1. Introduction: Relational Query Languages

Reinhard Pichler
Institut für Informationssysteme
Arbeitsbereich DBAI
Technische Universität Wien

1 March, 2016

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 (\(\text{AID integer, name string, age integer})\)
- Paper (\(\text{PID string, title string, year integer})\)
- Write (\(\text{AID integer, PID string})\)

**Instance**
- \{\(\langle 142, \text{Knuth}, 73 \rangle, \langle 123, \text{Ullman}, 67 \rangle, \ldots \} \)
- \{\(\langle 181140 \text{pods}, \text{Querycontainment}, 1998 \rangle, \ldots \} \)
- \{\(\langle 123, 181140 \text{pods} \rangle, \langle 142, 193214 \text{algo} \rangle, \ldots \} \)

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., \( M[Q] : D \to \text{dom}(D)^k \)
  - 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 \{\} (= false) or \{\} (= 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)

- \( \sigma \to \text{Selection}^* \)
- \( \pi \to \text{Projection}^* \)
- \( \times \to \text{Cross product}^* \)
- \( \bowtie \to \text{Join} \)
- \( \rho \to \text{Rename}^* \)
- \( = \to \text{Difference}^* \)
- \( \cup \to \text{Union}^* \)
- \( \cap \to \text{Intersection} \)

*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: *PIDs of the papers NOT written by Knuth*

\[ \pi_{\text{PID}}(\text{Paper}) - \pi_{\text{PID}}(\text{Write} \land \alpha_{\text{name} = "\text{Knuth}"}(\text{Author})) \]

Example query: *AIDs of authors who wrote exactly one paper*

\[ S_2 = \text{Write} \land \alpha_{\text{AID} = \text{AID}' \land \text{PID} \neq \text{PID}'} \pi_{\text{AID}, \text{PID}}(\text{Write}) \]

\[ S = \pi_{\text{AID}} \text{Write} - \pi_{\text{AID}} S_2 \]

Relational (Domain) Calculus

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

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

is a \( k \)-ary query of the domain calculus. On database \( A \) with domain \( A \), it returns the set of all tuples \( (a_1, \ldots, a_k) \in (A)^k \) 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 \( \{x_1, \ldots, x_k \mid \varphi \} \)
  (i.e., the free variables of a formula are considered as the output).
- In particular, we usually write \( \varphi \) rather than \( \{\} \mid \varphi \) for Boolean queries (\( k = 0 \)).

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:

- \( \forall x \exists y E(x, y) \)
- \( \exists \forall y E(x, y) \)
- \( \forall x \exists y \exists z (E(z, x) \land E(x, y)) \)
- \( \forall x \exists y \exists z (\neg(y = z) \land E(x, y) \land E(x, z)) \)
**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 JoINE) 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"
```

**Towards other query languages**

Paul Erdős (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.

**Erdős number**

- The Erdős number, is a way of describing the “collaborative distance”, in regard to mathematical papers, between an author and Erdős.
- An author’s Erdős number is defined inductively as follows:
  - Paul Erdős has an Erdős number of zero.
  - The Erdős number of author $M$ is one plus the minimum among the Erdős numbers of all the authors with whom $M$ co-authored a mathematical paper.
- Rothschild B.L. co-authored a paper with Erdős $\rightarrow$ Rothschild B.L.’s Erdős number is 1.
  - Kolaitis P.G. co-authored a paper with Rothschild B.L. $\rightarrow$ Kolaitis P.G.’s Erdős number is 2.
  - Pichler R. co-authored a paper with Kolaitis P.G. $\rightarrow$ Pichler R.’s Erdős number is 3.
- Rowling J.K.’s Erdős number is $\infty$
Queries about the Erdős 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 Erdős’s AID is 17
- Query “AIDs of the authors whose Erdős number ≤ 1”
  \[ P_1 = \pi_{\text{PID}}(\sigma_{\text{AID}=17} \text{Write}) \]
  \[ A_1 = \pi_{\text{AID}}(P_1 \bowtie \text{Write}) \]
- Query “AIDs of the authors whose Erdős number ≤ 2”
  \[ P_2 = \pi_{\text{PID}}(A_1 \bowtie \text{Write}) \]
  \[ A_2 = \pi_{\text{AID}}(P_2 \bowtie \text{Write}) \]

Queries about the Erdős number (continued)

- What about Q1 = “AIDs of the authors whose Erdős number < ∞”?
- What about Q2 = “AIDs of the authors whose Erdős number = ∞”?
- Can we express Q1 and Q2 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 Q1 and Q2?
  - 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