CSE 544
Theory of Query Languages

Outline

- Conjunctive queries
- Query containment and equivalence
- Query minimization

Conjunctive Queries (CQ)

- CQ = one datalog rule
- CQ = SELECT-DISTINCT-FROM-WHERE
- CQ = select/project/join (σ, Π, $\bowtie$) fragment of RA
- CQ = existential/conjunctive ($\exists$, $\wedge$) fragment of RC

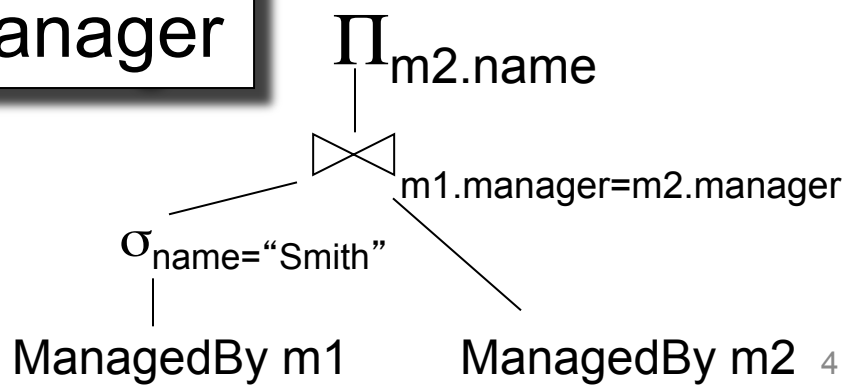
Notice: strictly speaking we are not allowed to use <, ≤, >, ≥, ≠ in CQ's. If we include these, then the language is called $CQ^{<}$, or $CQ^{\neq}$, etc.

Examples

Find all employees having the same manager as "Smith":

```
A(x) :-  ManagedBy("Smith",y),   ManagedBy(x,y)
```

```
SELECT DISTINCT m2.name
FROM ManagedBy m1,
          ManagedBy m2
WHERE m1.name="Smith"
   AND m1.manager=m2.manager
```



Examples

- Example of CQ

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

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

- Examples of non-CQ:

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

$$q(x) = T(x) \vee \exists z.S(x,z)$$

Query Equivalence and Containment

Needed in a variety of static analysis tasks:

- Query optimization
- Query rewriting using views
- Testing for semijoin reduction
- etc

Query Equivalence

Definition. Queries q_1 and q_2 are **equivalent** if for every database **D**, $q_1(\mathbf{D}) = q_2(\mathbf{D})$.

Notation: $q_1 \equiv q_2$

Query Containment

Definition. Query q_1 is **contained** in q_2 if for every database **D**, $q_1(\mathbf{D}) \subseteq q_2(\mathbf{D})$.

Notation: $q_1 \subseteq q_2$

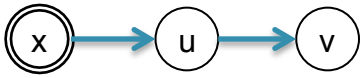
Fact: $q_1 \subseteq q_2$ and $q_2 \subseteq q_1$ iff $q_1 \equiv q_2$

We will study the containment problem only.

Examples of Query Containments

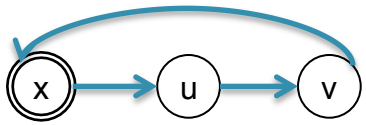
Is $q_1 \subseteq q_2$?

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

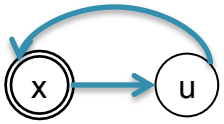


Examples of Query Containments

Is $q_1 \subseteq q_2$?



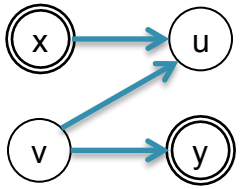
$q_1(x)$:- R(x,u), R(u,v), R(v,x)
$q_2(x)$:- R(x,u), R(u,x)



Examples of Query Containments

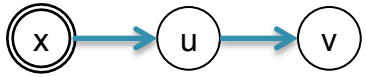
Is $q_1 \subseteq q_2$?

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

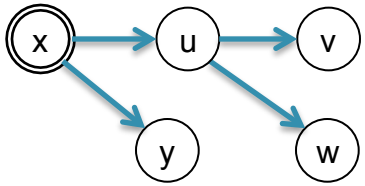


Examples of Query Containments

Is $q_1 \subseteq q_2$?

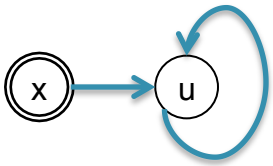


$q_1(x)$:- R(x,u), R(u,v)
$q_2(x)$:- R(x,u), R(x,y), R(u,v), R(u,w)



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)

Examples of Query Containments

Is $q_1 \subseteq q_2$?

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

Query Containment

Theorem The query containment and query equivalence problems for CQ are NP-complete.

Theorem The query containment and query equivalence problems for Relational Calculus are undecidable

Query Containment for CQ

There are two ways to test query containment $q_1 \subseteq q_2$

- Check if q_2 holds on the canonical database of q_1
- Check if there exists a homomorphism from $q_2 \rightarrow q_1$

Canonical Database

- **Canonical database** for q_1 is:

  $\mathbf{D_{q1}} = (D, R_1^D, \ldots, R_k^D)$

  - D = all variables and constants in q_1
  - $R_1^D, \ldots, R_k^D$ = the body of q_1

- **Canonical tuple** for q_1 is:

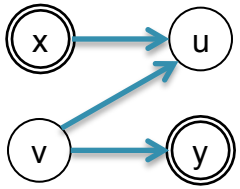
  t_{q1} = the head of q_1

Example

q1(x,y) :- R(x,u),R(v,u),R(v,y)

- Canonical database: $\mathbf{D}_{q1} = (D, R^D)$
  - D={x,y,u,v}
  - R^D =

x	u
v	u
v	y



- Canonical tuple: $t_{q1} = (x,y)$

Example

q1(x) :- R(x,u), R(u,”Smith”), R(u,”Fred”), R(u, u)

- $\mathbf{D}_{q1}$ = (D, R)
  - D={x,u,”Smith”,”Fred”}
  - R =

x	u
u	“Smith”
u	“Fred”
u	u

- t_{q1} = (x)

Checking Containment Using the Canonical Database

Theorem: $q_1 \subseteq q_2$ iff $t_{q1} \in q_2(\mathbf{D}_{q1})$

Example:

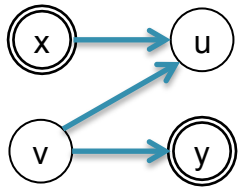
$q_1(x,y)$:- R(x,u),R(v,u),R(v,y)

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

- D={x,y,u,v}
- R = t_{q1} = (x,y)

- Yes, $q_1 \subseteq q_2$

x	u
v	u
v	y



Query Homomorphisms

- A ***homomorphism*** $f : q_2 \rightarrow q_1$ is a function $f: var(q_2) \rightarrow var(q_1) \cup const(q_1)$ such that:
  - $f(body(q_2)) \subseteq body(q_1)$
  - $f(t_{q1}) = t_{q2}$

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

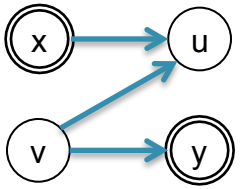
Example of Query Homeomorphism

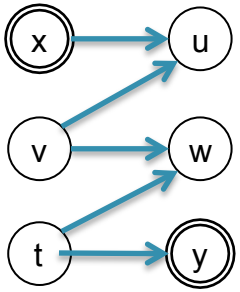
$var(q_1) = \{x, u, v, y\}$

$var(q_2) = \{x, u, v, w, t, y\}$

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

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





Therefore $q_1 \subseteq q_2$

Example

$var(q_1) \cup const(q_1) = \{x, u, \text{“Smith”}\}$

$var(q_2) = \{x, u, v, w\}$

$q_1(x)$:- R(x,u), R(u,”Smith”), R(u,”Fred”), R(u, u)

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

Therefore $q_1 \subseteq q_2$

The Complexity

Theorem Checking containment of two CQ queries is NP-complete

Proof

Reduction from 3SAT
Given a 3CNF Φ
Step 1:
  construct q1 independently of Φ.
Step 2:
  construct q2 from Φ.
Prove:
  there exists a
  homomorphism q2→q1
  iff Φ is satisfiable

Example:

$$\Phi = (\neg X_3 \vee \neg X_1 \vee X_4)$$
$$(X_1 \vee X_2 \vee X_3)$$
$$(\neg X_2 \vee \neg X_3 \vee X_1)$$

Proof: Step 1

Constructing q1

There are four types of clauses in any 3SAT:

Type 1 = ¬X ∨ ¬Y ∨ ¬Z
Type 2 = ¬X ∨ ¬Y ∨ Z
Type 3 = ¬X ∨ Y ∨ Z
Type 4 = X ∨ Y ∨ Z

For each type, q1 contains one relation with all 7 "satisfying assignments", where u=0, v=1

R1 (misses v,v,v)

u	u	u
u	u	v
u	v	u
u	v	v
v	u	u
v	u	v
v	v	u

R2 (misses v,v,u)

u	u	u
u	u	v
u	v	u
u	v	v
v	u	u
v	u	v
v	v	v

R3 (misses v,u,u)

u	u	u
u	u	v
u	v	u
u	v	v
v	u	v
v	v	u
v	v	v

R4 (misses u,u,u)

u	u	v
u	v	u
u	v	v
v	u	u
v	u	v
v	v	u
v	v	v

So what are the atoms in q1?

Proof: Step 2

Constructing q2

q2 has one atom for each clause in Φ:

- Relation name is R1, or R2, or R3, or R4
- The variables are the same as those in the clause

Example: $\Phi = (\neg X_3 \vee \neg X_1 \vee X_4) \wedge (X_1 \vee X_2 \vee X_3) \wedge (\neg X_2 \vee \neg X_3 \vee X_1)$

$q_2 = R_2(x_3,x_1,x_4),\ R_4(x_1,x_2,x_3),\ R_2(x_2,x_3,x_1)$

Proof

1. Suppose there is a satisfying assignment for Φ: it maps each X_i to either 0 or 1
   - Define the function f: Vars(q2)→ Vars(q1):
     - if $X_i = 0$ then $f(x_i) = u$
     - if $X_i = 1$ then $f(x_i) = v$
   - Then f is a homomorphism f : q2 → q1 (why??)
2. Suppose there exists a homomorphism f:q2→q1.
   - Define the following assignment:
     - if $f(x_i) = u$ then $X_i = 0$
     - if $f(x_i) = v$ then $X_i = 1$
   - This is a satisfying assignment for Φ (why??)

Beyond CQ

- Containment for arbitrary relational queries is undecidable.
- In fact, any static analysis on relational queries is undecidable, much like Rice's undecidability theorem for Turning machines
- All these results for relational queries follow from Trakthenbrot's theorem
- Will illustrate next

Trakhtenbrot's Theorem

Definition A sentence φ, is called *finitely satisfiable* if there exists a finite database instance D s.t. D |= φ

Satisfiable:
∃x.∃y.∀z.(R(x,z) → R(y,z))
∃x.∃y.T(x) ∨ ∃z.S(x,z)

Unsatisfiable:
∀x.∀y.∀z.(R(x,y)∧R(x,z) → y=z)
∧∃y. ∀x. not R(x,y)

Theorem The following problem is undecidable:
Given FO sentence φ, check if φ is finitely satisfiable

Query Containment

Theorem Query containment for Relational Calculus is undecidable

Proof: By reduction from the finite satisfiability problem:

Given a sentence φ, define two queries:

$q1(x) = R(x) \wedge \varphi$, and $q2(x) = R(x) \wedge x \neq x$

Then $q1 \subseteq q2$ iff φ is not finitely satisfiable

Containment for extensions of CQ

- CQ -- NP complete
- $CQ^{\neq}$ -- ??
- $CQ^{<}$ -- ??
- UCQ -- ??
- Relational Calculus -- undecidable

Query Containment for UCQ

$q_1 \cup q_2 \cup q_3 \cup \ldots. \subseteq q_1' \cup q_2' \cup q_3' \cup \ldots.$

Notice: $q_1 \cup q_2 \cup q_3 \cup \ldots. \subseteq q$ iff
$q_1 \subseteq q$ and $q_2 \subseteq q$ and $q_3 \subseteq q$ and ….

Theorem $q \subseteq q_1' \cup q_2' \cup q_3' \cup \ldots.$ Iff there exists some k such that $q \subseteq q_k'$

It follows that containment for UCQ is decidable, NP-complete.

Query Containment for $CQ^{<}$

$q_1() :- R(x,y), R(y,x)$
$q_2() :- R(x,y), x \leq y$

$q_1 \subseteq q_2$ although there is no homomorphism !

To check containment do this:
-Consider all possible orderings of variables in q1
-For each of them check containment of q1 in q2
-If all hold, then $q_1 \subseteq q_2$

Still decidable, but harder than NP: now in Π^p_2

Query Minimization

Definition A conjunctive query q is minimal if for every other conjunctive query q' s.t. $q \equiv q'$, q' has at least as many predicates ('subgoals') as q

Are these queries minimal ?

q(x) :- R(x,y), R(y,z), R(x,x)

q(x) :- R(x,y), R(y,z), R(x,'Alice')

Query Minimization

- Query minimization algorithm

  Choose a subgoal g of q
  Remove g: let q' be the new query
  We already know q ⊆ q' (why ?)
  If q' ⊆ q then permanently remove g

- Notice: the order in which we inspect subgoals doesn't matter

Query Minimization In Practice

- No database system today performs minimization !!!
- Reason:
  - It's hard (NP-complete)
  - Users don't write non-minimal queries
- However, non-minimal queries arise when using views intensively

Query Minimization for Views

```
CREATE VIEW HappyBoaters

   SELECT DISTINCT E1.name, E1.manager
   FROM Employee E1, Employee E2
   WHERE E1.manager = E2.name
         and E1.boater= 'YES'
         and E2.boater= 'YES'
```

This query is minimal

Query Minimization for Views

Now compute the Very-Happy-Boaters

```
SELECT DISTINCT  H1.name
FROM HappyBoaters H1, HappyBoaters H2
WHERE H1.manager = H2.name
```

This query is also minimal

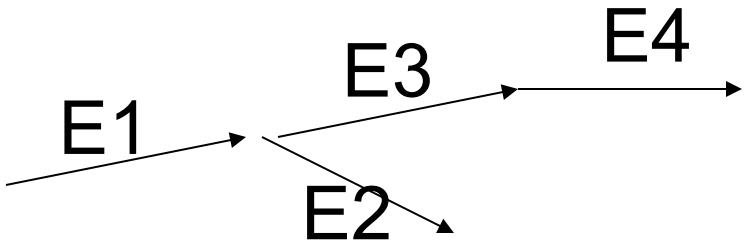
What happens in SQL when we run a query on a view ?

Query Minimization for Views

View Expansion

```
SELECT DISTINCT  E1.name
FROM Employee E1, Employee E2, Employee E3, Empolyee E4
WHERE E1.manager = E2.name and E1.boater = 'YES' and E2.boater = 'YES'
    and E3.manager = E4.name and E3.boater = 'YES' and E4.boater = 'YES'
    and E1.manager = E3.name
```

This query is no longer minimal !



E2 is redundant