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

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

Basic form:

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

FO queries
- RA
- (safe) RC
- SFW +
  - UNION
  - EXCEPT

Recursive Queries
- single datalog rule

Conjunctive Queries
- Algebra
- Logic
- SQL

Expressiveness
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) ← R(x,y) & R(y,z)
```

**Variables**

**Subgoals**

**Head**

**Body**

“IF”

Conjunction

Implicit $\exists$
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) :- 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. \( x \rightarrow a, y \rightarrow a, z \rightarrow b, u \rightarrow c \)

Let \( I \) be an instance, i.e. relations \( I(R_1) \ldots I(R_n) \)

A tuple \( t \) is in the answer \( q(I) \) if there is a substitution \( v \) s.t:
- \( v(u_1) \in I(R_1) \) for each \( i \), and
- \( t = v(u) \)

General form of a CQ \( q \):
\[
\text{ans}(u) \ :- \ R_1(u_1) \ & \ldots \ & R_n(u_n)
\]
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:

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

(Your director is your manager’s manager)
Query language classes

- FO queries
- Recursive Queries
- Conjunctive Queries

RA:
- $\sigma$, $\pi$, $\times$
- single datalog rule

(safe) RC

SFW +
- UNION
- EXCEPT

Algebra
Logic
SQL

Expressiveness
CQ and RA

Relational Algebra:

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

$$A(x) :- \text{ManagedBy(“Smith”,y), ManagedBy(x,y)}$$
Query language classes

Conjunctive Queries: $\sigma, \pi, \times$

RA: single datalog rule

(safe) RC

Recursive Queries

RA

SFW +

UNION

EXCEPT

SQL
CQ and SQL

Rule-based:

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

SQL:

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

Notice “distinct”
Boolean queries

\[ A() \text{ :- } \text{ManagedBy("Smith", } x\text{), ManagedBy("Sally", } x\text{)} \]

Is there someone who manages both Smith and Sally?

- **Returns:**
  - relation \{ \{ \} \} 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.

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

\[
\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{b} \\
\text{c} \\
\text{d} \\
\text{d} \\
\text{e}
\end{array}
\]
Monotonicity of CQs

general form of a CQ q

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

- Consider two databases I, J s.t. I \subseteq J.
- let \( t \in q(I) \).
  - Then for some substitution \( v \):
    - \( v(ui) \in I(Ri) \) for each \( i \).
    - \( t = v(u) \)
  - Since I \subseteq J, \( v(ui) \in J(Ri) \) 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=ʼGizmo-Worksʼ)
```

- 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

- FO queries
  - Conjunctive Queries: RA, $\sigma, \pi, \times$
  - Recursive Queries: (safe) RC

- Algebra
- Logic
- SQL

- RA: single datalog rule
- SFW + UNION EXCEPT
Extensions of CQ: disequality

\[ \text{CQ} \neq \]

Find managers that manage at least 2 employees

\[ A(y) :- \text{ManagedBy}(x,y), \text{ManagedBy}(z,y), x \neq z \]
Extensions of CQ: inequality

\[
\text{CQ<}
\]

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

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

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

UCQ
Unions of conjunctive queries

Rule-based:

A(name) :- Employee(name, dept, age, salary), age > 50
A(name) :- RetiredEmployee(name, address)

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

Recursive Queries

FO queries

Expressiveness

Conjunctive Queries

RA: $\sigma, \pi, \times$

(safe) RC

SFW +

SQL

Union

EXCEPT

UCQ

CQ<

CQ≠

CQ−

RA: single datalog rule

SdFW
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

FO queries

Conjunctive Queries

RA: $\sigma, \pi, \times$

UCQ

UCQ $^-$

CQ

CQ $^<$

CQ $^\neq$

CQ $^-$

RA: (safe) RC

SFW +

UNION

EXCEPT

SdFW

single datalog rule
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 $q_1 \subseteq q_2$?

$q_1(x) \leftarrow R(x,u), R(u,u)$

$q_2(x) \leftarrow R(x,u), R(u,v), R(v,w)$

Example
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** Query containment for CQ is decidable and NP-complete.

(query complexity)
Checking containment

1. “Freeze” $q_1$
   - Replace variables by unique constants
   - $x \rightarrow a_x$, $u \rightarrow a_y$
   - this is called canonical database of $q_1$

2. Evaluate $q_2$ on frozen body of $q_1$

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

$$q_1(x) :\text{-} R(x,u), R(u,u)$$

$$q_2(x) :\text{-} R(x,u), R(u,v), R(v,w)$$

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 $\mathbf{v}: \text{var}(q2) \rightarrow \text{"canonical domain"}$
  - this implies $\mathbf{f}: \text{var}(q2) \rightarrow \text{var}(q1) \cup \text{const}(q1)$
  - Now suppose $t \in q1(I)$ for any instance $I$
    - there is substitution $\mathbf{w}: \text{var}(q1) \rightarrow \text{domain}$
      - such that $t$ is derived.
    - then $\mathbf{f}$ followed-by $\mathbf{w}$ is a substitution showing that $t$ will be in $q2(I)$.  

Query Homomorphisms

- A **homomorphism** $f : \mathbf{q}_2 \rightarrow \mathbf{q}_1$ is a function $f : \text{var}(\mathbf{q}_2) \rightarrow \text{var}(\mathbf{q}_1) \cup \text{const}(\mathbf{q}_1)$ such that:
  - $f(\text{body}(\mathbf{q}_2)) \subseteq \text{body}(\mathbf{q}_1)$
  - $f(\mathbf{t}_{\mathbf{q}_1}) = \mathbf{t}_{\mathbf{q}_2}$

The Homomorphism Theorem $\mathbf{q}_1 \subseteq \mathbf{q}_2$ iff there exists a homomorphism $f : \mathbf{q}_2 \rightarrow \mathbf{q}_1$

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
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 $q \subseteq q'$ (why?)
If $q' \subseteq q$ then permanently remove $g$

• Notice: the order in which we inspect subgoals doesn’t matter
Other containment problems

• Extensions of CQs:
  – Unions of CQs
  – CQs with inequality

• FO queries
• Containment under constraints
• What about bags?
  – strange things happen
Containment under constraints

- Recall: query $q_1$ is **contained** in $q_2$ if for every database $D$, $q_1(D) \subseteq q_2(D)$.
- What if we know more about our input databases?
- Replace “every database $D$”, with:
  - “every database satisfying constraint $C$”

Containment under FD is NP-complete
Containment for FO queries

• **Theorem** Satisfiability for FO queries is undecidable

• **Lemma** Query containment/equivalence for FO is undecidable
  • if we had an algorithm for equivalence, we could use it to decide satisfiability of q:
    • check: $q \equiv \text{false}$

Consequence: we cannot do global query optimization for first-order queries.
Review

• CQs are an important fragment of FO
  – Equivalences: RA: $σ,π,\times$  SQL: $S^dFW$
  – Properties: satisfiable, monotonic
  – containment/equivalence decidable, NPC

• Expressiveness
  – CQs strictly less expressive than FO

• Hardness of static optimization