Database Theory

VU 181.140, SS 2018

6. Conjunctive Queries

Reinhard Pichler

Institut für Informationssysteme
Arbeitsbereich DBAI
Technische Universität Wien

24 April, 2018

Outline

Query Optimization

The common approach to (first-order) query optimization is via equivalence preserving transformations in relational algebra. E.g.:

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

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

- Cascaded selections might be merged:

$$\sigma_{c_1}(\sigma_{c_2}(E)) \quad = \quad \sigma_{c_1 \wedge c_2}(E)$$

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

$$\sigma_c(E_1 \bowtie E_2) \quad = \quad \sigma_c(E_1) \bowtie E_2$$

We do not treat such transformations in this course.

Beyond Standard Equivalences

- The known equivalences are not always sufficient:
  - e.g.: none of the equivalences reduces the number of joins!
- For further optimization, the following decision problems are crucial:

Definition (Query Equivalence and Containment)

We say a query Q_1 is equivalent to a query Q_2 (in symbols, $Q_1 \equiv Q_2$) if $Q_1(D) = Q_2(D)$ for every database instance D. Similarly, we say Q_1 is contained in Q_2 (written $Q_1 \subseteq Q_2$) if $Q_1(D) \subseteq Q_2(D)$ for every D.

QUERY-EQUIVALENCE

INSTANCE: A pair Q_1, Q_2 of queries.
QUESTION: Does $Q_1 \equiv Q_2$ hold?

QUERY-CONTAINMENT

INSTANCE: A pair Q_1, Q_2 of queries.
QUESTION: Does $Q_1 \subseteq Q_2$ hold?

- In the following we concentrate w.l.o.g. on query containment because

$$Q_1 \equiv Q_2 \Leftrightarrow Q_1 \subseteq Q_2 \text{ and } Q_2 \subseteq Q_1 \text{ and}$$
$$Q_1 \subseteq Q_2 \Leftrightarrow Q_1 \equiv (Q_1 \cap Q_2).$$

- Observe that if Q_1, Q_2 are formulated in relational algebra, then deciding $Q_1 \subseteq Q_2$ (and thus also $Q_1 \equiv Q_2$) is undecidable!
  - Indeed, Q is empty over all databases $\Leftrightarrow Q \subseteq \emptyset$.
  - By Traktenbrots Theorem, checking emptiness is undecidable for RA!
- Good news: $Q_1 \subseteq Q_2$ is decidable for conjunctive queries!
- The decidability comes from the Homomorphism Theorem (see below).
- The theorem also gives rise to optimization of conjunctive queries that reduces the number of joins.

Datalog-like notation for CQs

- Next we use Datalog notation for CQs!
- E.g.: the conjunctive query

$$\{\langle x, y\rangle \mid \exists z, w.B(x,y) \wedge R(y,z) \wedge R(y,w) \wedge R(w,y)\}$$

is written as the rule

$$Q(x,y){:}{-}B(x,y), R(y,z), R(y,w), R(w,y).$$

Conjunctive Queries into Tableaux

- Tableau: representation of a conjunctive query as a database
- A tableau for a CQ Q is just a database where variables can appear in tuples, plus a set of distinguished variables.
- Assume a query Q such that

$$Q(x,y){:}{-}B(x,y), R(y,z), R(y,w), R(w,y)$$

- Then the tableau of Q is:

B:

A	B
x	y

R:

A	B
y	z
y	w
w	y

x y ← answer line

- Variables in the answer line are called distinguished

Tableau homomorphisms

Definition (Tableau homomorphism)

A homomorphism of two tableaux $f\colon T_1 \to T_2$ is a mapping

$$f\colon \{\text{variables of } T_1\} \to \{\text{variables of } T_2\} \ \bigcup\ \{\text{constants}\}$$

such that:

- For every distinguished x, $f(x) = x$
- For every relation R in T_1 and row $(x_1, \dots, x_k)$ in R, tuple $(f(x_1), \dots, f(x_k))$ is a row of R in T_2

Theorem (Homomorphism Theorem)

Let Q_1, Q_2 be two conjunctive queries, and T_{Q_1}, T_{Q_2} their tableaux. Then

$$Q_1 \subseteq Q_2 \ \Leftrightarrow \text{ there exists a homomorphism } f\colon T_{Q_2} \to T_{Q_1}.$$

Applying the Homomorphism Theorem

- We first consider queries over a single relation:
- $Q_1(x, y)$:– $R(y, x), R(x, z)$
- $Q_2(x, y)$:– $R(y, x), R(w, x), R(x, u)$

Tableau for Q_1:

R:

A	B
y	x
x	z
x	**y**

Tableau for Q_2:

R:

A	B
y	x
w	x
x	u
x	**y**

T_{Q_1}

R:

A	B
y	x
x	z
x	**y**

T_{Q_2}

R:

A	B
y	x
w	x
x	u
x	**y**

Take f such that:

- $f(w) = y$,

T_{Q_1}

R:

A	B
y	x
x	z
x	**y**

T_{Q_2}

R:

A	B
y	x
w	x
x	u
x	**y**

Take f such that:

- $f(w) = y$,
- $f(u) = z$,

T_{Q_1}

R:

A	B
y	x
x	z
x	**y**

T_{Q_2}

R:

A	B
y	x
w	x
x	u
x	**y**

Take f such that:

- $f(w) = y$,
- $f(u) = z$,
- $f(x) = x$ and $f(y) = y$.

T_{Q_1}

R:

A	B
y	x
x	z
x	y

T_{Q_2}

R:

A	B
y	x
w	x
x	u
x	y

Take f such that:

- $f(w) = y$,
- $f(u) = z$,
- $f(x) = x$ and $f(y) = y$.
- Hence $Q_1 \subseteq Q_2$!

T_{Q_1}

R:

A	B
y	x
x	z
x	y

T_{Q_2}

R:

A	B
y	x
w	x
x	u
x	y

Take f such that:

- $f(z) = u$,

T_{Q_1}

R:

A	B
y	x
x	z
x	y

T_{Q_2}

R:

A	B
y	x
w	x
x	u
x	y

Take f such that:

- $f(z) = u$,
- $f(x) = x$ and $f(y) = y$.

T_{Q_1}

R:

A	B
y	x
x	z
x	y

T_{Q_2}

R:

A	B
y	x
w	x
x	u
x	y

Take f such that:

- $f(z) = u$,
- $f(x) = x$ and $f(y) = y$.
- Hence $Q_2 \subseteq Q_1$!

T_{Q_1}

R:	A	B
	y	x
	x	$\boxed{z}$
	x	y

T_{Q_2}

R:	A	B
	y	x
	w	x
	x	$\boxed{u}$
	x	y

Take f such that:

- $f(z) = u$,
- $f(x) = x$ and $f(y) = y$.
- Hence $Q_2 \subseteq Q_1$!
- Since $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$, we have $Q_2 \equiv Q_1$!

Proof of the Homomorphism Theorem.

Observation. A tuple $\vec{c}$ is in the answer to a CQ Q over a database D iff there is a homomorphism f from the tableau of Q to the database D such that $f(\vec{x}) = \vec{c}$, where $\vec{x}$ is the tuple of distinguished variables of Q.

Assume a pair Q_1, Q_2 of CQs with variables V_1, V_2, respectively. Assume that $\vec{x}$ is the tuple of answer variables of Q_1 and Q_2.

Suppose there exists a homomorphism $f\colon T_{Q_2} \to T_{Q_1}$. Assume a database D and an arbitrary tuple $\vec{c} \in Q_1(D)$. By the above observation there is a homomorphism g from T_{Q_1} to D such that $g(\vec{x}) = \vec{c}$. Observe that the composition $h(\cdot) = g(f(\cdot))$ is a homomorphism from T_{Q_2} to D such that $h(\vec{x}) = \vec{c}$. Hence $\vec{c} \in Q_2(D)$.

Suppose $Q_1 \subseteq Q_2$. Then, by assumption, $Q_1(D) \subseteq Q_2(D)$ for all instances D. Take the tableau T_{Q_1} as database instance D. Clearly, $\vec{x}$ is in the answer to Q_1 over T_{Q_1}. Then using the assumption we get $\vec{x} \in Q_2(T_{Q_1})$. By the observation above, then there is a homomorphism f from T_{Q_2} to T_{Q_1} such that $f(\vec{x}) = \vec{x}$. □

Existence of a Homomorphism: Complexity

Theorem

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

Proof.

NP-membership. Guess a candidate mapping f and check in polynomial time whether f is a homorphism.

NP-hardness. By a straightforward reduction from the NP-complete problem **BQE** for CQs. Let the Boolean CQ Q and database D be an arbitrary instance of **BQE**. We define the following tableaux T_1 and T_2:

T_1: tableau of the Boolean CQ Q.
T_2: consider D as tableau of a Boolean CQ

We clearly have: Query Q over DB D is non-empty $\Leftrightarrow$ there exists a homomorphism from T_1 to T_2. □

CQ Containment and Equivalence: Complexity

Corollary

Given two conjunctive queries Q_1 and Q_2, both deciding $Q_1 \subseteq Q_2$ and $Q_1 \equiv Q_2$ are NP*-complete.*

Proof.

The NP-completeness of CQ Containment follows immediately from the Homomorphism Theorem together with the above theorem.

From this, we may conclude the NP-completeness of CQ Equivalence via the following equivalences:

$$Q_1 \equiv Q_2 \Leftrightarrow Q_1 \subseteq Q_2 \text{ and } Q_2 \subseteq Q_1 \text{ and}$$
$$Q_1 \subseteq Q_2 \Leftrightarrow Q_1 \equiv (Q_1 \cap Q_2).$$ □

Minimizing Conjunctive Queries

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

More formally:

Definition

A conjunctive query Q is minimal if there does not exist a conjunctive query Q' such that

- $Q \equiv Q'$, and
- Q' has fewer atoms than Q.

Minimization by Deletion

- The following is an easy consequence of the Homomorphism Theorem:
  - Assume Q is
    $$Q(\vec{x}) \;\text{:-}\; R_1(\vec{u}_1), \ldots, R_k(\vec{u}_k)$$
  - Assume that there is an equivalent conjunctive query Q' of the form
    $$Q'(\vec{x}) \;\text{:-}\; S_1(\vec{v}_1), \ldots, S_l(\vec{v}_l), \qquad l < k.$$
  - Then Q is equivalent to a query of the form
    $$Q''(\vec{x}) \;\text{:-}\; R_{i_1}(\vec{u}_{i_1}), \ldots, R_{i_m}(\vec{u}_{i_m}), \text{ with } m \le l$$
- In other words, to minimize a conjunctive query, it suffices to consider deletions of atoms on the right of ":–". Why?

Minimization by Deletion (continued)

Proof idea

Consider CQs Q and Q' with $Q \equiv Q'$, s.t.
$Q(\vec{x}) \;\text{:-}\; R_1(\vec{u}_1), \ldots, R_k(\vec{u}_k)$ and
$Q'(\vec{x}) \;\text{:-}\; S_1(\vec{v}_1), \ldots, S_l(\vec{v}_l)$ and $l < k$.

By the Homomorphism Theorem, there exist homomorphisms $f\colon T_Q \to T_{Q'}$ and $g\colon T_{Q'} \to T_Q$.

Clearly, for the image of g, we have $|Im(g)| \le l$.

Let $Im(g) = \{R_{i_1}(\vec{u}_{i_1}), \ldots, R_{i_m}(\vec{u}_{i_m})\}$ with $m \le l$ and
let $Q''(\vec{x}) \;\text{:-}\; R_{i_1}(\vec{u}_{i_1}), \ldots, R_{i_m}(\vec{u}_{i_m})$.

We claim that then $Q'' \equiv Q$ holds.

Again, we apply the Homomorphism Theorem: We have to show that there exist homomorphisms $f''\colon T_Q \to T_{Q''}$ and $g''\colon T_{Q''} \to T_Q$.

Actually, g'' trivially exists – just take the identity.
Moreover, f'' can be obtained via composition: $f''(\cdot) = g(f(\cdot))$. $\square$

Minimization Procedure

- Given a conjunctive query Q, transform it into the tableau T_Q.
- Algorithm to obtain a minimal equivalent query:

  $T' := T_Q$;
  repeat until no change
      choose a row t in T';
      if there is a homomorphism $f\colon T' \to T' \setminus \{t\}$
      then $T' := T' \setminus \{t\}$
  end;
  return (the query defined by) T';

- Note: If a homomorphism $T' \to T' \setminus \{t\}$ exists, then T', $T' \setminus \{t\}$ define equivalent queries, as a homomorphism from $T' \setminus \{t\}$ to T' exists.

Minimizing Conjunctive Queries: example

- Conjunctive query with one relation R only:

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

- Tableau T_Q (relation R omitted):

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_Q to

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

- Answer: No. For any homomorphism f, $f(x) = x$ (why?), thus the image of the first row is not in the small tableau.

- Step 2: Is T_Q equivalent to

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

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

A	B	C
x	4	z_1
x	4	z

or

A	B	C
x_1	4	z
x	4	z

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

- Minimal tableau:

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

- Back to conjunctive query. CQ Q is equivalent to CQ Q' with

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

Complexity of Minimization (1)

Theorem

Given a tableau T and a tuple t in T, checking whether there is a homomorphism from T to $T \setminus \{t\}$ is NP*-complete.*

Proof.

Membership in NP is immediate. For the hardness part, we provide a reduction from 3-**COLORABILITY**. We exploit a well-known trick: a graph is 3-colorable iff it can be homomorphically embedded into a "triangle". Assume a graph $G = (V, E)$, where $V = \{1, \dots, n\}$. W.l.o.g., G is assumed to be connected. Take the Boolean CQ Q_G with the following atoms and test if atom $V_1(x_1)$ is "redundant":

1. $V_1(x_1), \dots, V_n(x_n)$,
2. $E(x_i, x_j)$ for each edge $(i, j) \in E$,
3. $R(y_r), G(y_g), B(y_b)$,
4. $E(y_r, y_g), E(y_g, y_r), E(y_g, y_b), E(y_b, y_g)$ and $E(y_r, y_b), E(y_b, y_r)$.
5. $V_i(y_c)$ for all $i \in V$ and $c \in \{r, g, b\}$.

Proof (continued).

It is not difficult to see that G is 3-colorable iff there is a homomorphishm from T_{Q_G} to $T_{Q_G} \setminus \{V_1(x_1)\}$.

($\Rightarrow$) Assume G is 3-colorable with $\mu\colon V \to \{r, g, b\}$ a witnessing coloring. Then the following function f is a homomorphism from T_{Q_G} to $T_{Q_G} \setminus \{V_1(x_1)\}$:

- $f(x_i) = y_{\mu(i)}$, for all $i \in V$,
- $f(y_c) = y_c$, for all $c \in \{r, g, b\}$.

($\Leftarrow$) Assume there is a homomorphishm f from T_{Q_G} to $T_{Q_G} \setminus \{V_1(x_1)\}$. Then $f(x_1) \in \{y_r, y_g, y_b\}$ due to the atom $V_1(x_1)$ of Q_G. Since G is connected, we must also have $f(x_i) \in \{y_r, y_g, y_b\}$ for all $i \in V$.

Take the function $\mu\colon V \to \{r, g, b\}$ such that (a) $\mu(i) = r$ if $f(x_i) = y_r$, (b) $\mu(i) = g$ if $f(x_i) = y_g$, and (c) $\mu(i) = b$ if $f(x_i) = y_b$.

We claim that μ is a valid 3-coloring of G. Let (i, j) be an arbitrary edge in E. Then $E(x_i, x_j)$ is an atom in Q_G. Since f is a homomorphism, we have $\langle f(x_i), f(x_j)\rangle$ in the relation E of $T_{Q_G} \setminus \{V_1(x_1)\}$. Then by construction of Q_G, we have $f(x_i) \neq f(x_j)$ and thus $\mu(i) \neq \mu(j)$. □

Complexity of Minimization (2)

Theorem

Given a conjunctive query Q, checking whether Q is minimal is co-NP*-complete.*

Proof.

We prove by showing that checking whether a query is not minimal is NP-complete. NP-Membership of the latter problem is immediate. For the hardness part, we observe that the query Q_G obtained from G in the previous proof can be reused. We show below that G is 3-colorable iff Q_G is not minimal.

($\Rightarrow$) Assume G is 3-colorable with $\mu\colon V \to \{r, g, b\}$ a witnessing coloring. Then the following function f (also used in the previous proof) is a homomorphism from T_{Q_G} to $T_{Q_G} \setminus \{V_1(x_1)\}$:

- $f(x_i) = y_{\mu(i)}$, for all $i \in V$,
- $f(y_c) = y_c$, for all $c \in \{r, g, b\}$.

Hence, Q_G is not minimal.

Proof (continued).

($\Leftarrow$) Assume Q_G is not minimal. Then there is $M \subset T_{Q_G}$ such that $M \neq \emptyset$ and there is a homomorphism f from T_{Q_G} to $T_{Q_G} \setminus M$. Let us analyze f. The domain of f is $\{y_r, y_g, y_b\} \cup \{x_1, \ldots, x_n\}$.

The atoms $R(y_r), G(y_g), B(y_b)$ in Q_G are the only atoms with leading symbol R, G, and B, respectively. Hence, none of the atoms $R(y_r), G(y_g), B(y_b)$ can be in M. Moreover, we must have $f(y_r) = y_r$, $f(y_g) = y_g$ and $f(y_b) = y_b$.

Since f is a homomorphism from T_{Q_G} to $T_{Q_G} \setminus M$, f cannot be the identity function and thus there exists $k \in V$ such that $f(x_k) \neq x_k$. Recall that for all $i \in V$ and all $V_i(t)$ of Q_G we have $t = x_i$, $t = y_r$, $t = y_g$ or $t = y_b$. Then we must have $f(x_k) \in \{y_r, y_g, y_b\}$.

Since G is connected, we must also have $f(x_i) \in \{y_r, y_g, y_b\}$ for all $i \in V$. Analogously to the proof of the theorem, we can define a valid 3-coloring of G as follows: $\mu\colon V \to \{r, g, b\}$ such that (a) $\mu(i) = r$ if $f(x_i) = y_r$, (b) $\mu(i) = g$ if $f(x_i) = y_g$, and (c) $\mu(i) = b$ if $f(x_i) = y_b$.

Uniqueness of Minimal Queries

A natural question: does the order in which we remove tuples from the tableaux matter? The answer is "no" by the following theorem.

Theorem

If Q_1, Q_2 are two minimal queries equivalent to a query Q, then the tableaux T_{Q_1} and T_{Q_2} are isomorphic.

Proof.

The proof proceeds in several steps.

Homomorphisms. By the equivalences $Q_1 \equiv Q \equiv Q_2$, there exists a homomorphism $f\colon T_{Q_1} \to T_{Q_2}$ and a homomorphism $g\colon T_{Q_2} \to T_{Q_1}$. Let $h = g \circ f$. Clearly, $h\colon T_{Q_1} \to T_{Q_1}$ is also a homomorphism.

$|T_{Q_1}| = |T_{Q_2}|$. Suppose that $|T_{Q_2}| < |T_{Q_1}|$ (the case $|T_{Q_1}| < |T_{Q_2}|$ is symmetric). Then $|h(T_{Q_1})| < |T_{Q_1}|$ and, hence, $h(T_{Q_1}) \subset T_{Q_1}$. Thus the query corresponding to $h(T_{Q_1})$ is strictly smaller than Q_1. This contradicts the assumption that Q_1 is a minimal CQ equivalent to Q.

Proof (continued).

h preserves the number of variables. Consider h as a mapping from the variables in T_{Q_1} to terms (i.e., variables and constants) in T_{Q_1}. We claim that $|Var(h(T_{Q_1}))| = |Var(T_{Q_1})|$. Suppose to the contrary that $Var(h(T_{Q_1})) < Var(T_{Q_1})$. Then $h(T_{Q_1}) \subset T_{Q_1}$ and again we get a contradiction since this would mean that the query corresponding to $h(T_{Q_1})$ is strictly smaller than Q_1.

h is a permutation of the variables in T_{Q_1}. $|Var(h(T_{Q_1}))| = |Var(T_{Q_1})|$ implies that h maps every variable in $Var(T_{Q_1})$ to a variable in $Var(T_{Q_1})$ (and not to a constant). Hence, h is a function $h\colon Var(T_{Q_1}) \to Var(T_{Q_1})$. Moreover, $|Var(h(T_{Q_1}))| = |Var(T_{Q_1})|$ also implies that h is bijective.

Isomorphism. Every multiple application of h (i.e., h, h^2, h^3, ...) again yields a permutation on $Var(T_{Q_1})$ and a homomorphism $T_{Q_1} \to T_{Q_1}$. For every permutation, there exists an $n \geq 1$ with $h^n = id$, i.e., $(g \circ f)^n = id$. Let $f^* = f \circ h^{n-1}$. Clearly, f^* is a homomorphism and $g \circ f^* = id$. In other words, $f^*\colon T_{Q_1} \to T_{Q_2}$ is bijective with inverse function g. Hence, f^* is an isomorphism. □

Acyclic Conjunctive Queries

- Many CQs in practice enjoy the so-called acyclicity property
- Acyclic CQs can be evaluated efficiently (in polynomial time)

Definition

A conjunctive query Q is acyclic if it is has a join tree.

- A join tree can be seen as (an efficiently executable) query plan

Definition (Join Tree)

Let $Q(\vec{x}) :- R_1(\vec{z}_1), \ldots, R_n(\vec{z}_n)$ be a CQ.

A join tree $T = (V, E)$ is a tree where

- $V = \{R_1(\vec{z}_1), \ldots, R_n(\vec{z}_n)\}$, i.e. V is the set of atoms in Q
- E satisfies for all variables z of Q:
  $\{R_j(\vec{z}_j) \in V \mid z \text{ occurs in } R_j(\vec{z}_j)\}$ induces a connected subtree in T

Join Tree – Example

Example

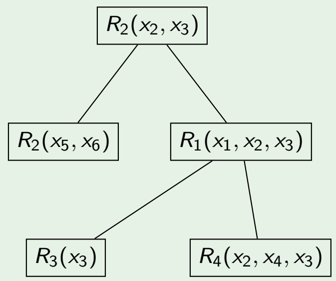
$Q(x_1, x_2, x_3, x_4, x_5, x_6) :-$
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$

Database Theory | 6. Conjunctive Queries | 6.4. Acyclic Conjunctive Queries

Join Tree – Example

Example

$Q(x_1, x_2, x_3, x_4, x_5, x_6) :-$
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$

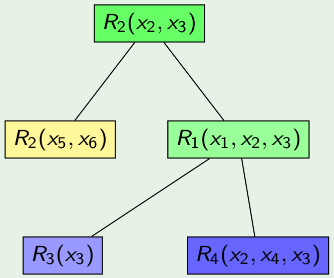


Pichler | 24 April, 2018 | Page 29

Join Tree – Example

Example

$Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$

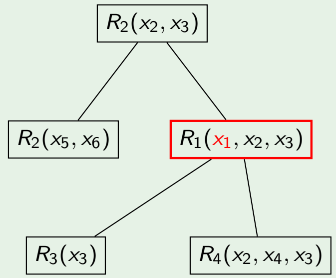


Database Theory | 6. Conjunctive Queries | 6.4. Acyclic Conjunctive Queries

Join Tree – Example

Example

$Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$



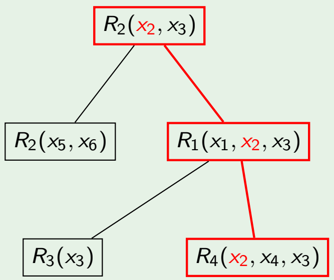
Pichler | 24 April, 2018 | Page 29

Database Theory | 6. Conjunctive Queries | 6.4. Acyclic Conjunctive Queries

Join Tree – Example

Example

$Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$



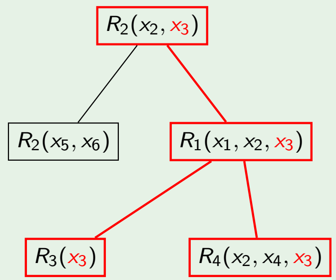
Pichler | 24 April, 2018 | Page 29

Database Theory | 6. Conjunctive Queries | 6.4. Acyclic Conjunctive Queries

Join Tree – Example

Example

$Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$

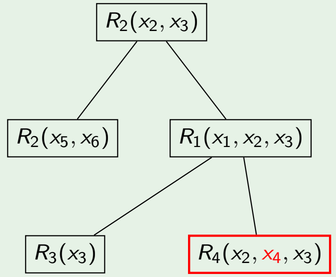


Pichler | 24 April, 2018 | Page 29

Join Tree – Example

Example

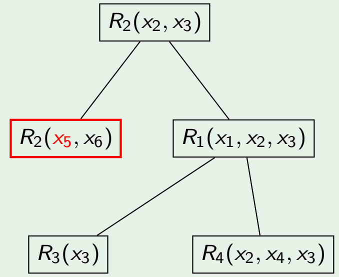
$Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \land R_4(x_2, x_4, x_3) \land R_1(x_1, x_2, x_3) \land R_2(x_2, x_3) \land R_2(x_5, x_6)$



Join Tree – Example

Example

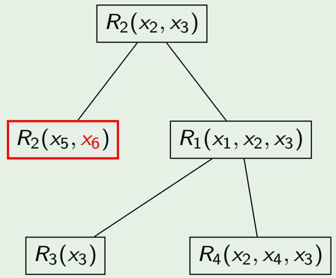
$Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \land R_4(x_2, x_4, x_3) \land R_1(x_1, x_2, x_3) \land R_2(x_2, x_3) \land R_2(x_5, x_6)$



Join Tree – Example

Example

$Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \land R_4(x_2, x_4, x_3) \land R_1(x_1, x_2, x_3) \land R_2(x_2, x_3) \land R_2(x_5, x_6)$



Finding Join Trees

Remarks:

- Existence of a join tree can be efficiently decided
- Join tree can be efficiently computed (if one exists)

$\rightarrow$ **GYO-reduction** (Graham, Yu, and Ozsoyoglu)

- Tests for acyclicity of hypergraphs
- Reduction sequence allows to build a join tree efficiently
- Easy to identify a query with a hypergraph
- Two equivalent definitions exist

Define

- Atom $R(\vec{z})$ is empty if $|\vec{z}| = 0$, and
- Atom $R_1(\vec{z}_1)$ is contained in atom $R_2(\vec{z}_2)$ if $\vec{z}_1 \subseteq \vec{z}_2$

GYO-Reduction

Definition (GYO/GYO'-reduction)

Let $Q(\vec{x})$:–$R_1(\vec{z}_1), \ldots, R_n(\vec{z}_n)$ be a CQ. Apply the following rules until no longer possible.

- GYO-reduction:
  - Eliminate variables that are contained in at most one atom.
  - Eliminate atoms that are empty or contained in another atom.
- GYO'-reduction:
  - Eliminate atoms that share no variables with other atoms.
  - Eliminate atoms R if there exists a witness R' s.t. each variable in R either appears in R only, or also appears in R'.

Theorem

- *GYO'(Q)* $= \emptyset$ *iff GYO(Q)* $= \emptyset$
- *GYO'(Q)* $= \emptyset$ *iff Q has a join tree (iff Q is acyclic)*

GYO-Reduction: Proof

Proof.

We only prove the second equivalence:

GYO'$(Q) = \emptyset \Rightarrow Q$ has a join tree: Consider the sequence $(R_1, \ldots, R_n)$ of atoms removed during the reduction. Create a join tree as follows:

- Whenever R_j was the witness for R_i, then make R_i a child node of R_j
- Merge the resulting forest to a tree "arbitrarily"

It is easy to check that this indeed gives a valid join tree.

Q has a join tree $\Rightarrow$ GYO'$(Q) = \emptyset$: Consider a join tree T for Q. Removing leaf nodes from T in arbitrary order gives a sequence of valid GYO'-reduction steps that eliminates all atoms:

- Either a leaf node shares no variable with its parent $\Rightarrow$ First rule
- All variables occuring not only in the leaf node must be contained in the parent node (connectedness condition) $\Rightarrow$ parent node is witness

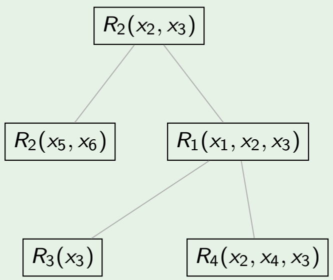
$\square$

GYO-reduction: Example

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–

$$\underbrace{R_3(x_3)}_{r_1} \wedge \underbrace{R_4(x_2, x_4, x_3)}_{r_2} \wedge \underbrace{R_1(x_1, x_2, x_3)}_{r_3} \wedge \underbrace{R_2(x_2, x_3)}_{r_4} \wedge \underbrace{R_2(x_5, x_6)}_{r_5}$$



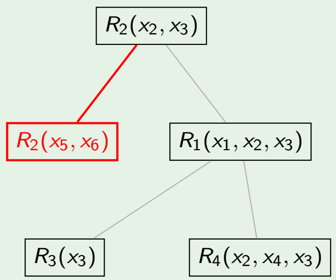
GYO-reduction: Example

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–

$$\underbrace{R_3(x_3)}_{r_1} \wedge \underbrace{R_4(x_2, x_4, x_3)}_{r_2} \wedge \underbrace{R_1(x_1, x_2, x_3)}_{r_3} \wedge \underbrace{R_2(x_2, x_3)}_{r_4} \wedge \underbrace{R_2(x_5, x_6)}_{r_5}$$

$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$

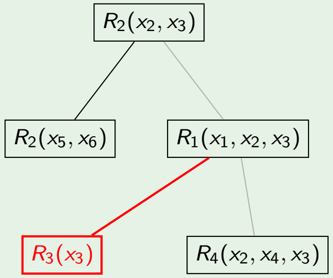


GYO-reduction: Example

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$
r_1 r_2 r_3 r_4 r_5

$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$
$\mathcal{A}_1 = \{r_1, r_2, r_3, r_4\}$



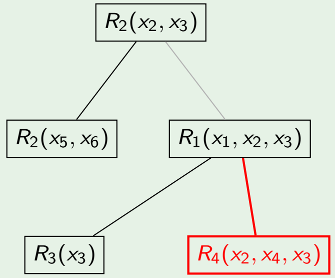
Database Theory 6. Conjunctive Queries 6.4. Acyclic Conjunctive Queries

GYO-reduction: Example

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$
r_1 r_2 r_3 r_4 r_5

$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$
$\mathcal{A}_1 = \{r_1, r_2, r_3, r_4\}$
$\mathcal{A}_2 = \{r_2, r_3, r_4\}$



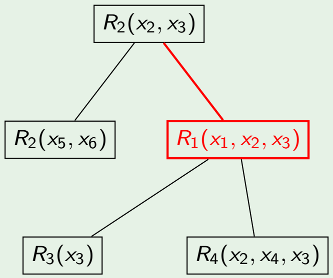
Database Theory 6. Conjunctive Queries 6.4. Acyclic Conjunctive Queries

GYO-reduction: Example

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$
r_1 r_2 r_3 r_4 r_5

$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$
$\mathcal{A}_1 = \{r_1, r_2, r_3, r_4\}$
$\mathcal{A}_2 = \{r_2, r_3, r_4\}$
$\mathcal{A}_3 = \{r_3, r_4\}$



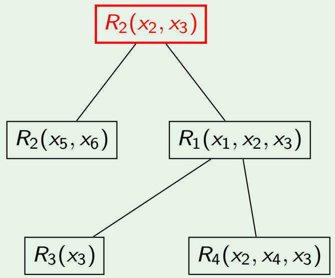
Database Theory 6. Conjunctive Queries 6.4. Acyclic Conjunctive Queries

GYO-reduction: Example

Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–
$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$
r_1 r_2 r_3 r_4 r_5

$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$
$\mathcal{A}_1 = \{r_1, r_2, r_3, r_4\}$
$\mathcal{A}_2 = \{r_2, r_3, r_4\}$
$\mathcal{A}_3 = \{r_3, r_4\}$
$\mathcal{A}_4 = \{r_4\}$



GYO-reduction: Example

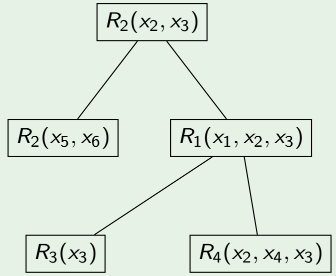
Example

Consider again $Q(x_1, x_2, x_3, x_4, x_5, x_6)$:–

$$R_3(x_3) \wedge R_4(x_2, x_4, x_3) \wedge R_1(x_1, x_2, x_3) \wedge R_2(x_2, x_3) \wedge R_2(x_5, x_6)$$

$r_1 \quad r_2 \quad r_3 \quad r_4 \quad r_5$

$\mathcal{A}_0 = \{r_1, r_2, r_3, r_4, r_5\}$
$\mathcal{A}_1 = \{r_1, r_2, r_3, r_4\}$
$\mathcal{A}_2 = \{r_2, r_3, r_4\}$
$\mathcal{A}_3 = \{r_3, r_4\}$
$\mathcal{A}_4 = \{r_4\}$
$\mathcal{A}_5 = \{\}$



Deciding ACQs Efficiently (Yannakakis)

Dynamic Programming Algorithm over the join tree $T = (V, E)$

Algorithm by Yannakakis

Let $T = (V, E)$ be a join tree of a query Q.
Given database instance D, decide $Q(D) = \emptyset$ as follows:

1. Assign to each $R_j(\vec{z}_j) \in V$ the corresponding relation R_j^D of D.
2. In a bottom up traversal of T: compute semijoins of R_j^D
3. If the resulting relation at root node is
   empty, then $Q(D) = \emptyset$,
   nonempty, then $Q(D) \neq \emptyset$.

Theorem

For ACQs Q:

- *Deciding $Q(D) = \emptyset$ is feasible in polynomial time.*
- *Computing $Q(D)$ can be done in output polynomial time.*

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1 : R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2 : R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3 : R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4 : R_3(x_3)$

x_3
b_1
b_2

$t_5 : R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1 : R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2 : R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3 : R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4 : R_3(x_3)$

x_3
b_1
b_2

$t_5 : R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1 : R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2 : R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3 : R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4 : R_3(x_3)$

x_3
b_1
b_2

$t_5 : R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1 : R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2 : R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3 : R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4 : R_3(x_3)$

x_3
b_1
b_2

$t_5 : R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
s_3	c_1	b_4
s_2	c_2	b_3

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
s_2	c_2	b_3

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
s_2	c_2	b_3

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
s_2	c_2	b_3

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1 : R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2 : R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3 : R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4 : R_3(x_3)$

x_3
b_1
b_2

$t_5 : R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1 : R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2 : R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3 : R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4 : R_3(x_3)$

x_3
b_1
b_2

$t_5 : R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1 : R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2 : R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3 : R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4 : R_3(x_3)$

x_3
b_1
b_2

$t_5 : R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1 : R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2 : R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3 : R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4 : R_3(x_3)$

x_3
b_1
b_2

$t_5 : R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Database Theory | 6. Conjunctive Queries | 6.4. Acyclic Conjunctive Queries

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Pichler | 24 April, 2018 | Page 35

Database Theory | 6. Conjunctive Queries | 6.4. Acyclic Conjunctive Queries

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Pichler | 24 April, 2018 | Page 35

Database Theory | 6. Conjunctive Queries | 6.4. Acyclic Conjunctive Queries

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Pichler | 24 April, 2018 | Page 35

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
s_3	c_3	b_1
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
c_4	b_6

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Yannakakis Algorithm – Enumeration

Two additional traversals allow us to enumerate all answers.

Theorem

Let Q be an acyclic conjunctive query. Given some database instance D, $Q(D)$ can be computed in output polynomial time, i.e., in time $O\left((||D|| + ||Q(D)||)^k\right)$ for some constant $k \geq 1$.

Enumeration Algorithm

Given a join tree of query Q; a database instance D. Compute $Q(D)$:

1. 1^{st} bottom-up traversal: semijoins as before (upwards propagation)
2. top-down traversal: "reverse" semijoins (downwards propagation)
3. 2^{nd} bottom-up traversal: compute solutions using joins.

Yannakakis Algorithm – Proof

Proof sketch.

Correctness of the algorithm follows from the following propositions: Given join tree T, for $t \in V(T)$ let T_t be the subtree of T rooted at t, R_t the relation computed by semijois and R'_t the one by joins:

$\square$

Database Theory | 6. Conjunctive Queries | 6.4. Acyclic Conjunctive Queries

Yannakakis Algorithm – Proof

Proof sketch.

Correctness of the algorithm follows from the following propositions: Given join tree T, for $t \in V(T)$ let T_t be the subtree of T rooted at t, R_t the relation computed by semijois and R'_t the one by joins:

1. After the 1^{st} bottom-up traversal:
$$R_t = \pi_{vars(t)}(\bowtie_{v \in V(T_t)} v) \text{ for each } t \in T$$

$\square$

Pichler | 24 April, 2018 | Page 37

Database Theory | 6. Conjunctive Queries | 6.4. Acyclic Conjunctive Queries

Yannakakis Algorithm – Proof

Proof sketch.

Correctness of the algorithm follows from the following propositions: Given join tree T, for $t \in V(T)$ let T_t be the subtree of T rooted at t, R_t the relation computed by semijois and R'_t the one by joins:

1. After the 1^{st} bottom-up traversal:
$$R_t = \pi_{vars(t)}(\bowtie_{v \in V(T_t)} v) \text{ for each } t \in T$$

2. After the top-down traversal:
$$R_t = \pi_{vars(t)}(\bowtie_{v \in V(T)} v) \text{ for each } t \in T$$

$\square$

Pichler | 24 April, 2018 | Page 37

Yannakakis Algorithm – Proof

Proof sketch.

Correctness of the algorithm follows from the following propositions:
Given join tree T, for $t \in V(T)$ let T_t be the subtree of T rooted at t, R_t the relation computed by semijois and R'_t the one by joins:

1. After the 1^{st} bottom-up traversal:
$$R_t = \pi_{vars(t)}(\bowtie_{v \in V(T_t)} v) \text{ for each } t \in T$$
2. After the top-down traversal:
$$R_t = \pi_{vars(t)}(\bowtie_{v \in V(T)} v) \text{ for each } t \in T$$
3. After the 2^{nd} bottom-up traversal:
$$R'_t = \pi_{vars(T_t)}(\bowtie_{v \in V(T)} v) \text{ for each } t \in T$$

□

Yannakakis Algorithm – Proof

Proof sketch.

Correctness of the algorithm follows from the following propositions:
Given join tree T, for $t \in V(T)$ let T_t be the subtree of T rooted at t, R_t the relation computed by semijois and R'_t the one by joins:

1. After the 1^{st} bottom-up traversal:
$$R_t = \pi_{vars(t)}(\bowtie_{v \in V(T_t)} v) \text{ for each } t \in T$$
2. After the top-down traversal:
$$R_t = \pi_{vars(t)}(\bowtie_{v \in V(T)} v) \text{ for each } t \in T$$
3. After the 2^{nd} bottom-up traversal:
$$R'_t = \pi_{vars(T_t)}(\bowtie_{v \in V(T)} v) \text{ for each } t \in T$$
$$\Rightarrow R'_r \text{ at root } r \text{ contains all results}$$

□

Enumeration – Example

Example

1. We have already performed the 1^{st} bottom-up traversal

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2
b_3

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Enumeration – Example

Example

1. We have already performed the 1^{st} bottom-up traversal
2. Top-down semijoins

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2
b_3

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Enumeration – Example

Example

1. We have already performed the 1^{st} bottom-up traversal
2. Top-down semijoins

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2
b_3

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Enumeration – Example

Example

1. We have already performed the 1^{st} bottom-up traversal
2. Top-down semijoins

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2
b_3

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Enumeration – Example

Example

1. We have already performed the 1^{st} bottom-up traversal
2. Top-down semijoins

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2
b_3

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Enumeration – Example

Example

1. We have already performed the 1^{st} bottom-up traversal
2. Top-down semijoins

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2
b_3

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Enumeration – Example

Example

1. We have already performed the 1^{st} bottom-up traversal
2. Top-down semijoins

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2
b_3

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Enumeration – Example

Example

1. We have already performed the 1^{st} bottom-up traversal
2. Top-down semijoins

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2
~~b_3~~

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Enumeration – Example

Example

1. We have already performed the 1^{st} bottom-up traversal
2. Top-down semijoins
3. Compute result in 2^{nd} bottom-up traversal

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

$t_4: R_3(x_3)$

x_3
b_1
b_2
~~b_3~~

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Enumeration – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
~~c_4~~	~~b_6~~

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
~~s_3~~	~~c_3~~	~~b_1~~
~~s_3~~	~~c_1~~	~~b_4~~
~~s_2~~	~~c_2~~	~~b_3~~

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2

$t_4: R_3(x_3)$

x_3
b_1
b_2
~~b_3~~

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

Enumeration – Example

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
$\cancel{c_4}$	$\cancel{b_6}$

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
$\cancel{s_3}$	$\cancel{c_3}$	$\cancel{b_1}$
$\cancel{s_3}$	$\cancel{c_1}$	$\cancel{b_4}$
$\cancel{s_2}$	$\cancel{c_2}$	$\cancel{b_3}$

$t_4: R_3(x_3)$

x_3
b_1
b_2
$\cancel{b_3}$

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

x_1	x_2	x_3	x_4
s_1	c_1	b_1	a_1
s_1	c_1	b_2	a_1
s_1	c_1	b_2	a_2

Enumeration – Example

x_2	x_3	x_5	x_6
c_1	b_2	c_1	b_2
c_1	b_2	c_1	b_1
c_1	b_2	c_4	b_6
c_1	b_1	c_1	b_2
c_1	b_1	c_1	b_1
c_1	b_1	c_4	b_6

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
$\cancel{c_4}$	$\cancel{b_6}$

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
$\cancel{s_3}$	$\cancel{c_3}$	$\cancel{b_1}$
$\cancel{s_3}$	$\cancel{c_1}$	$\cancel{b_4}$
$\cancel{s_2}$	$\cancel{c_2}$	$\cancel{b_3}$

$t_4: R_3(x_3)$

x_3
b_1
b_2
$\cancel{b_3}$

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

x_1	x_2	x_3	x_4
s_1	c_1	b_1	a_1
s_1	c_1	b_2	a_1
s_1	c_1	b_2	a_2

Enumeration – Example

x_1	x_2	x_3	x_4	x_5	x_6
s_1	c_1	b_2	a_1	c_1	b_2
s_1	c_1	b_2	a_1	c_1	b_1
s_1	c_1	b_2	a_1	c_4	b_6
s_1	c_1	b_2	a_2	c_1	b_2
s_1	c_1	b_2	a_2	c_1	b_1
s_1	c_1	b_2	a_2	c_4	b_6
s_1	c_1	b_1	a_1	c_1	b_2
s_1	c_1	b_1	a_1	c_1	b_1
s_1	c_1	b_1	a_1	c_4	b_6

$t_1: R_2(x_2, x_3)$

x_2	x_3
c_1	b_2
c_1	b_1
$\cancel{c_4}$	$\cancel{b_6}$

$t_2: R_2(x_5, x_6)$

x_5	x_6
c_1	b_2
c_1	b_1
c_4	b_6

$t_3: R_1(x_1, x_2, x_3)$

x_1	x_2	x_3
s_1	c_1	b_1
s_1	c_1	b_2
$\cancel{s_3}$	$\cancel{c_3}$	$\cancel{b_1}$
$\cancel{s_3}$	$\cancel{c_1}$	$\cancel{b_4}$
$\cancel{s_2}$	$\cancel{c_2}$	$\cancel{b_3}$

$t_4: R_3(x_3)$

x_3
b_1
b_2
$\cancel{b_3}$

$t_5: R_4(x_2, x_4, x_3)$

x_2	x_4	x_3
c_1	a_1	b_1
c_1	a_1	b_2
c_1	a_2	b_2

x_1	x_2	x_3	x_4
s_1	c_1	b_1	a_1
s_1	c_1	b_2	a_1
s_1	c_1	b_2	a_2

Learning Objectives

- The notions of query equivalence and containment,
- The Homomorphism Theorem,
- The complexity of query equivalence and containment,
- Minimization of conjunctive queries,
- Acyclic conjunctive queries,
- The Yannakakis algorithm.