( t \) is in the answer \( q(I) \) if there is a substitution \( v \) s.t:
  - \( v(u1) \in I(R1) \) for each \( i \), and
  - \( t = v(u) \)

General form of a CQ \( q \)

\[
\text{ans}(u) :- R1(u1) \land \ldots \land Rn(un)
\]
e.g. \( u_i = (x,y,z) \)

\( v(u_i) = (a,a,b) \)
Examples

EDB Relation: ManagedBy(emp,mgr)

• Find all employees having the same director as Smith:

A(x) :- ManagedBy("Smith",y), ManagedBy(y,z), ManagedBy(x,u), ManagedBy(u,z)
Query language classes

Recursive Queries

FO queries

Conjunctive Queries

RA

(safe) RC

SFW +
UNION
EXCEPT

RA:
\(\sigma, \pi, \times\)

single
data
golog
rule
CQ and RA

Relational Algebra:
• CQ correspond precisely to $\sigma_C$, $\Pi_A$, $\times$
  (missing: $\cup$, $-$)

$A(x) :-$ ManagedBy("Smith",y), ManagedBy(x,y)
Query language classes

FO queries

Recursive Queries

Conjunctive Queries

Expressiveness

Algebra
Logic
SQL

RA
(safe) RC
SFW +
UNION
EXCEPT

RA:
σ, π, ×
single datalog rule

S_dfW
CQ and SQL

Rule-based:

\[ A(x) \ :- \ \text{ManagedBy(“Smith”,y), ManagedBy(x,y)} \]

SQL:

```
select distinct m2.name
from ManagedBy m1, ManagedBy m2
where m1.name=“Smith” AND
      m1.manager=m2.manager
```
Boolean queries

\[ A() \rightarrow \text{ManagedBy(“Smith”, } x) , \text{ManagedBy(“Sally”, } x) \]

Is there someone who manages both Smith and Sally?

- **Returns:**
  - relation \( \{ (x) \} \) if the answer is yes
  - relation \( \{ \} \) if the answer is no
Properties of Conjunctive Queries

• Satisfiability
  – A query $q$ is **satisfiable** if there exists some input relation $I$ such that $q(I)$ is non-empty.
  – FACT: **Every CQ is satisfiable.**

• Monotonicity
  – A query $q$ is **monotonic** if for each instance $I,J$ over schema, $I \subseteq J$ implies $q(I) \subseteq q(J)$.
  – FACT: **Every CQ is monotonic.**
Satisfiability of CQs

We can always generate satisfying EDB relations from the body of the rule.

\[
S(x,y,z) \iff P(x,w) \land R(w,y,v) \land P(v,z)
\]

\[
\begin{align*}
S & \quad P & \quad R \\
\text{a} & \quad \text{b} & \quad \text{c} & \quad \text{d} & \quad \text{e} \\
\text{a} & \quad \text{b} & \quad \text{d} & \quad \text{e} & \quad \text{b} & \quad \text{c} & \quad \text{d}
\end{align*}
\]
**Monotonicity of CQs**

**general form of a CQ q**

\[
\text{ans}(u) :- R_1(u_1) \land \ldots \land R_n(u_n)
\]

\[\text{e.g. } u_i = (x,y,z)\]

- Consider two databases \( I, J \) s.t. \( I \subseteq J \).
- Let \( t \in q(I) \).
  - Then for some substitution \( v \):
    - \( v(u_i) \in I(R_i) \) for each \( i \).
    - \( t = v(u) \)
  - Since \( I \subseteq J \), \( v(u_i) \in J(R_i) \) for each \( i \)
  - So \( t \in q(J) \)
Consequence of monotonicity

Product (pname, price, category, maker)
Find products that are more expensive than all those produced by “Gizmo-Works”

```
SELECT name
FROM Product
WHERE price > ALL (SELECT price
                    FROM Purchase
                    WHERE maker='Gizmo-Works')
```

- This query is NOT monotone.
- Therefore, it is not in the class of conjunctive queries.
- It cannot be expressed as a simple SFW query.
Extensions of CQs
Query language classes

Expressiveness

Recursive Queries

FO queries

Conjunctive Queries

RA

(safe) RC

SFW +

UNION

EXCEPT

RA:

σ, π, x

single datalog rule

S_dFW

Algebra

Logic

SQL
Extensions of CQ: disequality

\[ \text{CQ} \neq \]

Find managers that manage at least 2 employees

A(y) :- ManagedBy(x,y), ManagedBy(z,y), x \neq z
Extensions of CQ: inequality

Find employees earning more than their manager

\[ A(y) :\text{- ManagedBy}(x,y), \text{Salary}(x,u), \text{Salary}(y,v), u > v \]

Additional EDB Relation: Salary(emp,money)
Extensions of CQ: negation

\[ \text{CQ}^- \]

*Find people sharing the same office with Alice, but with a different manager*

\[ A(y) :- \text{Office}(“Alice”,u), \text{Office}(y,u), \text{ManagedBy}(“Alice”,x), \neg\text{ManagedBy}(y,x) \]

Additional EDB Relation: Office(emp,officenum)
Extensions of CQ: union

UCQ
Unions of conjunctive queries

Rule-based:

\[
\begin{align*}
A(name) & : - \quad \text{Employee(name, dept, age, salary), age > 50} \\
A(name) & : - \quad \text{RetiredEmployee(name, address)}
\end{align*}
\]

Datalog notation is very convenient for expressing unions (no need for ∨)
Query language classes

- **FO queries**
  - Expressiveness
  - Recursive Queries
  - Conjunctive Queries

- **Algebra**
  - RA
  - (safe) RC

- **Logic**
  - SFW +
  - UNION
  - EXCEPT

- **SQL**
  - UCQ
  - CQ≤
  - CQ≠
  - CQ⁻
  - RA:
    - σ,π,x
  - single datalog rule
  - S^dFW
Extensions of CQ

• If we extend too much, we capture FO
  – Namely: CQs + Union, Negation
• Theoreticians need to be careful: small extensions may make a huge difference on certain theoretical properties of CQ
Query language classes

FO queries

Recursive Queries

Conjunctive Queries

Expressiveness

Algebra
Logic
SQL

RA
(safe) RC
UCQ⁻
SFW +
UNION
EXCEPT

UCQ
CQ<
CQ≠
CQ⁻

RA:
σ,π,x

single
datalog
rule

S^dFW
Query Equivalence and Containment

- One kind of static analysis
- Useful for query optimization
- Intensively studied since 1977
Query Equivalence

SELECT x.name, x.manager
FROM   Employee x, Employee y
WHERE  x.dept = 'Sales' and x.office = y.office
       and  x.floor = 5 and y.dept = 'Sales'

Hmmmm…. Is there a simpler way to write that?
Query Equivalence

- Queries $q_1$ and $q_2$ are **equivalent** if for every database $D$, $q_1(D) = q_2(D)$.

- Notation: $q_1 \equiv q_2$
Query Containment

- Query $q_1$ is **contained** in $q_2$ if for every database $D$, $q_1(D) \subseteq q_2(D)$.

- Notation: $q_1 \subseteq q_2$

- Obviously: $q_1 \subseteq q_2$ and $q_2 \subseteq q_1$ iff $q_1 \equiv q_2$

- Conversely: $q_1 \land q_2 \equiv q_2$ iff $q_1 \subseteq q_2$

We will study the containment problem only.
Sidenote:
containment for Boolean queries

- Recall: $q_1$ is **contained** in $q_2$ if for every database $D$, $q_1(D) \subseteq q_2(D)$.
  - if $q_1$, $q_2$ are boolean they return $\{ \langle \rangle \}$ or $\{ \}$
  - containment says:
    - whenever $q_1(D) = \{ \langle \rangle \}$ then $q_2(D) = \{ \langle \rangle \}$.
- Containment is implication: $q_1 \rightarrow q_2$
Examples of Query Containments

Is \( q_1 \subseteq q_2 \)?

\[
q_1(x) :\ R(x,y), \ R(y,z), \ R(z,w)
\]

\[
q_2(x) :\ R(x,y), \ R(y,z)
\]
Examples of Query Containments

Is $q_1 \subseteq q_2$?

\[
q_1(x) :- R(x,y), R(y,z), R(z,x)
\]

\[
q_2(x) :- R(x,y), R(y,x)
\]

Counter-example
Examples of Query Containments

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u,u)$

$q_2(x) :- R(x,u), R(u,v), R(v,w)$

Example

Diagram: A → B
Examples of Query Containments

Is $q_1 \subseteq q_2$?

$q_1(x) :\ - R(x,u), \ R(u,^{\text{"Smith"}})$
$q_2(x) :\ - R(x,u), \ R(u,v)$
Query Containment

• **Theorem** Query containment for CQ is decidable and NP-complete.
Checking containment

1. “Freeze” q1
   - Replace variables by unique constants
   - $x \rightarrow a_x$, $u \rightarrow a_y$
   - this is called canonical database of q1

2. Evaluate q2 on frozen body of q1

3. If frozen head is derived, then $q_1 \subseteq q_2$

$q_1(x) :- R(x,u), R(u,u)$
$q_2(x) :- R(x,u), R(u,v), R(v,w)$

Containment!
Why does this test work?

• If the test is negative, the canonical database constructed is a counterexample to containment.

• If the test is positive:
  – substitution $\nu$: var(q2) --> “canonical domain”
  – this implies $f$: var(q2) --> var(q1) $\cup$ const(q1)
  – Now suppose $t \in q1(I)$ for any instance $I$
    • there is substitution $w$: var(q1) --> domain
      – such that $t$ is derived.
    • then $f$ followed-by $w$ is a substitution showing that $t$ will be in $q2(I)$. 
Query Homomorphisms

- A **homomorphism** \( f : q_2 \rightarrow q_1 \) is a function
  \[ f: \text{var}(q_2) \rightarrow \text{var}(q_1) \cup \text{const}(q_1) \]
such that:
  - \( f(\text{body}(q_2)) \subseteq \text{body}(q_1) \)
  - \( f(t_{q_1}) = t_{q_2} \)

The Homomorphism Theorem \( q_1 \subseteq q_2 \) iff
there exists a homomorphism \( f : q_2 \rightarrow q_1 \)

\[
\begin{align*}
q_1(x) & : R(x,u), R(u,u) \\
q_2(x') & : R(x',u'), R(u',v'), R(v',w')
\end{align*}
\]

homomorphism \( f: \)
\[
\begin{align*}
x' & \rightarrow x \\
u' & \rightarrow u \\
v' & \rightarrow u \\
w' & \rightarrow u
\end{align*}
\]

Chandra & Merlin 1977
The Homeomorphism Theorem

• **Theorem** Conjunctive query containment is:
  1. decidable (why ?)
  2. in NP (why ?)
  3. NP-hard

• In short: containment for CQs is NP-complete