Conjunctive Queries

- **Definition:**
  \[ Q(X) :- R_1(X_1), R_2(X_2), ..., R_m(X_m) \]
- **Same as a single datalog rule**
- **Terminology:**
  - Atoms
  - Head variables
  - Existential variables

- **CQ =** denotes the set of conjunctive queries
Examples

- Example of CQ

\[ q(x,y) = \exists z. (R(x,z) \land \exists u. (R(z,u) \land R(u,y))) \]

\[ q(x) = \exists z. \exists u. (R(x,z) \land R(z,u) \land R(u,y)) \]

- Examples of non-CQ:

\[ q(x,y) = S(x,y) \land \forall z. (R(x,z) \rightarrow R(y,z)) \]

\[ q(x) = T(x) \lor \exists z. S(x,z) \]
Types of CQ

- **Full** CQ: head variables are all variables
  \[ Q(x,y,z,u) :- R(x,y),S(y,z),T(z,u) \]

- **Boolean** CQ: no head variables
  \[ Q() :- R(x,y),S(y,z),T(z,u) \]

- With or without self-joins:
  \[ Q(x,u) :- R(x,y),S(y,z),R(z,u) \]
  \[ Q(x,u) :- R(x,y),S(y,z),T(z,u) \]
Extensions

- With inequalities $CQ^\le$:
  \[ Q(x) \gets R(x,y), S(y,z), T(z,u), y < u \]

- With disequalities $CQ^\ne$:
  \[ Q(x) \gets R(x,y), S(y,z), T(z,u), y \ne u \]

- With aggregates:
  \[ Q(x, \text{count}(\ast)) \gets R(x,y), S(y,z), T(z,u) \]
  \[ Q(x, \text{sum}(u)) \gets R(x,y), S(y,z), T(z,u) \]
Complexity of Query Evaluation

• The query evaluation problem is this: given a query Q and a database D, compute Q(D)

• Three complexity measures:
  – **Data complexity.** Fix Q. The complexity is $f(|D|)$
    Variation: $f(|Input|, |Output|)$
  – **Query (or expression) complexity.** Fix D. The complexity is $f(|Q|)$
  – **Combined complexity.** The complexity if $f(|D|,|Q|)$

• Example: data complexity of $R \bowtie S$
Complexity of Query Evaluation

• The query evaluation problem is this: given a query Q and a database D, compute Q(D)

• Three complexity measures:
  – Data complexity. Fix Q. The complexity is \( f(|D|) \)
    Variation: \( f(|\text{Input}|, |\text{Output}|) \)
  – Query (or expression) complexity. Fix D. The complexity is \( f(|Q|) \)
  – Combined complexity. The complexity if \( f(|D|,|Q|) \)

• Example: data complexity of \( R \bowtie S \)
  – PTIME in \(|R|, |S|\) (trivially so...)
Complexity of Query Evaluation

• The query evaluation problem is this: given a query Q and a database D, compute Q(D)

• Three complexity measures:
  – **Data complexity.** Fix Q. The complexity is \( f(|D|) \)
    Variation: \( f(|Input|, |Output|) \)
  – **Query (or expression) complexity.** Fix D. The complexity is \( f(|Q|) \)
  – **Combined complexity.** The complexity if \( f(|D|,|Q|) \)

• Example: data complexity of \( R \bowtie S \)
  – PTIME in \(|R|, |S|\) (trivially so...)
  – Better: \( O(|R| * |S|) \)
Complexity of Query Evaluation

• The query evaluation problem is this: given a query Q and a database D, compute Q(D)

• Three complexity measures:
  – **Data complexity.** Fix Q. The complexity is f(|D|)
    Variation: f(|Input|, |Output|)
  – **Query (or expression) complexity.** Fix D. The complexity is f(|Q|)
  – **Combined complexity.** The complexity if f(|D|,|Q|)

• Example: data complexity of R \( \bowtie \) S
  – PTIME in |R|, |S| (trivially so...)
  – Better: O(|R| * |S|)
  – Even better: O( (|R|+|S|) log (|R|+|S|) + |Output| )
Complexity of Query Evaluation

• The query evaluation problem is this:
given a query Q and a database D, compute Q(D)

• Three complexity measures:
  – **Data complexity.** Fix Q. The complexity is f(|D|)
    Variation: f(|Input|, |Output|)
  – **Query (or expression) complexity.** Fix D. The complexity is f(|Q|)
  – **Combined complexity.** The complexity if f(|D|,|Q|)

• Example: data complexity of R \( \bowtie \) S
  – PTIME in |R|, |S| (trivially so...)
  – Better: \( O(|R| \times |S|) \)
  – Even better: \( O( (|R|+|S|) \log (|R|+|S|) + |Output| ) \)

• Discuss more about complexity in class...
Question in Class

- $Q(x,w) :- R(x,y), S(y,z), T(z,u), K(u,v), L(v,w)$

- Assume $|R| = |S| = |T| = |K| = |L| = N$

- What is the complexity of $Q$?
Question in Class

- $Q(x,w) : - R(x,y), S(y,z), T(z,u), K(u,v), L(v,w)$

- Assume $|R| = |S| = |T| = |K| = |L| = N$

- What is the complexity of $Q$?

- What is the complexity of this plan?
Question in Class

• Q(x,w) :- R(x,y), S(y,z), T(z,u), K(u,v), L(v,w)

• Assume |R| = |S| = |T| = |K| = |L| = N

• What is the complexity of Q?

• What is the complexity of this plan?

• Can you find a more efficient plan?
Question in Class

- Push projections down: What about this complexity?
- Can we still improve?

\[
\begin{align*}
\Pi_x &\quad \times \\
\Pi_u &\quad \times \\
\Pi_x &\quad \times \\
\Pi_z &\quad \times \\
R(x,y) &\quad S(y,z)
\end{align*}
\]
Law of Semijoins

- **Input**: $R(A_1, \ldots, A_n), \ S(B_1, \ldots, B_m)$
- **Output**: $T(A_1, \ldots, A_n)$

**Definition**: the semi-join operation is

$$R \bowtie S = \Pi_{A_1, \ldots, A_n} (R \bowtie S)$$

- **Data complexity**: $O(|R| + |S|)$ ignoring log-factors
- **The law of semijoins is**:

$$R \bowtie S = (R \bowtie S) \bowtie S$$
Laws with Semijoins

- Very important in parallel databases
- Often combined with Bloom Filters (my plan is to discuss them in the next lecture)
- Read pp. 747 in the textbook

- Also used in query optimization, sometimes called “magic sets” (see Chaudhuri’s paper)

- Historical note: magic sets were invented after semi-join reductions, and the connection became clear only later
Semijoin Reducer

- Given a query:

\[ Q = R_1 \Join R_2 \Join \ldots \Join R_n \]

- A **semijoin reducer** for Q is

\[
\begin{align*}
R_{i1} &= R_{i1} \Join R_{j1} \\
R_{i2} &= R_{i2} \Join R_{j2} \\
\vdots \\
R_{ip} &= R_{ip} \Join R_{jp}
\end{align*}
\]

such that the query is equivalent to:

\[ Q = R_{k1} \Join R_{k2} \Join \ldots \Join R_{kn} \]

- A **full reducer** is such that no dangling tuples remain
Example

- Example:
  \[ Q = R(A,B) \Join S(B,C) \]

- A semijoin reducer is:
  \[ R_1(A,B) = R(A,B) \Join S(B,C) \]

- The rewritten query is:
  \[ Q = R_1(A,B) \Join S(B,C) \]
Semijoin Reducer

• More complex example:
  \[ Q = R(A,B) \bowtie S(B,C) \bowtie T(C,D,E) \]

• What is a full reducer?
Semijoin Reducer

• More complex example:

\[ Q = R(A,B) \bowtie S(B,C) \bowtie T(C,D,E) \]

• A full reducer is:

\[
\begin{align*}
S'(B,C) &= S(B,C) \bowtie R(A,B) \\
T'(C,D,E) &= T(C,D,E) \bowtie S'(B,C) \\
S''(B,C) &= S'(B,C) \bowtie T'(C,D,E) \\
R'(A,B) &= R(A,B) \bowtie S''(B,C)
\end{align*}
\]

\[ Q = R'(A,B) \bowtie S''(B,C) \bowtie T'(C,D,E) \]
Practice at Home...

• Find semi-join reducer for
  \( \text{R}(x,y), \text{S}(y,z), \text{T}(z,u), \text{K}(u,v), \text{L}(v,w) \)
Not All Queries Have Full Reducers

• Example:

\[ Q = R(A,B) \bowtie S(B,C) \bowtie T(A,C) \]

• Can write many different semi-join reducers

• But no full reducer of length \( O(1) \) exists
Acyclic Queries

- Fix a Conjunctive Query without self-joins

- Q is *acyclic* if its atoms can be organized in a tree such that for every variable the set of nodes that contain that variable form a connected component.

R(x,y), S(y,z), T(z,x) is cyclic
Yannakakis Algorithm

- Given: acyclic query Q
- Compute Q on any database in time $O(|\text{Input}| + |\text{Output}|)$

- Step 1: semi-join reduction
  - Pick any root node $x$ in the tree decomposition of Q
  - Do a semi-join reduction sweep from the leaves to $x$
  - Do a semi-join reduction sweep from $x$ to the leaves

- Step 2: compute the joins bottom up, with early projections
Examples in Class

• Boolean query: Q() :- ...

• Non-boolean: Q(x,m) :- ...

• With aggregate: Q(x,sum(m)) :- ...

• And also: Q(x,count(*)) :- ...

In all cases: runtime = O(|R|+|S|+...+|L| + |Output|)
Testing if Q is Acyclic

• An *ear* of Q is an atom R(X) with the following property:
  – Let $X' \subseteq X$ be the set of join variables (meaning: they occur in at least one other atom)
  – There exists some other atom S(Y) such that $X' \subseteq Y$

• The GYO algorithm (Graham, Yu, Özsoyoğlu) for testing if Q is acyclic:
  – While Q has an ear R(X), remove the atom R(X) from the query
  – If all atoms were removed, then Q is acyclic
  – If atoms remain but there is no ear, then Q is cyclic

• Show example in class
Computing Cyclic Queries

Next lecture...