Foundations of Databases

Query Processing and Optimization

(Slides adapted from Thomas Eiter, Leonid Libkin and Werner Nutt)

Query Processing and Optimization

- *Query optimization*: finding a good way to evaluate a query
- Queries are declarative, and can be translated into procedural languages in more than one way
- Hence one has to choose the best (or at least good) procedural query
- This happens in the context of *query processing*
- A query processor turns queries and updates into sequences of operations on the database

Query Processing and Optimization Stages

- Queries are translated into an extended relational algebra (operator + execution method): Which algebra expressions will allow for an efficient execution?
- Algebraic operators can be implemented by different methods (Examples?): Which algorithm should be used for each operator?
- How do operators pass data (write into main memory, write on disk, pipeline into other operators)?

Issues:

- Translate the query into extended relational algebra ("query plans")
- Esimate the execution of the plans: We need to know how data is stored, how it accessed, how large are intermediate results, etc.

Decisions are based on general guidelines and statistical information

Overview of Query Processing

- Start with a declarative query:

```
SELECT R.A, S.B, T.E
FROM R,S,T
WHERE R.A>5 AND S.B<3 AND T.D=T.E
```

- Translate into an algebra expression:

$$\pi_{R.A,S.B,T.E}(\sigma_{R.A>5 \wedge S.B<3 \wedge T.D=T.E}(R \bowtie S \bowtie T))$$

- Optimization step: rewrite to an equivalent but more efficient expression:

$$\pi_{R.A,S.B,T.E}(\sigma_{A>5}(R) \bowtie \sigma_{B<3}(S) \bowtie \sigma_{D=E}(T)))$$

Why may this be more efficient? Is there a still more efficient plan?

Optimization by Algebraic Equivalences

- Given a relational algebra expression E, find another expression E' equivalent to E that is easier (faster) to evaluate.
- Basic question: Given two relational algebra expressions E_1, E_2, are they equivalent?
- If there were a method to decide equivalence, we could turn it into one to decide whether an expression E is empty, i.e., whether $E(\mathbf{I}) = \emptyset$ for every instance $\mathbf{I}$. *How does one show this?*
- Problem: Testing whether $E \equiv \emptyset$ is undecidable for expressions of full relational algebra i.e., including union and difference.
- Good news:

  We can still list some useful equalities.

  Equivalence (and emptiness testing) is decidable for important classes of

queries (SPJR, SPC).

Optimization by Algebraic Equivalences (cntd)

Systematic way of query optimization: Apply equivalences

- $\bowtie$ and $\times$ are commutative and associative, hence applicable in any order
- Cascaded projections can be simplified: If the attributes $A_1, \ldots, A_n$ are among $B_1, \ldots, B_m$, then

$$\pi_{A_1,\ldots,A_n}(\pi_{B_1,\ldots,B_m}(E)) \;=\; \pi_{A_1,\ldots,A_n}(E)$$

- Cascaded selections might be merged:

$$\sigma_{C_1}(\sigma_{C_2}(E)) \;=\; \sigma_{C_1 \wedge C_2}(E)$$

- Commuting selection with join. If c only involves attributes from E_1, then

$$\sigma_C(E_1 \bowtie E_2) \;=\; \sigma_C(E_1) \bowtie E_2$$

Optimization by Algebraic Equivalences (contd)

- Rules combining σ, π with $\cup$ and $\setminus$.
  - Commuting selection and union:

$$\sigma_C(E_1 \cup E_2) \;=\; \sigma_C(E_1) \cup \sigma_C(E_2)$$

  - Commuting selection and difference:

$$\sigma_C(E_1 \setminus E_2) \;=\; \sigma_C(E_1) \setminus \sigma_C(E_2)$$

  - Commuting projection and union:

$$\pi_{A_1,\ldots,A_n}(E_1 \cup E_2) \;=\; \pi_{A_1,\ldots,A_n}(E_1) \cup \pi_{A_1,\ldots,A_n}(E_2)$$

  - Question: what about projection and difference?
    Is $\pi_A(E_1 \setminus E_2)$ equal to $\pi_A(E_1) \setminus \pi_A(E_2)$?
- etc

Optimization of Conjunctive Queries

- Reminder:

  Conjunctive queries
  = SPJR queries
  = simple SELECT-FROM-WHERE SQL queries
  (only AND and (in)equality in the WHERE clause)

- Extremely common, and thus special optimization techniques have been developed
- Reminder: for two relational algebra expressions E_1, E_2,
  "$E_1 = E_2$" is undecidable.
- But for conjunctive queries, even $E_1 \subseteq E_2$ is decidable.
- Main goal of optimizing conjunctive queries: reduce the number of joins.

Optimization of Conjunctive Queries: An Example

- Given a relation R with two attributes A, B
- Two SQL queries:

Q1

```
SELECT  DISTINCT R1.B, R1.A
FROM    R R1, R R2
WHERE   R2.A=R1.B
```

Q2

```
SELECT  DISTINCT R3.A, R1.A
FROM    R R1, R R2, R R3
WHERE   R1.B=R2.B AND R2.B=R3.A
```

- Are they equivalent?
- If they are, we can save one join operation.
- In relational algebra:

$$Q_1 = \pi_{2,1}(\sigma_{2=3}(R \times R))$$

$$Q_2 = \pi_{5,1}(\sigma_{2=4 \wedge 4=5}(R \times R \times R))$$

Optimization of Conjunctive Queries (contd)

- We will show that Q_1 and Q_2 are equivalent!
- We cannot do this by using our equivalences for relational algebra expression. *(Why?)*
- Alternative idea: CQs are like patterns that have to be matched by a database. If a CQ returns an answer over a db, a part of the db must "look like" the query.
- Use rule based notation to find a representation of part of the db:

$$\begin{aligned} Q_1(x,y) &\;:\!-\; R(y,x), R(x,z) \\ Q_2(x,y) &\;:\!-\; R(y,x), R(w,x), R(x,u) \end{aligned}$$

Containment is a Key Property

Definition. (Query containment) A query Q is *contained* in a query Q' (written $Q \subseteq Q'$) if $Q(\mathbf{I}) \subseteq Q'(\mathbf{I})$ for every database instance $\mathbf{I}$.

Translations:

- If we can decide containment for a class of queries, then we can also decide equivalence.
- If $\mathcal{Q}$ is a class of queries that is closed under intersection, then the containment problem for $\mathcal{Q}$ can be reduced to the equivalence problem for $\mathcal{Q}$.

Checking Containment of Conjunctive Queries: Ideas

- We want to check containment of two CQs Q, Q'
- We consider CQs without equalities and inequalities
- We also assume that Q and Q' have the same arity and identical vectors of head variables, that is, they are defined as $Q(\vec{x})$:– B, $Q'(\vec{x})$:– B'
- Intuition: a conjunctive query $Q(\vec{x})$:– B returns an answer over instance $\mathbf{I}$ if $\mathbf{I}$ "matches the pattern B"
- Instead of checking containment over all instances, we consider only finitely many test instances (or better, one!)

Tableau Notation of Conjunctive Queries

- Tableaux notation blurs the distinction between query and database instance
- We first consider queries over a single relation

$$Q_1(x, y) \ :\!-\ R(y, x), R(x, z)$$

$$Q_2(x, y) \ :\!-\ R(y, x), R(w, x), R(x, u)$$

- Tableaux:

A	B	
y	x	
x	z	
x	y	⟵ answer line

A	B	
y	x	
w	x	
x	u	
x	y	⟵ answer line

- Variables in the answer line (or summary) are called "distinguished"

Tableau Homomorphisms

- Tableaux (as well as database instances or first order structures) are compared by homomorphisms
- **Idea:** T' is more general than T if there is a homomorphism from T' to T
- **Reminder**: *Terms* are variables or constants
- A homomorphism δ from Tableau T_1 to tableau T_2 is a mapping

$$\delta\colon \{\text{variables of } T_1\} \rightarrow \{\text{terms of } T_2\}$$

  such that
  - $\delta(x) = x$ for every distinguished x
  - if $(t_1, ..., t_k)$ is a row in T_1, then $(\delta(t_1), ..., \delta(t_k))$ is a row in T_2, where δ is extended to constants such that $\delta(a) = a$ for every constant a

Tableaux for Queries with Multiple Relations

- So far we assumed that there is only one relation R, but what if there are many?
- Construct a tableau for the query:

$$Q(x, y) \text{:–} S(x, y), R(y, z), R(y, w), R(w, y)$$

- We create rows for each relation:

S:

A	B
x	y

R:

A	B
y	z
y	w
w	y

x y

- Formally, a tableau is just a database where variables can appear in tuples, plus a set of distinguished variables.

Tableaux Homomorphisms: General Case

- Let T_1, T_2 be tableaux with the same distinguished variables
- A homomorphism δ from T_1 to T_2 is a mapping

  $$\delta\colon \{\text{variables of } T_1\} \;\longrightarrow\; \{\text{terms of } T_2\}$$

  such that
  - $\delta(a) = a$ for every constant
  - $\delta(x) = x$ for every distinguished variable
  - if $\vec{t}$ is a row of R in T_1, then $\delta(\vec{t})$ is a row of R in T_2, for every relation R

The Homomorphism Theorem for Tableaux

Homomorphism Theorem: Let Q, Q' be two conjunctive queries without equalities and inequalities that have the same distinguished variables and let T, T' be their tableaux. Then

$$Q \subseteq Q' \;\Leftrightarrow\; \text{there exists a homomorphism from } T' \text{ to } T$$

Proof: Necessity

- Assume that $Q \subseteq Q'$
- Note that T, ignoring the declaration of distinguished variables, can be seen as a database instance
- Note also that $\vec{x} \in Q(T)$
- Since $Q \subseteq Q'$, it follows that $\vec{x} \in Q'(T)$
- There is a valuation ν such
  - ν satisfies Q' over T
  - $\nu(\vec{x}) = \vec{x}$
- Thus, defining $\delta(y) = \nu(y)$ for every variable y in Q', we obtain a homomorphism from T' to T

Proof: Sufficiency

- Assume there is a homomorphism δ from T' to T. We will show that $Q \subseteq Q'$
- Let $\mathbf{I}$ be a db instance. We show that $Q(\mathbf{I}) \subseteq Q'(\mathbf{I})$
- Let $\vec{a} \in Q(\mathbf{I})$. We show that $\vec{a} \in Q'(\mathbf{I})$
- There is a valuation ν: $\{$ variables of $Q\,\} \rightarrow \mathbf{dom}(\mathbf{I})$ such that
  - ν satisfies B, the body of Q

    (that is, "$\nu(B) \subseteq \mathbf{I}$" in logic programming perspective)
  - $\nu(\vec{x}) = \vec{a}$

Proof: Sufficiency (cntd)

- Let $\nu' := \nu \circ \delta$. Then ν' is a valuation for Q' with the following properties.
- First, $\nu'(\vec{x}) = \nu(\delta(\vec{x})) = \nu(\vec{x}) = \vec{a}$.
- Moreover, if $R(\vec{t})$ is an atom of B', then
  - $\delta(R(\vec{t})) = R(\delta(\vec{t}))$ is an atom of B, since δ is a homomorphism, and
  - $\nu'(R(\vec{t})) = \nu(\delta(R(\vec{t}))) = \nu(R(\delta(\vec{t})))$ is an atom in $\mathbf{I}$, since ν satisfies B
- Hence, $\vec{a} \in Q'(\mathbf{I})$ and the other claims are shown as well.

Applying the Homomorphism Theorem: $Q_1 \equiv Q_2$

T1

A	B
y	x
x	z
x	y

T2

A	B
y	x
w	x
x	u
x	y

f(x)=x, f(y)=y
f(u)=z, f(w)=y

Hence Q1 $\subseteq$ Q2

T1

A	B
y	x
x	z
x	y

T2

A	B
y	x
w	x
x	u
x	y

f(x)=x, f(y)=y
f(z)=u

Hence Q2 $\subseteq$ Q1

Query Containment: Exercise

Find all containments and equivalences among the following conjunctive queries:

$$
\begin{aligned}
q_1(x,y) &\;\text{:–}\; r(x,y),\, r(y,z),\, r(z,w) \\
q_2(x,y) &\;\text{:–}\; r(x,y),\, r(y,z),\, r(z,u),\, r(u,w) \\
q_3(x,y) &\;\text{:–}\; r(x,y),\, r(z,u),\, r(v,w),\, r(x,z),\, r(y,u),\, r(u,w) \\
q_4(x,y) &\;\text{:–}\; r(x,y),\, r(y,3),\, r(3,z),\, r(z,w)
\end{aligned}
$$

Query Containment: Complexity

- Given two conjunctive queries, how hard is it to test whether $Q_1 \subseteq Q_2$?
- It is easy to transform them into tableaux, from either SPJ relational algebra queries, or SQL queries, or rule-based queries.
- However, a polynomial algorithm for deciding “equivalence” is unlikely to exist:

Theorem. Given two tableaux, deciding the existence of a homomorphism between them is NP-complete.

- In practice, query expressions are small, and thus conjunctive query optimization is nonetheless feasible in polynomial time

Minimizing Conjunctive Queries

Goal: Given a conjunctive query Q, find an equivalent conjunctive query Q' with the minimum number of joins.

Questions: How many such queries can exist? How different are they?

Assumption: We consider only CQs without equalities and inequalities.
We call these queries **simple conjunctive queries** (SCQs) .
If nothing else is said in this chapter, CQs are simple CQs

Terminology: If

$$Q(\vec{x}) \coloneq R_1(\vec{u}_1), \ldots, R_k(\vec{u}_k)$$

is a CQ, then Q' is a **subquery** of Q if Q' is of the form

$$Q'(\vec{x}) \coloneq R_{i_1}(\vec{u}_{i_1}), \ldots, R_{i_l}(\vec{u}_{i_l})$$

where $1 \leq i_1 < i_2 < \ldots < i_l \leq k$.

Minimization: Background Theory

Proposition 1. Let q be a SCQ with n atoms and q' be an equivalent SCQ with m atoms where $m < n$. Then there exists a subquery q_0 of q such that q_0 has at most m atoms in the body and q_0 is equivalent to q.

Proposition 2. Let q and q' be two equivalent minimal SCQs. Then q and q' are identical up to renaming of variables.

Conclusions:

- There is essentially one minimal version of each SCQ Q.
- We can obtain it by dropping atoms from Q's body.

An Algorithm for Minimizing SCQs

Given a conjunctive query Q, transform it into a tableau T.

Minimization algorithm:

$T' := T$;
repeat until no change
  choose a row a in T';
  if there is a homomorphism $\delta\colon T' \to T' \setminus \{a\}$
  then $T' := T' \setminus \{a\}$
end

Output: The query Q' corresponding to the tableau T'

Questions about the Algorithm

- Does it terminate?
- Is Q' equivalent to Q?
- Is Q' of minimal length among the queries equivalent to Q?

Minimizing SPJ/Conjunctive Queries: Example

- R with three attributes A, B, C
- SPJ query

$$Q = \pi_{AB}(\sigma_{B=4}(R)) \bowtie \pi_{BC}(\pi_{AB}(R) \bowtie \pi_{AC}(\sigma_{B=4}(R)))$$

- Translate into relational calculus (instead of normalizing):

$$\big(\exists z_1\, R(x,y,z_1) \wedge y = 4\big) \wedge \exists x_1 \Big(\big(\exists z_2\, R(x_1,y,z_2)\big) \wedge \big(\exists y_1\, R(x_1,y_1,z) \wedge y_1 = 4\big)\Big)$$

- Simplify, by substituting the constant, and putting quantifiers forward:

$$\exists x_1, z_1, z_2\, (R(x,4,z_1) \wedge R(x_1,4,z_2) \wedge R(x_1,4,z) \wedge y = 4)$$

- Conjunctive query:

$$Q(x,y,z) \text{ :– } R(x,4,z_1), R(x_1,4,z_2), R(x_1,4,z), y = 4$$

Minimizing SPJ/Conjunctive Queries (contd)

- Tableau T:

A	B	C
x	4	z_1
x_1	4	z_2
x_1	4	z
x	4	z

- Minimization, step 1: Is there a homomorphism from T to

A	B	C
x_1	4	z_2
x_1	4	z
x	4	z

?

- Answer: No. For any homomorphism δ, we have $\delta(x) = x$ (why?), thus the image of the first row is not in the smaller tableau.

Minimizing SPJ/Conjunctive Queries (contd)

- Step 2: Is T equivalent to

$$\begin{array}{ccc} \text{A} & \text{B} & \text{C} \\ \hline x & 4 & z_1 \\ x_1 & 4 & z \\ \hline x & 4 & z \end{array}$$

?

- Answer: Yes. Homomorphism δ: $\delta(z_2) = z$, all other variables stay the same.
- The new tableau is not equivalent to

$$\begin{array}{ccc} \text{A} & \text{B} & \text{C} \\ \hline x & 4 & z_1 \\ \hline x & 4 & z \end{array} \quad \text{or} \quad \begin{array}{ccc} \text{A} & \text{B} & \text{C} \\ \hline x_1 & 4 & z \\ \hline x & 4 & z \end{array}$$

- Because $\delta(x) = x$, $\delta(z) = z$, and the image of one of the rows is not present.

Minimizing SPJ/Conjunctive Queries (contd)

- Minimal tableau:

$$\begin{array}{ccc} \text{A} & \text{B} & \text{C} \\ \hline x & 4 & z_1 \\ x_1 & 4 & z \\ \hline x & 4 & z \end{array}$$

- Back to conjunctive query:

$$Q'(x, y, z) \;\text{:-}\; R(x, y, z_1), R(x_1, y, z), y = 4$$

- An SPJ query:

$$\sigma_{B=4}(\pi_{AB}(R) \bowtie \pi_{BC}(R))$$

- Pushing selections:

$$\pi_{AB}(\sigma_{B=4}(R)) \;\bowtie\; \pi_{BC}(\sigma_{B=4}(R))$$

Review of the Journey

- We started with

$$\pi_{AB}(\sigma_{B=4}(R)) \bowtie \pi_{BC}(\pi_{AB}(R) \bowtie \pi_{AC}(\sigma_{B=4}(R)))$$

- Translated into a conjunctive query
- Built a tableau and minimized it
- Translated back into conjunctive query and SPJ query
- Applied algebraic equivalences and obtained

$$\pi_{AB}(\sigma_{B=4}(R)) \;\bowtie\; \pi_{BC}(\sigma_{B=4}(R))$$

- Savings: one join.

Minimization of Conjunctive Queries: Multiple Relations

- We consider again the query:

$$Q(x, y)\text{:–}B(x, y), R(y, z), R(y, w), R(w, y)$$

- The tableau was:

B:

A	B
x	y

R:

A	B
y	z
y	w
w	y

Summary: x y

Minimization with Multiple Relations

- The algorithm is the same as before, but one has to try rows in different relations. Consider the homomorphism where $\delta(z) = w$, and δ is the identity for all other variables. Applying this to the tableau for Q yields

B:

A	B
x	y

R:

A	B
y	w
w	y

Summary: x y

- This can't be further reduced, as for any homomorphism δ, $\delta(x) = x$, $\delta(y) = y$.
- Thus Q is equivalent to

$$Q'(x, y) \ :\!-\ B(x, y), R(y, w), R(w, y)$$

- One join is eliminated.

Query Optimization and Functional Dependencies

- Additional equivalences can be inferred if integrity constraints are known
- We consider here *functional dependencies*
- Example: Let R have attributes A, B, C. Assume that R satisfies $A \rightarrow B$.
- Then it holds that

$$R \ = \ \pi_{AB}(R) \bowtie \pi_{AC}(R)$$

- Tableaux can help with these optimizations!
- $\pi_{AB}(R) \bowtie \pi_{AC}(R)$ as a conjunctive query:

$$Q(x, y, z) :\!- R(x, y, z_1), R(x, y_1, z)$$

Query Optimization and Functional Dependencies (contd)

- Tableau:

A	B	C
x	y	z_1
x	y_1	z
x	y	z

- Using the FD $A \rightarrow B$ infer $y = y_1$
- Next, minimize the resulting tableau:

A	B	C
x	y	z_1
x	y	z
x	y	z

$\longrightarrow$

A	B	C
x	y	z
x	y	z

- And this says that the query is equivalent to $Q'(x, y, z)$:–$R(x, y, z)$, that is, R
- This is known as the "chase" technique

Query Optimization and Functional Dependencies (contd)

- General idea: simplify the tableau using functional dependencies and then minimize.
- Given: a conjunctive query Q, and a set of FDs F
- Algorithm:

  Step 1. Construct the tableau T for Q

  Step 2. Apply algorithm CHASE(T, F)

  Step 3. Minimize output of CHASE(T, F)

  Step 4. Construct a query from the tableau produced in Step 3

The CHASE

We assume that all FDs are of the form $X \to A$, where A is a single attribute. For simplicity, we also assume that the tableau has only a single relation. The generalisation is straightforward.

for each $X \to A$ in F do
  for each t_1, t_2 in T such that $t_1[X] = t_2[X]$ and $t_1[A] \neq t_2[A]$ do
    case $t_1[A], t_2[A]$ of
      one nondistinguished variable $\Rightarrow$
        replace the nondistinguished variable by the other term
      one distinguished variable, one distinguished variable or constant $\Rightarrow$
        replace the distinguished variable by the other term
      two constants $\Rightarrow$
        output $\perp$ and STOP
  end
end

Query Optimization and Functional Dependencies: Example 2

- R is over A, B, C; $F = \{ B \to A \}$
- $Q = \pi_{BC}(\sigma_{A=4}(R)) \bowtie \pi_{AB}(R)$
- Q as a conjunctive query:

$$Q(x, y, z) \text{ :- } R(4, y, z), R(x, y, z_1)$$

- Tableau:

A	B	C
4	y	z
x	y	z_1
x	y	z

$\xrightarrow{\text{CHASE}}$

A	B	C
4	y	z
4	y	z_1
4	y	z

$\xrightarrow{\text{minimize}}$

A	B	C
4	y	z
4	y	z

- Final result: $Q(x, y, z) \text{ :- } R(x, y, z), x = 4$, that is, $\sigma_{A=4}(R)$.

Query Optimization and Functional Dependencies: Example 3

- Same R and F; the query is:

$$Q \;=\; \pi_{BC}(\sigma_{A=4}(R)) \bowtie \pi_{AB}(\sigma_{A=5}(R))$$

- As a conjunctive query:

$$Q(x,y,z) \;\text{:-}\; R(4,y,z), R(x,y,z_1), x=5$$

- Tableau:

$$\begin{array}{ccc} \text{A} & \text{B} & \text{C} \\ \hline 4 & y & z \\ 5 & y & z_1 \\ \hline 5 & y & z \end{array} \quad \stackrel{\text{CHASE}}{\longrightarrow} \quad \perp$$

- Final result: $\perp$ (the empty query)
- This equivalence does not hold *without* the FD $B \rightarrow A$

Query Optimization and Functional Dependencies: Example 4

- Sometimes simplifications are quite dramatic
- Same R, FD is $A \rightarrow B$, the query is

$$Q \;=\; \pi_{AB}(R) \bowtie \pi_A(\sigma_{B=4}(R)) \bowtie \pi_{AB}(\pi_{AC}(R) \bowtie \pi_{BC}(R))$$

- Convert into conjunctive query:

$$Q(x,y) \;\text{:-}\; R(x,y,z_1), R(x,y_1,z), R(x_1,y,z), R(x,4,z_2)$$

Query Optimization and Functional Dependencies: Example 4 (contd)

- Tableau:

$$\begin{array}{ccc} \text{A} & \text{B} & \text{C} \\ \hline x & y & z_1 \\ x & y_1 & z \\ x_1 & y & z \\ x & 4 & z_2 \\ \hline x & y & \end{array} \quad \overset{\text{CHASE}}{\longrightarrow} \quad \begin{array}{ccc} \text{A} & \text{B} & \text{C} \\ \hline x & 4 & z_1 \\ x & 4 & z \\ x_1 & 4 & z \\ x & 4 & z_2 \\ \hline x & 4 & \end{array} \quad \overset{\text{minimize}}{\longrightarrow} \quad \begin{array}{ccc} \text{A} & \text{B} & \text{C} \\ \hline x & 4 & z \\ \hline x & 4 & \end{array}$$

Query Optimization and Functional Dependencies: Example 4 (contd)

- $\begin{array}{ccc} \text{A} & \text{B} & \text{C} \\ \hline x & 4 & z \\ \hline x & 4 & \end{array}$ is translated into $Q(x,y) \coloneq R(x,y,z), y = 4$
- or, equivalently $\pi_{AB}(\sigma_{B=4}(R))$.
- Thus,

  $$\pi_{AB}(R) \bowtie \pi_A(\sigma_{B=4}(R)) \bowtie \pi_{AB}(\pi_{AC}(R) \bowtie \pi_{BC}(R)) \;=\; \pi_{AB}(\sigma_{B=4}(R))$$

  in the presence of FD $A \rightarrow B$.
- Savings: 3 joins!
- This cannot be derived by algebraic manipulations, nor conjunctive query minimization without using CHASE.

Questions about the CHASE

- Does the CHASE algorithm terminate? What is the run time?
- What is the relation between a tableau and its CHASE'd version?
- Query containment wrt a set of FD's:
  - How can we define this problem?
  - Can we decide this problem?
- Query minimsation wrt to a set of FDs
- Consider SCQs: we know from previous exercises that all such queries are satisfiable. Is the same true if we assume only database instances that satisfy a given set of FDs F?

Conjunctive Queries with Equalities and Inequalities

- Equality / Inequality atoms $x = y$, $x = a$, $x \neq z$, etc
- Let T, T' be the tableaux of the parts of conjunctive queries Q and Q' with ordinary relations
- **Sufficiency:** $Q \subseteq Q'$ if there exists a homomorphism $\delta\colon T' \to T$ such that for each (in)equality atom $t_1 \theta t_2$ in Q', we have that $\delta(t_1)\theta\delta(t_2)$ is logically implied by the equality and inequality atoms in Q
- However, existence of a homomorphims is no more a necessary condition for containment.
- It holds under certain conditions, though.

Note: Deciding whether a set of equality / inequality atoms A logically implies an equality / inequality atom is easy.

Query Homomorphism: Definition

Instead of tableau homomorphims, one often defines query homomorphisms.

An **homomorphism** from $q'(\vec{x}) \;\text{:–}\; R', C'$

to

$$q(\vec{x}) \;\text{:–}\; R, C$$

is a substitution δ such that

- $\delta(\vec{x}) = \vec{x}$
- $\delta(R') \subseteq R$
- $C \models \delta(C')$.

Here, $\delta(\vec{x})$, $\delta(R')$ and $\delta(C')$ are the extensions of δ to complex syntactic entities.

Note that we view R, R', C, and C' as sets of atoms.

Query Homomorphism: Example

$$\begin{aligned} Q'(x) \quad &\text{:–} \quad P(x,y),\ R(y,z), \\ &\qquad y \leq 3 \\ Q(x) \quad &\text{:–} \quad P(x,w),\ P(x,x),\ R(x,u), \\ &\qquad w \geq 5,\ x \leq 2 \end{aligned}$$

An homomorphism from Q' to Q is

$$\delta(x) = x,\ \ \delta(y) = x,\ \ \delta(z) = u.$$

Homomorphisms: The General Case

Existence of a homomorphism is not a necessary, but a sufficient condition for containment.

Theorem: Let Q, Q' be two conjunctive queries, possibly with comparisons and inequalities, such that Q and Q' have the same distinguished variables. Then

$Q \subseteq Q'$ if there exists a homomorphism from Q' to Q.

Containment of Queries With Comparisons

Classical example:

$$
\begin{aligned}
Q'() \quad &:- \quad P(u,v),\ u \leq v \\
Q() \quad &:- \quad P(y,z),\ P(z,y)
\end{aligned}
$$

- Q is *contained* in Q',
- but there is no homomorphism from Q' to Q.

Checking Containment of Queries with Comparisons: Idea

$$Q'() \quad :- \quad P(u,v),\ u \leq v$$
$$Q() \quad :- \quad P(y,z),\ P(z,y)$$

Replace Q with its **linear expansion** $(Q_L)_L$!

$$Q_{\{\,y<z\,\}} \quad :- \quad P(y,z),\ P(z,y),\ y<z$$
$$Q_{\{\,y=z\,\}} \quad :- \quad P(y,z),\ P(z,y),\ y=z$$
$$Q_{\{\,y>z\,\}} \quad :- \quad P(y,z),\ P(z,y),\ y>z$$

(case analysis)

Klug's Theorem

Theorem (Klug 88): If

- Q, Q' are **conjunctive** queries **with comparisons**
- $(Q_L)_L$ is the **linear expansion** of Q,

then:

$Q \subseteq Q' \quad \Leftrightarrow \quad$ for every Q_L in $(Q_L)_L$, there is an homomorphism from Q' to Q_L

(analogous for disjunctive queries)

Containment with comparisons is Π_2^P-complete.

(van der Meyden)

Readings

- S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

  Chapter 6 (Sections 6.1 and 6.2 for containment, equivalence and minimization of conjunctive queries) and Chapter 8 (Section 8.4 for the case with functional dependencies).

- If you want to study the same problems for the case of queries with inequalities, see the papers:
  - Anthony C. Klug. *On conjunctive queries containing inequalities.* Journal of the ACM 35(1): 146-160 (1988)
  - Ron van der Meyden. *The Complexity of Querying Indefinite Data about Linearly Ordered Domains.* Journal of Computer and System Sciences. 54(1): 113-135 (1997)