CMPS 277 – Principles of Database Systems

https://courses.soe.ucsc.edu/courses/cmps277/Fall11/01

Lecture #11
Limitations of Relational Algebra & Relational Calculus

Outline:

- Relational Algebra and Relational Calculus have substantial expressive power. In particular, they can express
  - Natural Join
  - Quotient
  - Unions of conjunctive queries
  - ...

- However, they cannot express recursive queries.

- Datalog is a declarative database query language that augments the language of conjunctive queries with a recursion mechanism.
Parents, Grandparents, and Greatgrandparents

- Let PARENT be a binary relational schema such that if \((a,b) \in \text{PARENT}\) in some database instance, then \(a\) is a parent of \(b\).

- Using PARENT, we can define GRANDPARENT and GREATGRANDPARENT by the conjunctive queries:

  \[
  \text{GRANDPARENT}(x,y) \Leftrightarrow \text{PARENT}(x,z), \text{PARENT}(z,y)
  \]

  and

  \[
  \text{GREATGRANDPARENT}(x,y) \Leftrightarrow \text{PARENT}(x,z), \text{PARENT}(z,w), \text{PARENT}(w,y)
  \]

- Similarly, we can define GREATGREATGRANDPARENT by a conjunctive query, and so on up to any fixed level of ancestry.
Parents and Ancestors

**Question:** Is there a relational algebra (relational calculus) expression that defines ANCESTOR from PARENT?

**Note:** This type of question occurs in other related concepts:
- Given a binary relation MANAGES(manager, employee), is there a relational algebra (relational calculus) expression that defines HIGHER-MANAGER?
- Given a binary relation DIRECT(from,to) about flights, is there a relational algebra (relational calculus) expression that defines CAN-FLY(from,to)?
- More abstractly, given a binary relation E, is there a relational algebra (relational calculus) expression that defines the **Transitive Closure** TC of E?
Edges and Paths

**Definition:** Let E be a binary relation
- For every $n \geq 1$, let $\text{PATH}_n$ be the binary query:
  “given a and b, is there a path of length $n$ from a to b along edges from E?”
- $\text{PATH}$ is the binary query:
  “given a and b, is there a path from a to b along edges from E?”

**Fact:**
- For every $n \geq 1$, the query $\text{PATH}_n$ is expressible by a conjunctive query. (Why?)
- Hence, $\text{PATH}$ is expressible by an infinite union of conjunctive queries:
  $$\text{PATH} \equiv \text{PATH}_1 \cup \text{PATH}_2 \cup \ldots \cup \text{PATH}_n \cup \ldots$$
Edges, Paths, and Transitive Closure

Facts: Let $E$ be a binary relation

- PATH is the Transitive Closure of $E$, i.e.,
  the smallest binary relation $T$ such that
  - $E \subseteq T$
  - $T$ is transitive (if $(a,b) \in T$ and $(b,c) \in T$, then $(a,c) \in T$).

- There are several well-known efficient algorithms for computing the Transitive Closure of a given binary relation $E$
  - Floyd–Warshall Algorithm (taught in CMPS 101).

- Recall that the following problem is NLOGSPACE-complete:
  Given $E$, $a$ and $b$, is there a path from $a$ to $b$? (i.e., is $(a,b) \in TC$?)
Transitive Closure and Relational Calculus

- **Question:** Is there a relational algebra (relational calculus) expression that defines ANCESTOR from PARENT? In other words, is there a relational algebra (relational calculus) expression that defines the Transitive Closure of a given binary relation E?

- **Theorem:** A. Aho and J. Ullman – 1979
  There is **no** relational algebra (or relational calculus) expression that defines the Transitive Closure of a given binary relation E.
Theorem: A. Aho and J. Ullman – 1979
There is no relational algebra (or relational calculus) expression that defines the Transitive Closure of a given binary relation E.

Note:
- The proof of this result requires methods from mathematical logic.
- Ehrenfeucht-Fraïssé Games is a powerful method for proving limitations in the expressive power of relational calculus.
  - Two-person perfect-information combinatorial games in which two players take turns and pick elements in two different database instances.
  - One player tries to maintain a partial isomorphism between the moves played; the other player tries to violate this.
Theorem: A. Aho and J. Ullman – 1979
There is no relational algebra (or relational calculus) expression that defines the Transitive Closure of a given binary relation E.

Intuition behind this result:
- Relational Calculus queries can only express “local” properties
Limitations of Relational Calculus and Relational Calculus

- There is **no** relational algebra (relational calculus) expression involving PARENT that defines ANCESTOR.

- ANCESTOR is definable by an **infinite union** of conjunctive queries, but is not definable by any **finite union** of conjunctive queries.

- Aho and Ullman’s Theorem reveals a **limitation** in the expressive power of relational algebra and relational calculus, namely
  - They **cannot** express recursive queries.
Overcoming the Limitations of Relational Calculus

- **Question:** What is to be done to overcome the limitations of the expressive power of relational calculus?

- **Answer 1:** Embedded Relational Calculus (Embedded SQL):
  - Allow SQL commands inside a conventional programming language, such as C, Java, etc.
  - This is an inferior solution, as it destroys the high-level character of SQL.

- **Answer 2:**
  - Augment relational calculus with a high-level declarative mechanism for recursion.
  - Conceptually, this a superior solution as it maintains the high-level declarative character of relational calculus.
Datalog

- Datalog = “Conjunctive Queries + Recursion”

- Datalog was introduced by Chandra and Harel in 1982 and has been studied by the research community in depth since that time:
  - Hundreds of research papers in major database conferences;
  - Numerous doctoral dissertations.
  - Recent applications outside databases:
    - Specification of network properties
    - Access control languages
    - Static program analysis (trace recursive calls)

- SQL:1999 and subsequent versions of the SQL standard provide support for a sublanguage of Datalog, called linear Datalog.
Datalog Syntax

- **Definition**: A **Datalog program** π is a finite set of rules each expressing a conjunctive query

\[ T(x_1, \ldots, x_k) :\!::=\! R_1(u_1), \ldots, R_n(u_n), \]

where each variable \( x_i \) occurs in the body of the rule.

- Some relational symbols occurring in the heads of the rules may also occur in the bodies of the rules (unlike the rules for conjunctive queries).
  - These relational symbols are the **recursive** relational symbols; they are also known as **intensional database predicates** (IDBs).

- The remaining relational symbols in the rules are known as the **extensional database predicates** (EDBs).
Datalog

- **Example:** Datalog program for Transitive Closure
  
  \[
  \begin{align*}
  T(x,y) & : \ E(x,y) \\
  T(x,y) & : \ E(x,z), T(z,y)
  \end{align*}
  \]
  
  - E is the EDB predicate
  - T is the IDB predicate
  - The intuition is that the Datalog program gives a recursive specification of the IDB predicate T in terms of the EDB E.

- **Example:** Another Datalog program for Transitive Closure
  
  \[
  \begin{align*}
  T(x,y) & : \ E(x,y) \\
  T(x,y) & : \ T(x,z), T(z,y)
  \end{align*}
  \]
  
  ("divide and conquer" algorithm for Transitive Closure)
Example: Paths of Even and Odd Length
Consider the Datalog program:

\[
\begin{align*}
\text{ODD}(x, y) & \quad :\quad \text{E}(x, y) \\
\text{ODD}(x, y) & \quad :\quad \text{E}(x, z), \ \text{EVEN}(z, y) \\
\text{EVEN}(x, y) & \quad :\quad \text{E}(x, z), \ \text{ODD}(z, y).
\end{align*}
\]

- E is the EDB predicate
- EVEN and ODD are the IDB predicates.
- So, a Datalog program may have several different IDB predicates (and it may have several different EDB predicates as well).
- This program gives a recursive specification of the IDB predicates EVEN and ODD in terms of the EDB predicate E.
- This is a Datalog program expressing mutual recursion.
 Conjunctive Queries vs. Datalog

- As we have seen, conjunctive queries can be written as rules:
  \[ T(x_1, \ldots, x_k) :\! : R_1(u_1), \ldots, R_n(u_n). \]
  - In such a rule, the relation symbol in the head does not occur in the body of the rule.

- Datalog programs are finite sets of rules.
  - In a Datalog program, however, a relation symbol occurring in the heads of a rule:
    - may also occur in the body of the same rule
      \[ T(x,y) :\! : E(x,z), T(z,y) \]
    - or, it may occur in the body of another rule in the program
      \[ ODD(x,y) :\! : E(x,z), EVEN(z,y) \]
      \[ EVEN(x,y) :\! : E(x,z), ODD(z,y). \]
Datalog Semantics

- **Question:** What is the precise semantics of a Datalog program?

- **Answer:** Datalog programs can be given two different types of semantics.
  - **Declarative Semantics (denotational semantics)**
    - Smallest solutions of recursive specifications.
    - Least fixed-points of monotone operators.
  
  - **Procedural Semantics (operational semantics)**
    - An iterative process for computing the “meaning” of Datalog programs.

- **Main Result:** The declarative semantics **coincides** with the procedural semantics.
Motivation:
- Recall the recursive definition of the **factorial function** $f(n) = n!$
  - $f(0) = 1$
  - $f(n+1) = (n+1) \cdot f(n)$
- These two equations give a **recursive specification** of the factorial function $f(n) = n!$
- The factorial function is the only function on the integers that satisfies this specification.
- Similarly, recall the recursive definition of $g(x,y) = x^y$
  - $g(x,0) = 1$
  - $g(x,y+1) = x \cdot g(x,y)$
- The **exponential function** $g(x,y) = x^y$ is the only function on the integers that satisfies this specification.
Each Datalog program can be viewed as a recursive specification of its IDB predicates. This specification is expressed using relational algebra operators:
- The body of each rule uses $\pi$, $\sigma$, and cartesian product $\times$
- All rules having the same predicate in the head are combined using union.
- The recursive specification is given by equations involving unions of conjunctive queries.

Example:  

\[
T(x,y) :\leftarrow E(x,y) \\
T(x,y) :\leftarrow T(x,z), T(z,y)
\]

Recursive equation:

\[
T = E \cup \pi_{1,4}(\sigma_{2=3}(T \times T))
\]
Example: Consider the Datalog program:

\begin{align*}
\text{ODD}(x,y) & :\ - \ E(x,y) \\
\text{ODD}(x,y) & :\ - \ E(x,z), \ \text{EVEN}(z,y) \\
\text{EVEN}(x,y) & :\ - \ E(x,z), \ \text{ODD}(z,y).
\end{align*}

- **System** of recursive equations:

\begin{align*}
\text{ODD} & = \ E \cup \pi_{1,4}(\sigma_{2=3}(E \times \text{EVEN})) \\
\text{EVEN} & = \pi_{1,4}(\sigma_{2=3}(E \times \text{ODD})).
\end{align*}
Unlike the recursive equations for the factorial and the exponential function, recursive equations arising from Datalog programs need not have a unique solution.

Example: Consider the recursive equation:

\[ T = E \cup \pi_{1,4}(\sigma_{2=3}(T \times T)) \]

Let \( E = \{ (1,2), (2,3) \} \).

Then both \( T_1 \) and \( T_2 \) satisfy this recursive equation, where

- \( T_1 = \{ (1,2), (2,3), (1,3) \} \)
- \( T_2 = \{ (1,2),(2,1),(2,3),(3,2),(1,3),(3,2),(1,1),(2,2),(3,3) \} \).

Furthermore, this recursive equation has many other solutions.
Theorem: Every recursive equation arising from a Datalog program with a single IDB has a smallest solution (smallest w.r.t. the $\subseteq$ partial order).

Example: Datalog program
- $T(x, y) :- E(x, y)$
- $T(x, y) :- T(x, z), T(z, y)$

- Recursive equation:
  - $T = E \cup \pi_{1,4}(\sigma_{2=3}(T \times T))$
- The smallest solution of this recursive equation is the Transitive Closure of E.

Note: This is a special case of the Knaster-Tarski Theorem for smallest solutions of recursive equations arising from monotone operators (it is important that Datalog uses monotone relational algebra operators only).
Declarative Semantics of Datalog Programs

- **Theorem:** Every recursive equation arising from a Datalog program with a single IDB has a smallest solution (smallest w.r.t. the $\subseteq$ partial order).

- **Definition:** The (declarative) semantics of a Datalog program is the smallest solution of the system of recursive equations arising from the Datalog program.

- **Note:**
  - If the Datalog program has more than one IDBs, then we get a system of recursive equations, instead of single one.

- **Question:**
  - What does “smallest solution of a system” mean in this case?
Example: Consider again the Datalog program:

\[
\begin{align*}
\text{ODD}(x,y) & : \text{E}(x,y) \\
\text{ODD}(x,y) & : \text{E}(x,z), \text{EVEN}(z,y) \\
\text{EVEN}(x,y) & : \text{E}(x,z), \text{ODD}(z,y).
\end{align*}
\]

- **System** of recursive equations:

\[
\begin{align*}
\text{ODD} & = \text{E} \cup \pi_{1,4}(\sigma_{2=3}(\text{E} \times \text{EVEN})) \\
\text{EVEN} & = \pi_{1,4}(\sigma_{2=3}(\text{E} \times \text{ODD})).
\end{align*}
\]

- The **smallest solution** of this system is a pair of relations (P,Q) such that

1. (P,Q) satisfies this system (e.g., Q = \(\pi_{1,4}(\sigma_{2=3}(\text{E} \times P))\)).
2. If a pair (P’,Q’) satisfies this system, then P \(\subseteq\) P’ and Q \(\subseteq\) Q’.
Declarative Semantics of Datalog Programs

Question:
- How difficult is it to compute the declarative semantics of Datalog programs?
- In other words, what is the computational complexity of the query evaluation problem for Datalog queries?

Note: On the face of their definition, the declarative semantics of Datalog programs do not give rise to an algorithm for computing this semantics.