Database Theory

VU 181.140, SS 2011

1. Introduction: Relational Query Languages

Reinhard Pichler

Institut für Informationssysteme
Arbeitsbereich DBAI
Technische Universität Wien

8 March, 2011

Outline

A short history of databases

- 1970's: relational revolution
  - Relational model of databases (E. F. Codd), truly realizing Physical data independence
  - Relational query languages (SQL)
    - SEQUEL: SystemR from IBM
    - QUEL: Ingress from UC Berkeley
- 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
- Late 1990's until now: Internet revolution
  - Data integration on the web, managing huge data volumes
  - XML, Sensor networks, P2P

Database theory

- Cut-crossing many areas in Computer Science and Mathematics
  - *Complexity* $\longrightarrow$ efficiency of query evaluation, optimization
  - *Logics, Finite model theory* $\longrightarrow$ expressiveness
  - *Logic programming, constraint satisfaction (AI)* $\longrightarrow$ Datalog
  - *Graph theory* $\longrightarrow$ (hyper)tree-decompositions
  - *Automata* $\longrightarrow$ 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 (also called *structure*) is a collection of relations (or tables)
- Each database has a schema, i.e., the *vocabulary (or signature)*
  - Each relation r has a list of attributes (or columns) $\longrightarrow$ denoted $schema(r)$
- Each attribute A has a domain (or universe) denoted $dom(A)$
  - We define

$$dom(r) = \bigcup_{A \in schema(r)} dom(A)$$

- Each relation contains a set of tuples (or rows)
  - Formally, a tuple in r is a mapping $t : schema(r) \longrightarrow dom(r)$ such that $t(A) \in dom(A)$ for all $A \in schema(r)$
- Note: For ease of notation, we often use ordered lists of attributes instead of sets.

Example

- Schema
  - *Author* (*AID* integer, *name* string, *age* integer)
  - *Paper* (*PID* string, *title* string, *year* integer)
  - *Write* (*AID* integer, *PID* integer)
- Instance
  - $\{\langle 142, \mathit{Knuth}, 73\rangle, \langle 123, \mathit{Ullman}, 67\rangle, \ldots\}$
  - $\{\langle 181140\mathit{pods}, \mathit{Querycontainment}, 1998\rangle, \ldots\}$
  - $\{\langle 123, 181140\mathit{pods}\rangle, \langle 142, 193214\mathit{algo}\rangle, \ldots\}$

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)
- We always disregard queries that are dependent on the particular representation of domain values. We thus focus on generic queries

Definition

Generic queries are queries that produce isomorphic results on isomorphic databases.

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* integer)
- Example query: *PID*s of the papers NOT written by *Knuth*

$$\pi_{PID}(Paper) - \pi_{PID}(Write \bowtie \sigma_{name=''Knuth''}(Author))$$

- Example query: *AID*s of authors who wrote exactly one paper

$$S_2 = Write \bowtie_{AID=AID' \wedge PID \neq PID'} \rho_{AID' \leftarrow AID, PID' \leftarrow PID}(Write)$$

$$S = \pi_{AID} Write - \pi_{AID} S_2$$

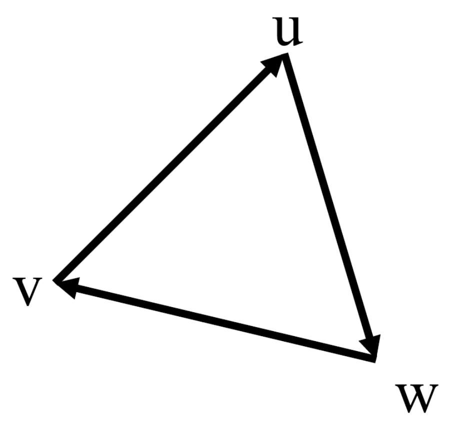
Recall First-order Logic (FO)

Formulas built using:

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

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

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



Example sentences of FO:

- $\forall x \exists y E(x, y)$
- $\forall x \exists y \exists z (E(z, x) \wedge E(x, y))$
- $\exists x \forall y \exists z (E(z, x) \wedge E(x, y))$
- $\forall x \exists y \exists z (\neg(y = z) \wedge E(x, y) \wedge E(x, z))$

Free variables of a formula

FO formulas may have free variables (i.e., not bound by a quantifier).

$$
\begin{aligned}
\mathit{free} \quad &: \quad \text{Formulae} \rightarrow 2^{\text{Variables}} \\
\mathit{free}(R(t_1, \ldots, t_n)) \quad &:= \quad \{t_i \mid t_i \text{ is a variable}, 1 \leq i \leq n\} \\
\mathit{free}(t_1 = t_2) \quad &:= \quad \{t_i \mid t_i \text{ is a variable}, 1 \leq i \leq 2\} \\
\mathit{free}(\varphi \wedge \psi) \quad &:= \quad \mathit{free}(\varphi) \cup \mathit{free}(\psi) \\
\mathit{free}(\varphi \vee \psi) \quad &:= \quad \mathit{free}(\varphi) \cup \mathit{free}(\psi) \\
\mathit{free}(\neg\varphi) \quad &:= \quad \mathit{free}(\varphi) \\
\mathit{free}(\exists x\ \varphi) \quad &:= \quad \mathit{free}(\varphi) - \{x\} \\
\mathit{free}(\forall x\ \varphi) \quad &:= \quad \mathit{free}(\varphi) - \{x\}
\end{aligned}
$$

Example

$\mathit{free}(\exists z R(x, y, z)) = \{x, y\}$

$\mathit{free}\big(\exists x_1 \exists x_2\ R(x_1, x_2) \wedge S(x_2, x_3)\big) = \{x_3\}$

Note: if $\mathit{free}(\varphi) = \emptyset$, then φ is a sentence.

Relational (Domain) Calculus

If φ is an FO formula with free(φ) $= \{x_1, \ldots, x_n\}$, then

$$\{\langle x_1, \ldots, x_n \rangle \mid \varphi\}$$

is an n-ary *query* of the domain calculus. On database $\mathcal{A}$ with domain A, it returns the set of all tuples $\langle a_1, \ldots, a_n \rangle \in (A)^n$ such that the sentence $\varphi[a_1, \ldots, a_n]$ obtained from φ by replacing each x_i by a_i evaluates to true in the structure $\mathcal{A}$.

- **All** free variables of φ must occur in the output tuple $\langle x_1, \ldots, x_n \rangle$.
- Slight syntactic generalization: Variables may be repeated in the output tuple of the query.
  Example: We may write $\{\langle x_1, \ldots, x_n, x_1 \rangle \mid \varphi\}$ as a shortcut for $\{\langle x_1, \ldots, x_n, x_1' \rangle \mid \varphi \wedge x_1 = x_1'\}$.
- We often simply write φ rather than $\{\langle x_1, \ldots, x_n \rangle \mid \varphi\}$ (i.e., the free variables of a formula are considered as the output).
- In particular, we usually write φ rather than $\{\langle\rangle \mid \varphi\}$ for Boolean queries ($n = 0$).

Example

- Recall the schema:
  - *Author* (*AID* integer, *name* string, *age* integer)
  - *Paper* (*PID* string, *title* string, *year* integer)
  - *Write* (*AID* integer, *PID* integer)
- Example query: "*PID*s of the papers NOT written by *Knuth*"

$$\{PID \mid \exists T \exists Y (Paper(PID, T, Y) \wedge$$

$$\wedge \neg (\exists A \exists AID (Write(AID, PID) \wedge Author(AID, \text{"}Knuth\text{"}, A))))\}$$

- Example query: "*AID*s of authors who wrote exactly one paper"

$$\{AID \mid \exists PID (Write(AID, PID) \wedge \neg \exists PID2 (Write(AID, PID2) \wedge PID \neq PID2))\}$$

Quantifier rank of a formula

We will need this for the future:

$$
\begin{array}{rcl}
qr & : & \text{Formulae} \rightarrow \mathbb{N} \\
qr(R(t_1, \ldots, t_n)) & := & 0 \\
qr(t_1 = t_2)) & := & 0 \\
qr(\varphi \wedge \psi) & := & \max(qr(\varphi), qr(\psi)) \\
qr(\varphi \vee \psi) & := & \max(qr(\varphi), qr(\psi)) \\
qr(\neg\varphi) & := & qr(\varphi) \\
qr(\exists x\ \varphi) & := & qr(\varphi) + 1 \\
qr(\forall x\ \varphi) & := & qr(\varphi) + 1
\end{array}
$$

Example

$$qr\big(\exists x_1 \exists x_2\ (x_1 = x_2 \wedge \neg\exists x_3\ R(x_1, x_2, x_3))\big) = 3.$$

$$qr\big(\exists x_1\ (\exists x_2\ x_1 = x_2) \wedge \neg(\exists x_3\ S(x_1, x_3))\big) = 2.$$

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
WHERE W1.PID=W2.PID AND W2.AID="Knuth"
```

Relational Algebra vs. Relational Calculus vs. SQL

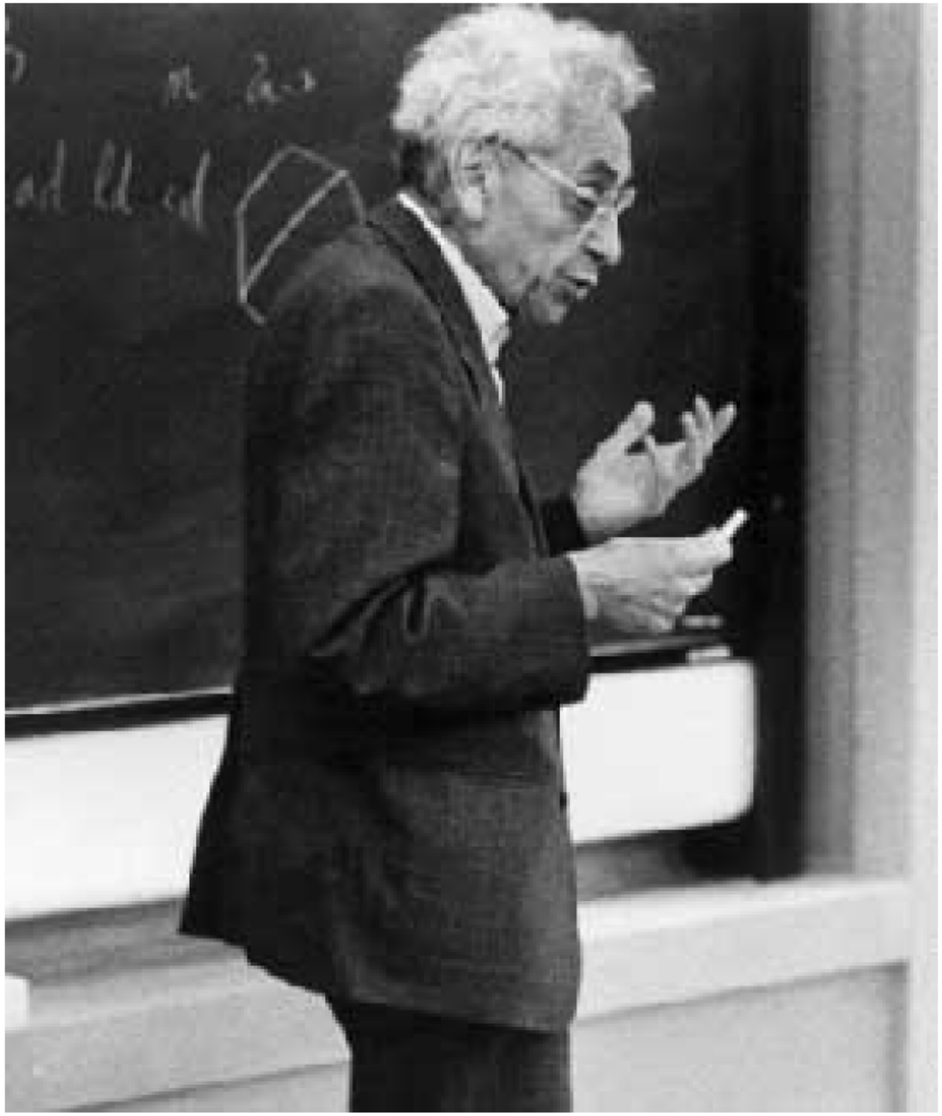
Theorem (following Codd 1972)

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

- queries in the 3 languages can be translated from one language to another while preserving the query answer
- 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

Restrictions apply: no aggregation in SQL queries, "safety" requirements for relational calculus.

Towards other query languages 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.
- Rothschild B.L. co-authored a paper with Erdös $\rightarrow$ Rothschild B.L.’s Erdös number is 1.
  - Kolaitis P.G. co-authored a paper with Rothschild B.L. $\rightarrow$ Kolaitis P.G.’s Erdös number is 2.
  - Gottlob G.,co-authored a paper with Kolaitis P.G. $\rightarrow$ Gottlob G.’s Erdös number is 3.
- Rowling J.K.’s Erdös number is ∞

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* integer)
- Assume that Erdös's *AID* is 001
- Query "*AIDs* of the authors whose Erdös number ≤ 1"

$$P_1 = \pi_{PID}(\sigma_{AID=001} Write)$$

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

- 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 Q1="*AIDs* of the authors whose Erdös number $\leq \infty$"?
- What about Q2="*AIDs* of the authors whose Erdös number $= \infty$"?
- 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 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 (free variables, quantifier rank),
  - relationcal calculus,
  - SQL.
- Some fundamental aspects of query languages