Topics in Probabilistic and Statistical Databases

Lecture 5: Query Evaluation

Dan Suciu
University of Washington

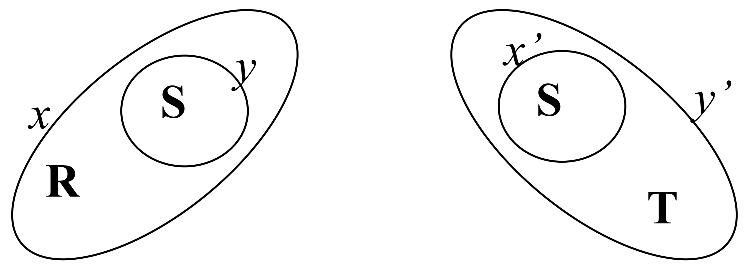
Inversion-Free Queries

Theorem
If q has no inversions then it is in PTIME

Stronger: there exists a polynomial size expression with + and * computing p(q)

Hierarchical Queries with Inversions

$H_0 = R(x),S(x,y),S(x',y'),T(y')$



There is an "inversion":
$sg(x) \supset sg(y)$, $sg(x') \subset sg(y')$ and $S(x,y)$ unifies with $S(x',y')$

Theorem H_0 is #P-hard

$q = R(x),S(x,y) \vee S(x',y'),T(y')$

Proof

Reduction from POSITIVE-PARTITIONED 2DNF

$$x_1 y_1 \vee x_2 y_1 \vee x_2 y_3 \vee \ldots$$

Let c_k = #satisfying assignments where exactly k clauses are false

The problem is to compute $c_0 + c_1 + \ldots + c_{m-1}$

R

x	P
1	0.5
2	0.5
3	0.5
. . .	

S

x	y	P
1	1	v
2	1	v
2	3	v
	. . .	

T

y	P
1	0.5
2	0.5
3	0.5
. . .	

OPEN: if **S** has probabilities 0.5

$$1 - p(q) = c_1\, 1/2^n\, (1-v) + c_2\, 1/2^n\, (1-v)^2 + \ldots + c_m\, 1/2^n\, (1-v)^m$$

Chose m different values for v; solve Vandermonde system

Longer Inversions

$H_k =$

$R(x), S_0(x,y),$

$S_0(u_1,v_1),S_1(u_1,v_1),$

$S_1(u_2,v_2),S_2(u_2,v_2),\ldots$

$S_{k-1}(u_k,v_k),S_k(u_k,v_k),$

$S_k(x',y'), T(y')$

Theorem : For each $k \geq 0$, H_k is #P-hard.

Proof: more involved, but same main idea

Unifications

Let g, g' be two subgoals.

Rename variables s.t. Vars(g) ∩ Vars(g') = Ø

g and g' *unify* if ∃ h, h' s.t. h(g)=h(g')

MGU = "most general unifier"

Don't unify

Unify

Don't unify

R(x), **S**(x,y,a), **S**(y,b,x),**S**(u,c,v)

Simple Fact

Proposition Let q, q' be two queries s.t. no two subgoals $g \in q$ and $g' \in q'$ unify. Then

$$p(q, q') = p(q)\, p(q')$$

Proof: q and q' are independent probabilistic events

q = R(x,y),S(y,a)   q' =T(u,v),S(v,b)

q, q' = R(x,y),S(y,a),T(u,v),S(v,b)

Inclusion/Exclusion Formula

$q = \exists x.\ f(x)$

Here f(x) is a query, and x is one of its variables

Proposition $p(q) = \sum_{T \neq \emptyset} (-1)^{|T|}\, p(f(T))$

Here f(T) means $f(a_1), f(a_2), \ldots, f(a_n)$, if $T = \{a_1, a_2, \ldots, a_n\}$

How does this generalize to $q = \exists x_1.\ f(x_1),\ \exists x_2.\ f(x_2), \ldots$

Example

Compute P(q), where:

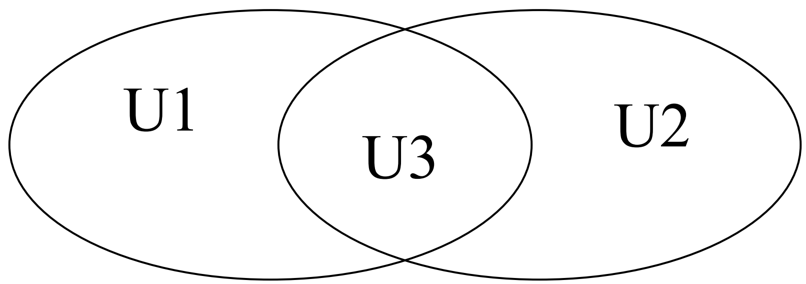
$$q = R(x), S(x), S(y), T(y) = f(x), g(y)$$

$$p(q) = \sum_{T1\neq\varnothing,\ T2\neq\varnothing} (-1)^{|T1|+|T2|}\, p(f(T_1)\ g(T_2))$$

We would like to commute p with f,g, but they are dependent…

Example

Idea: For each T1, T2, define U3=T1 ∩ T2, U1=T1-U3, U2=T1-U3



$$p(q)=\sum_{U1 \cup U3 \neq \emptyset,\ U2 \cup U3 \neq \emptyset,\ \mathrm{disjoint}(U1,U2,U3)} (-1)^{|U1|+|U2|} \; p(f(U_1)g(U_2)h(U_3))$$

Where f(x) = R(x),S(x), g(y) = S(y),T(y), h(z) = R(z), S(z), T(z)

Sums (1/2)

- We have ensured that all factors are independent
- Hence terms of the form:

  $$p(f_1(T_1)f_2(T_2)\ldots)$$

  become

  $$\prod_{i=1,k} \prod_{a \in T_I} g_i(a)$$

  where: $g_i(x) = p(f_i(x))$
- Now we examine how to compute sums of such terms, when $T_1, \ldots, T_k$ range over subsets of A, and are subject to predicates

Sums (2/2)

Exercise: compute $\sum_{T_1, T_2, T_3} g_1(T_1)\, g_2(T_2)\, g_3(T_3)$

Answer $\prod_{a \in A} (1 + g_1(a))(1 + g_2(a))(1 + g_3(a))$

$\sum_{T_1, T_2, T_3\,:\, T_i \cap T_j = \varnothing} g_1(T_1)\, g_2(T_2)\, g_3(T_3)$

$\prod_{a \in A} (1 + g_1(a) + g_2(a) + g_3(a))$

$\sum_{T_1 \cap T_2 = \varnothing,\, T_2 \cap T_3 = \varnothing,\, T_4 \subseteq T_2} g_1(T_1)\, g_2(T_2)\, g_3(T_3)\, g_4(T_4)$

$\prod_{a \in A} (1 + g_1(a) + g_2(a) + g_2(a)g_4(a) + g_3(a) + g_1(a)g_3(a))$

Theorem: for any FO predicate over $T_1, \ldots, T_k$, the sum $\sum_{T:\,\phi} g(T)$ admits a closed form linear in $|A|$

Challenge

- Sums are difficult
  - They are in PTIME, but they are so complex that we can't do on paper even the simplest examples

- Moreover: mismatch with relational algebra

What is a better abstraction to compute inversion free queries ?

Quiz

- Compute the following query (up to sum expressions):

  $q = R(x), S(x,y), S(x',y'), T(x')$

- Does this work for the following too ?

  $q = R(x), S(x,y), S(x',y'), T(y')$

Where we are (1/2)

- Query $q = f_1(x_1), \ldots, f_k(x_k)$
- Each xi is a root (in the sg-ordering) variable in fi (WHY DO WE NEED ?)
- Whenever a subgoal in fi unifies with one in fj, that unification results in xi=xj (WHERE DO WE NEED ?)

Where we are

- Then p(q) = big sum over (HOW MANY?) Ui's

$$p(q)=\sum_{U1,\ U2,\ \text{condition}} (-1)^{|\ldots|}\ p(f_1(U_1)\ldots)$$

Now $p(f_1(U_1)\ldots)$ is a probability of independent events;
hence: $= p(f_1(a_1))^* \ p(f_1(a_2))^* \ldots$
Need to compute a sum.

For each constant a_j, $f_i(a_j)$ is another query: recurs.

Coverage

$$q = R(x,b),R(a,y)$$

We have a problem, because x does not unify with y, but with a constant

We don't like that in the summation: it makes the transition from the Ti's to the Uj's too difficut.

Coverage

$$q = R(x,b),R(a,y)$$

Add predicates $x \neq a \lor x=a$ and also $x \neq a \lor x=a$

$$\begin{aligned} q = {} & R(a,b) \lor \\ & R(x,b),R(a,b),x\neq a \lor \\ & R(a,y),y\neq b \lor \\ & R(x,b), R(a,y), x\neq a,y\neq b \end{aligned}$$

$$\begin{aligned} f_0 &= R(a,b) \\ f_1(x) &= R(x,b),x\neq a \\ f_2(y) &= R(a,y),y\neq b \end{aligned}$$

$$q = f_1 f_2 \lor f_0 f_2 \lor f_0 f_1 \lor f_0$$

WHAT NEXT for p(q) ?

Coverage

- How is the root variable here ?

$$\boxed{q = R(x,y), R(y,x), S(x,y)}$$

Coverage

- Add predicates $x<y$, $x=y$, $x>y$
- Break ties using $<$

$q = R(x,y), R(y,x), S(x,y)$

$f_0(x) = R(x,x), S(x,x)$

Root var = x

$f_1(x,y) = R(x,y),R(y,x), S(x,y), x<y$

Root var = x

$f_2(x,y) = R(x,y),R(y,x), S(x,y), y<x$

Root var = y

Coverage

- The last thing we don't like:
  $q = R(x,x,y), R(u,v,v)$
- When we unify there is a "side-effect": $x=y$
- Easy to avoid: add predicates $x=y$, $x \neq y$ etc.

General Algorithm (1/2)

- Add predicates =,≠, or <, =, >
- Query is now "covered":
  - $q = c1 \vee c2 \vee \ldots$
  - Each ci = several "factors" (connected components)
- Each unifier:
  - maps variables only to variables (not constants)
  - is 1-to-1

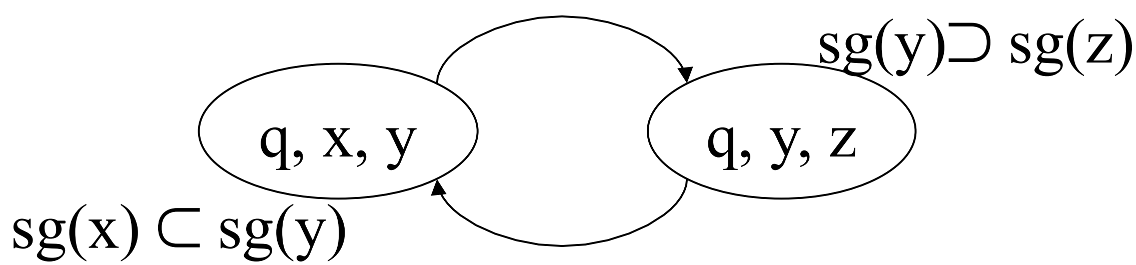
Inversions

Construct the graph:

- Nodes: (f, x, y) with $f \in F$, $x, y \in Vars(f)$
- Edges: (f, x, y) $\to$ (f', x', y') s.t. there exists an MGU mapping $x \to x'$ and $y \to y'$

Definition An *inversion* is a path from (f,x,y) to (f',x',y') s.t. $sg(x) \supset sg(y)$ and $sg(x') \subset sg(y')$

R(x,y), R(y,z)



General Algorithm (2/2)

- If there are no inversions, pick a unique root variable in each factor
- We have what we asked for: every unifier maps root variable to root variable

- Do summation…

$R(x)S_1(\underline{x}, y, \underline{y})$ $S_1(\underline{u}, v, \underline{w}), S_2(\underline{u}, v, \underline{w})$ $S_2(\underline{x'}, x', \underline{y'}), T(y')$	$qc_1 = R(x), S_1(\underline{x}, y, \underline{y}), x \neq y,$ $S_1(\underline{u}, v, \underline{v}), S_2(\underline{u}, v, \underline{v}), u \neq v$ $S_2(x', x', y'), T(y'), x' \neq y'$ $qc_2 = R(x), S_1(x, y, y), x \neq y$ $S_1(\underline{u}, u, \underline{w}), S_2(\underline{u}, u, \underline{w}), u \neq w$ $S_2(\underline{x'}, x', \underline{y'}), T(y'), x' \neq y'$	Illustrates the need for a strict coverage. The unification path forming an inversion in q in the trivial cover (which is non-strict) is interrupted when we add $\neq$ predicates to make the cover strict.

$R(x_1, x_2), S(\underline{x_1}, x_2, \underline{y}, y)$, $S(x_1, x_1, x_2, x_2)$ $S(\underline{x'}, x', \underline{y'}, y'), T(y')$	$\begin{aligned} qc &= R(x, x), S(\underline{x}, x, \underline{y}, y), \\ &\quad S(x, x, x, x), x \neq y \\ &\quad S(\underline{x'}, x', \underline{y'}, y'), T(y'), x' \neq y' \\ &= R(x, x), S(x, x, x, x), \\ &\quad S(x', x', y', y'), T(y'), x' \neq y' \end{aligned}$	This illustrates the need to minimize covers. The inversion disappears after minimizing *qc*.

$R(x_1,x_2), S(\underline{x_1}, x_2, \underline{y}, y)$ $S(x_1,x_2,x_1,x_2)$ $S(\underline{x'}, x', \underline{y'_1}, y'_2), T(y'_1, y'_2)$	$\begin{array}{rcl} qc_1 & = & R(x,x), S(\underline{x}, x, \underline{y}, y), x \neq y \\ & & S(\underline{x'}, x', \underline{y'}, y'), T(y',y'), x' \neq y' \\ & & S(x,x,x,x) \\ qc_2 & = & R(x,x), S(x,x,x,x), \\ & & S(x',x',y',y'), T(y',y'), x' \neq y' \end{array}$	This shows that we should not consider redundant coverages. There is an inversion in qc_1, but this cover is contained in qc_2 so it is redundant and after we remove qc_1 from the coverage there is no more inversion.