1. Introduction: Relational Query Languages

Reinhard Pichler

Institut für Informationssysteme
Arbeitsbereich DBAI
Technische Universität Wien

1 March, 2016
Outline

1. Overview
   1.1 Databases and Query Languages
   1.2 Query Languages: Relational Algebra
   1.3 Query Languages: Relational (Domain) Calculus
   1.4 Query Languages: SQL
   1.5 Query Languages: other Languages
   1.6 Some Fundamental Aspects of Query Languages
A short history of databases

- 1970’s:
  - relational model of databases (E. F. Codd)
  - relational query languages (SQL)

- 1980’s:
  - relational query optimization
  - constraints, dependency theory
  - datalog (extend the query language with recursion)

- 1990’s:
  - new models: temporal databases, OO, OR databases
  - data mining, data warehousing

- 2000’s:
  - data integration, data exchange
  - data on the web, managing huge data volumes
  - new data formats: XML, RDF
  - data streams
Database theory

- Cut-crossing many areas in Computer Science and Mathematics
  - *Complexity* → efficiency of query evaluation, optimization
  - *Logics, Finite model theory* → expressiveness
  - *Logic programming, constraint satisfaction (AI)* → Datalog
  - *Graph theory* → (hyper)tree-decompositions
  - *Automata* → XML query model, data stream processing

- Benefit from other fields on the one hand, contribute new results on the other hand
Relational data model

- A **database** is a collection of relations (or tables)
- Each database has a **schema**, i.e., the **vocabulary (or signature)**
  - Each **relation** (table) has a name and a schema;
  - the schema of each relation $r$ is defined by a list of attributes (columns), denoted $\text{schema}(r)$
- Each attribute $A$ has a **domain** (or **universe**), denoted $\text{dom}(A)$
  - We define
    \[
    \text{dom}(r) = \bigcup_{A \in \text{schema}(r)} \text{dom}(A)
    \]
- Each relation contains a set of **tuples** (or **rows**)
  - Formally, a tuple in $r$ is a mapping $t : \text{schema}(r) \to \text{dom}(r)$ such that $t(A) \in \text{dom}(A)$ for all $A \in \text{schema}(r)$
Example

- **Schema**
  - *Author* (*AID* integer, *name* string, *age* integer)
  - *Paper* (*PID* string, *title* string, *year* integer)
  - *Write* (*AID* integer, *PID* string)

- **Instance**
  - \{⟨142, Knuth, 73⟩, ⟨123, Ullman, 67⟩, …\}
  - \{⟨181140pods, Querycontainment, 1998⟩, …\}
  - \{⟨123, 181140pods⟩, ⟨142, 193214algo⟩, …\}
Relational query languages

- Query languages are formal languages with syntax and semantics:
  - **Syntax**: algebraic or logical formalism or specific query language (like SQL). Uses the vocabulary of the DB schema.
  - **Semantics**: $M[Q]$ a mapping that transforms a database (instance) $D$ into a database (instance) $D' = M[Q](D)$, i.e. the database $M[Q](D)$ is the answer of $Q$ over the DB $D$.

- Usually, $M[Q]$ produces a single table, i.e., $M[Q] : D \rightarrow \text{dom}(D)^k$
  - in general: $k \geq 0$. We say “$Q$ is a $k$-ary query”.
  - **Boolean queries**: $k = 0$, i.e.:
    - possible values of $M[Q](D)$ are $\{\} (= \text{false})$ or $\{\langle \rangle \} (= \text{true})$.

- **Expressive power** of a query language: which mappings $M[Q]$ can be defined by queries $Q$ of a given query language?
Relational Algebra (RA)

- $\sigma \rightarrow Selection$
- $\pi \rightarrow Projection$
- $\times \rightarrow Cross\ product$
- $\bowtie \rightarrow Join$
- $\rho \rightarrow Rename$
- $- \rightarrow Difference$
- $\cup \rightarrow Union$
- $\cap \rightarrow Intersection$

*Primitive operations, all others can be obtained from these.

For precise definition of RA see any DB textbook or Wikipedia.
Example

- Recall the schema:
  
  - Author (AID integer, name string, age integer)
  - Paper (PID string, title string, year integer)
  - Write (AID integer, PID string)

- Example query: \( \text{PIDs of the papers NOT written by Knuth} \)
  
  \[
  \pi_{\text{PID}}(\text{Paper}) - \pi_{\text{PID}}(\text{Write} \bowtie \sigma_{\text{name} = \text{"Knuth"}}(\text{Author}))
  \]

- Example query: \( \text{AID}s of authors who wrote exactly one paper \)
  
  \[
  S_2 = \text{Write} \bowtie_{\text{AID} = \text{AID}' \land \text{PID} \neq \text{PID}'} \rho_{\text{AID}' \leftarrow \text{AID}, \text{PID}' \leftarrow \text{PID}}(\text{Write})
  \]
  
  \[
  S = \pi_{\text{AID}} \text{Write} - \pi_{\text{AID}} S_2
  \]
Recall First-order Logic (FO)

*Formulas* built using:

- Quantifiers: \( \forall, \exists, \)
- Boolean connectives: \( \land, \lor, \neg \)
- Parentheses: (, )
- Atoms: \( R(t_1, \ldots, t_n), t_1 = t_2 \)

Example database (i.e. a first-order structure):

- Schema: \( E \text{(FROM string, TO string)} \)
- Instance: \( \{ \langle v, u \rangle, \langle u, w \rangle, \langle w, v \rangle \} \)

Example sentences of FO:

- \( \forall x \exists y E(x, y) \)
- \( \exists x \forall y E(x, y) \)
- \( \forall x \exists y \exists z (E(z, x) \land E(x, y)) \)
- \( \forall x \exists y \exists z (\neg(y = z) \land E(x, y) \land E(x, z)) \)
Relational (Domain) Calculus

If \( \varphi \) is an FO formula with free variables \( \{x_1, \ldots, x_k\} \), then

\[
\{\langle x_1, \ldots, x_k \rangle \mid \varphi\}
\]

is a \( k \)-ary query of the domain calculus. On database \( A \) with domain \( A \), it returns the set of all tuples \( \langle a_1, \ldots, a_k \rangle \in (A)^k \) such that the sentence \( \varphi[a_1, \ldots, a_k] \) obtained from \( \varphi \) by replacing each \( x_i \) by \( a_i \) evaluates to true in the structure \( A \).

Notational simplifications.

- We often simply write \( \varphi \) rather than \( \{\langle x_1, \ldots, x_k \rangle \mid \varphi\} \) (i.e., the free variables of a formula are considered as the output).
- In particular, we usually write \( \varphi \) rather than \( \{\langle \rangle \mid \varphi\} \) for Boolean queries (\( k = 0 \)).
Example

- Recall the schema:
  - Author (AID integer, name string, age integer)
  - Paper (PID string, title string, year integer)
  - Write (AID integer, PID string)

- Example query: “PIDs of the papers NOT written by Knuth”

  \[ \{ PID \mid \exists T \exists Y ( \text{Paper}(PID, T, Y) \land \\
  \quad \land \neg (\exists A \exists AID (\text{Write}(AID, PID) \land \text{Author}(AID, "Knuth", A)))) \} \]

- Example query: “AIDs of authors who wrote exactly one paper”

  \[ \{ AID \mid \exists PID (\text{Write}(AID, PID) \land \neg \exists PID2 (\text{Write}(AID, PID2) \land PID \neq PID2)) \} \]
SQL (Structured Query Language)

- A standardized language:
  - most database management systems (DBMSs) implement SQL
- SQL is not only a query language:
  - supports constructs to manage the database (create/delete tables/rows)
- Query constructs of SQL (SELECT/FROM/WHERE/JOIN) are based on relational algebra

Example query: “AIDs of the co-authors of Knuth”

```
SELECT W1.AID
FROM Write W1, Write W2, Author
WHERE W1.PID=W2.PID AND W1.AID <> W2.AID
AND W2.AID=Author.AID AND Author.Name="Knuth"
```
Relational Algebra vs. Relational Calculus vs. SQL

Theorem (following Codd 1972)

Relational algebra, relational calculus, and SQL queries essentially have equal expressive power.

Restrictions apply: no “group by” and aggregation in SQL queries, “safety” requirements for relational calculus.

Languages with this expressive power are “relational complete”.

All 3 languages have their advantages:

1. use the flexible syntax of relational calculus to specify the query
2. use the simplicity of relational algebra for query simplification/optimization
3. use SQL to implement the query over a DB

More expressive query languages:

- many interesting queries cannot be formulated in these languages
- Example: no recursive queries (SQL now offers a recursive construct)
Towards other query languages

Paul Erdös (1913-1996), one of the most prolific writers of mathematical papers, wrote around 1500 mathematical articles in his lifetime, mostly co-authored. He had 509 direct collaborators.
Erdös number

- The *Erdös number*, is a way of describing the “collaborative distance”, in regard to mathematical papers, between an author and Erdös.

- An author’s *Erdös number* is defined inductively as follows:
  - *Paul Erdös* has an *Erdös number* of zero.
  - The *Erdös number* of author *M* is one plus the minimum among the *Erdös numbers* of all the authors with whom *M* co-authored a mathematical paper.

- Rothshild B.L. co-authored a paper with Erdös → Rothshild B.L.’s Erdös number is 1.
  - Kolaitis P.G. co-authored a paper with Rothshild B.L. → Kolaitis P.G.’s Erdös number is 2.
  - Pichler R. co-authored a paper with Kolaitis P.G. → Pichler R.’s Erdös number is 3.

- Rowling J.K.’s Erdös number is $\infty$
Queries about the Erdös number

- Recall the schema:
  - `Author (AID integer, name string, age integer)`
  - `Paper (PID string, title string, year integer)`
  - `Write (AID integer, PID string)`

- Assume that Erdös’s `AID` is 17

- Query “`AIDs` of the authors whose Erdös number ≤ 1”

  \[ P_1 = \pi_{PID}(\sigma_{AID=17} Write) \]

  \[ A_1 = \pi_{AID}(P_1 \bowtie Write) \]

- Query “`AIDs` of the authors whose Erdös number ≤ 2”

  \[ P_2 = \pi_{PID}(A_1 \bowtie Write) \]

  \[ A_2 = \pi_{AID}(P_2 \bowtie Write) \]
Queries about the Erdös number (continued)

What about $Q_1 = \text{AIDs of the authors whose Erdös number } < \infty$? 
What about $Q_2 = \text{AIDs of the authors whose Erdös number } = \infty$? 
Can we express $Q_1$ and $Q_2$ in relational calculus (or equivalently in RA)?
   - We cannot!
   - Formal methods to prove this negative result will be presented in the course

Are there query languages that allow us to express $Q_1$ and $Q_2$?
   - Yes, we can do this in DATALOG (the topic of the next lecture)
Some fundamental aspects of query languages

Questions dealt with in this lecture

- Expressive power of a query language
- Comparison of query languages
- Complexity of query evaluation
- Undecidability of important properties of queries (e.g., redundancy, safety)
- Important special cases (conjunctive queries)
- Inexpressibility results
Learning objectives

- Short recapitulation of
  - the notion of a relational database,
  - the notion of a query language and its semantics,
  - relational algebra,
  - first-order logic,
  - relational calculus,
  - SQL.

- Some fundamental aspects of query languages