Schema Mappings and Data Exchange

Phokion G. Kolaitis
University of California, Santa Cruz
&
IBM Research-Almaden

EASSLC 2012
Southwest University
August 2012

Logic and Databases

- Extensive interaction between logic and databases during the past 40 years.
- Logic provides both a unifying framework and a set of tools for formalizing and studying data management tasks.
- The interaction between logic and databases is a prime example of
  - Logic **in** Computer Science

    but also
  - Logic **from** Computer Science

Logic and Databases

In the first half of the course, we will learn about:

Database Query languages: Expressive Power and Complexity

- Relational Algebra and Relational Calculus
- Conjunctive queries and homomorphisms

Note: Logic as a query language

In the second half of the course, we will learn about a different use of logic in databases:

Integrity Constraints in Databases (aka Database Dependencies)

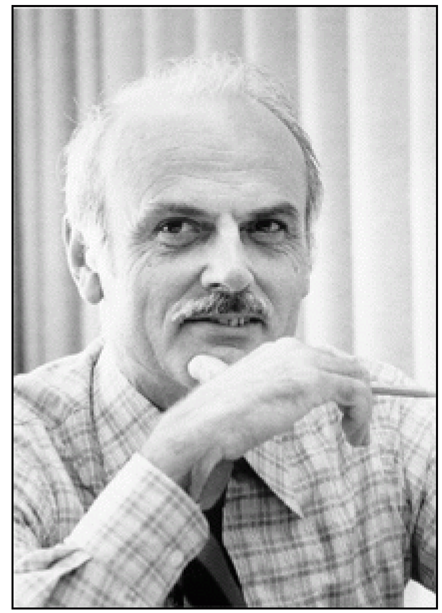
- Tuple-generating dependencies
- Equality Generating dependencies

Note: Logic as a constraint language

Relational Databases: How it all got started

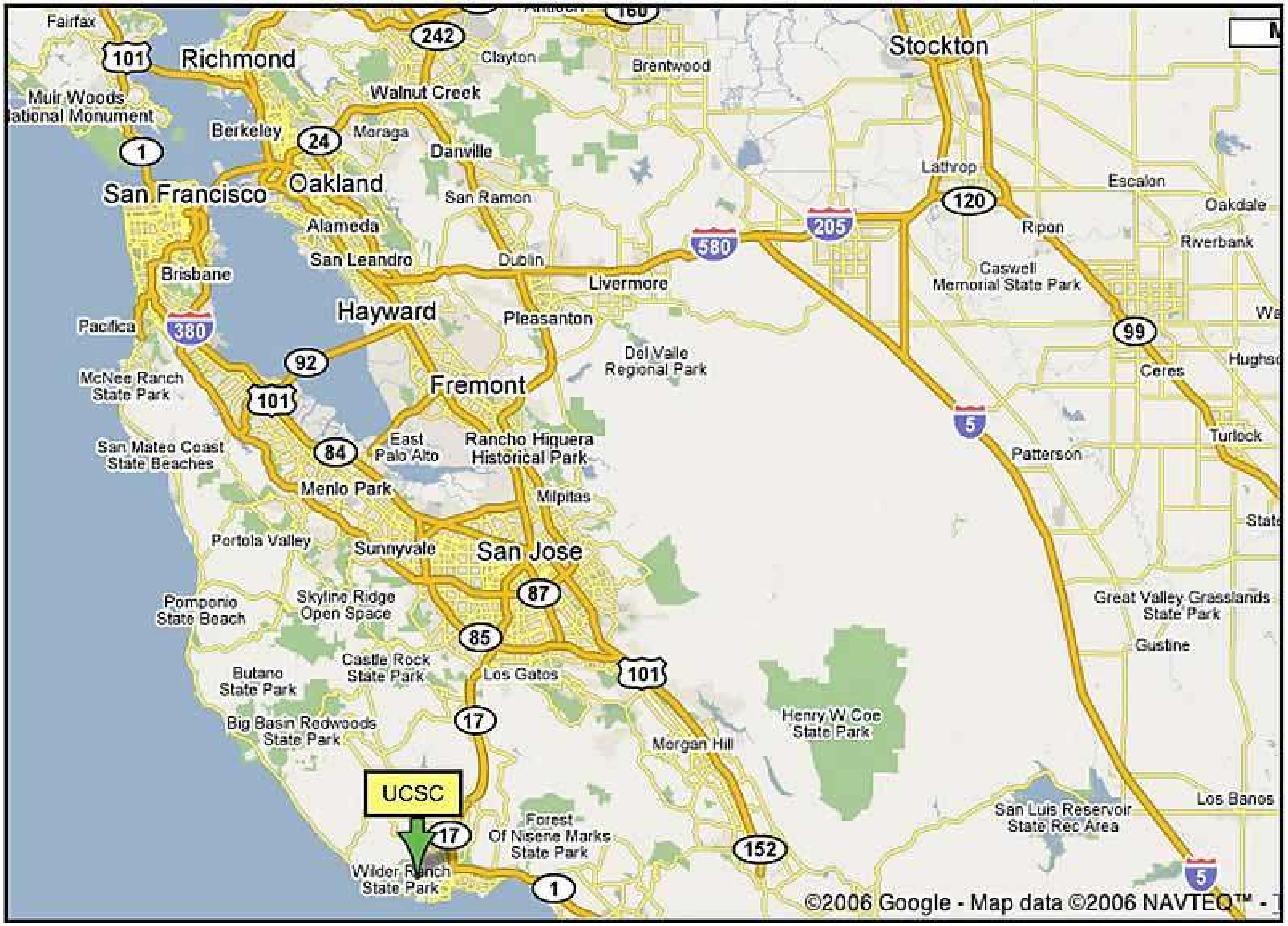
- The history of relational databases is the history of a scientific and technological revolution.
- The scientific revolution started in 1970 by Edgar (Ted) F. Codd at the IBM San Jose Research Laboratory (now the IBM Almaden Research Center)
- Codd introduced the relational data model and two database query languages: relational algebra and relational calculus.
  - "A relational model for data for large shared data banks", CACM, 1970.
  - "Relational completeness of data base sublanguages", in: Database Systems, ed. by R. Rustin, 1972.

Edgar F. Codd, 1923-2003



Relational Databases: A Very Brief History

- Researchers at the IBM San Jose Laboratory embark on the System R project, the first implementation of a relational database management system (RDBMS)
  - In 1974-1975, they develop SEQUEL, a query language that eventually became the industry standard SQL.
  - System R evolved to DB2 – released first in 1983.
- M. Stonebraker and E. Wong embark on the development of the Ingres RDBMS at UC Berkeley in 1973.
  - Ingres is commercialized in 1983; later, it became PostgreSQL, a free software OODBMS (object-oriented DBMS).
- L. Ellison founds a company in 1979 that eventually becomes Oracle Corporation; Oracle V2 is released in 1979 and Oracle V3 in 1983.
- Ted Codd receives the ACM Turing Award in 1981.


Fairfax
Richmond
Muir Woods
Berkeley
Moraga
Walnut Creek
Clayton
Brentwood
Stockton
Danville
San Francisco
Oakland
San Ramon
Lathrop
Escalon
Oakdale
Ripon
Riverbank
Alameda
San Leandro
Dublin
Livermore
Caswell Memorial State Park
Brisbane
Pacifica
Hayward
Pleasanton
Del Valle Regional Park
McNee Ranch State Park
Fremont
Ceres
San Mateo Coast State Beaches
East Palo Alto
Rancho Higuera Historical Park
Turlock
Patterson
Menlo Park
Milpitas
Portola Valley
Sunnyvale
San Jose
Pomponio State Beach
Skyline Ridge Open Space
Great Valley Grasslands State Park
Gustine
Castle Rock State Park
Los Gatos
Butano State Park
Big Basin Redwoods State Park
Henry W Coe State Park
Morgan Hill
UCSC
Forest Of Nisene Marks State Park
San Luis Reservoir State Rec Area
Los Banos
Wilder Ranch State Park
©2006 Google - Map data ©2006 NAVTEQ™

The Relational Data Model (E.F. Codd – 1970)

- The Relational Data Model uses the mathematical concept of a relation as the formalism for describing and representing data.
- Question: What is a relation?
- Answer:
  - Formally, a relation is a subset of a cartesian product of sets.
  - Informally, a relation is a "table" with rows and columns.

CHECKING Table

branch-name	**account-no**	**customer-name**	**balance**
Orsay	10991-06284	Abiteboul	$3,567.53
Hawthorne	10992-35671	Hull	$11,245.75
…	…	…	…

Basic Notions from Discrete Mathematics

- A k-ary relation R is a subset of a cartesian product of k sets, i.e.,
  - $R \subseteq D_1 \times D_2 \times \ldots \times D_k$.
- Examples:
  - Unary $R = \{0,2,4,\ldots,100\}$ $(R \subseteq D)$
  - Binary $T = \{(a,b)$: a and b have the same birthday$\}$
  - Ternary $S = \{(m,n,s): s = m+n\}$
  - ...

Relations and Attributes

- Note:
  $R \subseteq D_1 \times D_2 \times \ldots \times D_k$ can be viewed as a table with k columns

  Table S

3	5	8
150	100	250
...	...	...

- In the relational data model, we want to have names for the columns; these are the attributes of the relation.

Relation Schemas and Relational Database Schemas

- A k-ary relation schema $\mathbf{R}(A_1,A_2,\ldots,A_K)$ is a set $\{A_1,A_2,\ldots,A_k\}$ of k attributes.
  - **CHECKING**(branch-name, account-no, customer-name, balance)
    - Thus, a k-ary relation schema is a "blueprint" for k-ary relations.
    - It is a k-ary relation symbol in logic with names for the positions.

- An instance of a relation schema is a relation conforming to the schema (arities match; also, in DBMS, data types of attributes match).

- A relational database schema is a set of relation schemas $\mathbf{R_i}(A_1,A_2,\ldots,A_{k_i})$, for $1\leq i\leq m$.

- A relational database instance of a relational schema is a set of relations R_i each of which is an instance of the relation schema $\mathbf{R_i}$, $1\leq i\leq m$.

Relational Database Schemas - Examples

- BANKING relational database schema with relation schemas
  - CHECKING-ACCOUNT(branch, acc-no, cust-id, balance)
  - SAVINGS-ACCOUNT(branch, acc-no, cust-id, balance)
  - CUSTOMER(cust-id, name, address, phone, email)
  - ....
- UNIVERSITY relational database schema with relation schemas
  - STUDENT(student-id, student-name, major, status)
  - FACULTY(faculty-id, faculty-name, dpt, title, salary)
  - COURSE(course-no, course-name, term, instructor)
  - ENROLLS(student-id, course-no, term)
  - ...

Schemas vs. Instances

Keep in mind that there is a clear distinction between

- relation schemas and instances of relation schemas

and between

- relational database schemas and relational database instances.

Syntactic Notion	**Semantic Notion (discrete mathematics notion)**
Relation Schema	Instance of a relation schema (i.e., a relation)
Relational Database Schema	Relational database instance (i.e., a database)

Relational Structures vs. Relational Databases

- Relational Structure

  $$\mathbf{A} = (A, R_1,\ldots,R_m)$$

  - A is the **universe** of **A**
  - $R_1,\ldots,R_m$ are the relations of **A**

- Relational Database

  $$\mathbf{D} = (R_1,\ldots,R_m)$$

- Thus, a relational database can be thought of as a relational structure **without** its universe.
  - And this causes some problems down the road …

Query Languages for the Relational Data Model

Codd introduced two different query languages for the relational data model:

- Relational Algebra, which is a procedural language.
  - It is an algebraic formalism in which queries are expressed by applying a sequence of operations to relations.
- Relational Calculus, which is a declarative language.
  - It is a logical formalism in which queries are expressed as formulas of first-order logic.

Codd's Theorem: Relational Algebra and Relational Calculus are "essentially equivalent" in terms of expressive power.

(but what does this really mean?)

The Five Basic Operations of Relational Algebra

- Group I: Three standard set-theoretic binary operations:
  - Union
  - Difference
  - Cartesian Product.
- Group II. Two special unary operations on relations:
  - Projection
  - Selection.
- Relational Algebra consists of all expressions obtained by combining these five basic operations in syntactically correct ways.

Relational Algebra: Standard Set-Theoretic Operations

- Union
  - Input: Two k-ary relations R and S, for some k.
  - Output: The k-ary relation $R \cup S$, where
    $R \cup S = \{(a_1,\ldots,a_k): (a_1,\ldots,a_k) \text{ is in } R \text{ or } (a_1,\ldots,a_k) \text{ is in } S\}$

- Difference:
  - Input: Two k-ary relations R and S, for some k.
  - Output: The k-ary relation $R - S$, where
    $R - S = \{(a_1,\ldots,a_k): (a_1,\ldots,a_k) \text{ is in } R \text{ and } (a_1,\ldots,a_k) \text{ is not in } S\}$

- Note:
  - In relational algebra, both arguments to the union and the difference must be relations of the same arity.
  - In SQL, there is the additional requirement that the corresponding attributes must have the same data type.

Relational Algebra: Standard Set-Theoretic Operations

- Cartesian Product
  - Input: An m-ary relation R and an n-ary relation S
  - Output: The (m+n)-ary relation $R \times S$, where

$$R \times S = \{(a_1,\ldots,a_m,b_1,\ldots,b_n): (a_1,\ldots a_m) \text{ is in } R \text{ and } (b_1,\ldots,b_n) \text{ is in } S\}$$

- Note:

$$|R \times S| = |R| \times |S|$$

The Projection Operator

- Motivation:

  It is often the case that, given a table R, one wants to:
  - Rearrange the order of the columns
  - Suppress some columns
  - Do both of the above.

- Fact: The Projection Operation is tailored for this task

The Projection Operation

- Projection is a family of unary operations of the form

$$\pi_{\text{<attribute list>}} (\text{<relation name>})$$

- The intuitive description of the projection operation is as follows:
  - When projection is applied to a relation R, it removes all columns whose attributes do not appear in the <attribute list>.
  - The remaining columns may be re-arranged according to the order in the <attribute list>.
  - Any duplicate rows are also eliminated.

The Projection Operation - Example

SAVINGS

branch-name	acc-no	cust-name	balance
Aptos	153125	Vianu	3,450
Santa Cruz	123658	Hull	2,817
San Jose	321456	Codd	9,234
San Jose	334789	Codd	875

$\pi_{\text{cust-name,branch-name}}(\text{SAVINGS})$

cust-name	branch-name
Vianu	Aptos
Hull	Santa Cruz
Codd	San Jose

More on the Syntax of the Projection Operation

- In relational algebra, attributes can be referenced by position no.

- Projection Operation:
  - Syntax: $\pi_{i_1,\ldots,i_m}(R)$, where R is of arity k, and i_1, ….i_m are distinct integers from 1 up to k.
  - Semantics:

    $\pi_{i_1,\ldots,i_m}(R) = \{(a_1,\ldots,a_m)$: there is a tuple $(b_1,\ldots,b_k)$ in R such that $a_1 = b_{i_1}, \ldots, a_m = b_{i_m}\}$

- Example: If R is R(A,B,C,D), then $\pi_{C,A}(R) = \pi_{3,1}(R)$

The Selection Operation

- Motivation:
  - Consider the table
    SAVINGS(branch-name, acc-no, cust-name, balance)
  - We may want to extract the following information from it:
    - Find all records in the Aptos branch
    - Find all records with balance at least $50,000
    - Find all records in the Aptos branch with balance less than $1,000
- Fact: The Selection Operation is tailored for this task.

The Selection Operation

- Selection is a family of unary operations of the form

$$\sigma_{\Theta}\ (R),$$

where R is a relation and Θ is a condition that can be applied as a test to each row of R.

- When a selection operation is applied to R, it returns the subset of R consisting of all rows that satisfy the condition Θ

- Question: What is the precise definition of a “condition”?

The Selection Operation

- Definition: A condition in the selection operation is an expression built up from:
  - Comparison operators $=, <, >, \neq, \leq, \geq$ applied to operands that are constants or attribute names or component numbers.
    - These are the basic (atomic) clauses of the conditions.
  - The Boolean logic operators $\wedge$, $\vee$, $\neg$ applied to basic clauses.
- Examples:
  - balance > 10,000
  - branch-name = "Aptos"
  - (branch-name = "Aptos") $\wedge$ (balance < 1,000)

The Selection Operator

- Note:
  - The use of the comparison operators <, >, ≤, ≥ assumes that the underlying domain of values is totally ordered.
  - If the domain is not totally ordered, then only = and ≠ are allowed.
  - If we do not have attribute names (hence, we can only reference columns via their component number), then we need to have a special symbol, say \$, in front of a component number. Thus,
    - \$4 > 100 is a meaningful basic clause
    - \$1 = "Aptos" is a meaningful basic clause, and so on.

Relational Algebra

- Definition: A relational algebra expression is a string obtained from relation schemas using union, difference, cartesian product, projection, and selection.

- Context-free grammar for relational algebra expressions:

  $E := R, S, \ldots \mid (E_1 \cup E_2) \mid (E_1 - E_2) \mid (E_1 \times E_2) \mid \pi_L(E) \mid \sigma_\Theta(E)$,

  where
  - R, S, … are relation schemas
  - L is a list of attributes
  - Θ is a condition.

Strength from Unity and Combination

- By itself, each basic relational algebra operation has limited expressive power, as it carries out a specific and rather simple task.
- When used in combination, however, the five relational algebra operations can express interesting and, quite often, rather complex queries.
- Derived relational algebra operations are operations on relations that are expressible via a relational algebra expression (built from the five basic operators).

Intersection

- Intersection
  - Input: Two k-ary relations R and S, for some k.
  - Output: The k-ary relation $R \cap S$, where

$$R \cap S = \{(a_1,\ldots,a_k): (a_1,\ldots,a_k) \text{ is in } R \text{ and } (a_1,\ldots,a_k) \text{ is in } S\}$$

- Fact: $R \cap S = R - (R - S) = S - (S - R)$

Thus, intersection is a derived relational algebra operation.

Natural Join

- Fact: The most FAQs against databases involve the natural join operation $\bowtie$.

- Motivating Example: Given
  TEACHES(fac-name,course,term) and
  ENROLLS(stud-name,course,term),

  we want to obtain
  TAUGHT-BY(stud-name,course,term,fac-name)

  It turns out that TAUGHT-BY = ENROLSS $\bowtie$ TEACHES

Natural Join

Given TEACHES(fac-name,course,term) and
ENROLLS(stud-name, course,term):

To compute TAUGHT-BY(stud-name,course,term,fac-name)

1. ENROLLS $\times$ TEACHES
2. $\sigma_{\text{T.course = E.course} \wedge \text{T.term = E.term}}$ (ENROLLS $\times$ TEACHES)
3. $\pi_{\text{stud-name,E.course,E.term,fac-name}}$ $(\sigma_{\text{T.course = E.course} \wedge \text{T.term = E.term}}$ (ENROLLS $\times$ TEACHES))

The result is ENROLLS $\bowtie$ TEACHES.

Natural Join

- Definition: Let A1, …, Ak be the common attributes of two relation schemas R and S. Then

$$R \bowtie S = \pi_{\text{<list>}} (\sigma_{R.A1=S.A1 \wedge \ldots \wedge R.AK = S.Ak} (R \times S)),$$

where <list> contains all attributes of R×S, except for

S.A1, …, S.Ak (in other words, duplicate columns are eliminated).

- Algorithm for R ⋈ S:

  For every tuple in R, compare it with every tuple in S as follows:

  - test if they agree on all common attributes of R and S;
  - if they do, take the tuple in R × S formed by these two tuples,
  - remove all values of attributes of S that also occur in R;
  - put the resulting tuple in R ⋈ S.

Quotient (Division)

- Definition: Let R be a relation of arity r and let S be a relation of arity s, where r > s.

  The quotient (or division) $R \div S$ is the relation of arity r – s consisting of all tuples $(a_1,\ldots,a_{r-s})$ such that for every tuple $(b_1,\ldots,b_s)$ in S, we have that $(a_1,\ldots,a_{r-s}, b_1,\ldots,b_s)$ is in R.

- Example: Given

  ENROLLS(stud-name,course) and TEACHES(fac-name,course), find the names of students who take every course taught by V. Vianu

  (assumption: every course is taught by only one instructor)

  - Find the courses taught by V. Vianu

    $$\pi_{\text{course}} (\sigma_{\text{fac-name = "V. Vianu"}} (\text{TEACHES}))$$

  - The desired answer is given by the expression:

    $$\text{ENROLLS} \div \pi_{\text{course}} (\sigma_{\text{fac-name = "V. Vianu"}} (\text{TEACHES}))$$

Quotient (Division)

Fact: The quotient operation is expressible in relational algebra.

Proof: For concreteness, assume that R has arity 5 and S has arity 2.

Key Idea: Use the difference operation

- $R \div S = \pi_{1,2,3}(R)$ – "tuples in $\pi_{1,2,3}(R)$ that do not make it to $R \div S$"
- Consider the relational algebra expression $(\pi_{1,2,3}(R) \times S) - R$.

  Intuitively, it is the set of all tuples that fail the test for membership in $R \div S$. Hence,

- $R \div S = \pi_{1,2,3}(R) - \pi_{1,2,3}((\pi_{1,2,3}(R) \times S) - R)$.

Independence of the Basic Relational Algebra Operations

- Question: Are all five basic relational algebra operations really needed? Can one of them be expressed in terms of the other four?

- Theorem: Each of the five basic relational algebra operations is independent of the other four, that is, it cannot be expressed by a relational algebra expression that involves only the other four.

Proof Idea: For each relational algebra operation, we need to discover a property that is possessed by that operation, but is not possessed by any relational algebra expression that involves only the other four operations.

Exercise: Complete the proof for cartesian product, projection, and difference.

Independence of the Basic Relational Algebra Operations

Theorem: Each of the five basic relational algebra operations is independent of the other four, that is, it cannot be expressed by a relational algebra expression that involves only the other four.

Proof Sketch: (projection and cartesian product only)

- Property of projection:
  - It is the only operation whose output may have arity smaller than its input.
  - Show, by induction, that the output of every relational algebra expression in the other four basic relational algebra is of arity at least as big as the maximum arity of its arguments.

- Property of cartesian product:
  - It is the only operation whose output has arity bigger than its input.
  - Show, by induction, that the output of every relational algebra expression in the other four basic relational algebra is of arity at most as big as the maximum arity of its arguments.

Exercise: Complete this proof.

SQL vs. Relational Algebra

SQL	**Relational Algebra**
SELECT	Projection π
FROM	Cartesian Product $\times$
WHERE	Selection σ

Semantics of SQL via interpretation to Relational Algebra

SELECT $R_{i1}.A1, \ldots, R_{im}.A.m$
FROM $R_1, \ldots, R_K$ $\quad = \quad \pi_{Ri1.A1, \ldots, Rim.A.m} (\sigma_{\Psi} (R_1 \times \ldots \times R_K))$
WHERE Ψ

Relational Calculus

- In addition to relational algebra, Codd introduced relational calculus.
- Relational calculus is a declarative database query language based on first-order logic.
- Relational calculus comes into two different flavors:
  - Tuple relational calculus
  - Domain relational calculus.

  We will focus on domain relational calculus.

  There is an easy translation between these two formalisms.
- Codd's main technical result is that relational algebra and relational calculus have "essentially" the same expressive power.

Relational Calculus (First-Order Logic for Databases)

- First-order variables: $x, y, z, \ldots, x_1, \ldots, x_k, \ldots$
  - They range over values that may occur in tables.
- Relation symbols: R, S, T, … of specified arities (names of relations)
- Atomic (Basic) Formulas:
  - $R(x_1,\ldots,x_k)$, where R is a k-ary relation symbol (alternatively, $(x_1,\ldots,x_k) \in R$; the variables need not be distinct)
  - $(x \text{ op } y)$, where op is one of $=, \neq, <, >, \leq, \geq$
  - $(x \text{ op } c)$, where c is a constant and op is one of $=, \neq, <, >, \leq, \geq$.
- Relational Calculus Formulas:
  - Every atomic formula is a relational calculus formula.
  - If φ and ψ are relational calculus formulas, then so are:
    - $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $\neg \psi$, $(\varphi \rightarrow \psi)$ (propositional connectives)
    - $(\exists\, x\, \varphi)$ (existential quantification)
    - $(\forall\, x\, \varphi)$ (universal quantification).

Relational Calculus as a Database Query Language

Definition:

- A relational calculus expression is an expression of the form

$$\{ (x_1,\ldots,x_k):\ \varphi(x_1,\ldots x_k) \},$$

  where $\varphi(x_1,\ldots,x_k)$ is a relational calculus formula with $x_1,\ldots,x_k$ as its free variables.
- When applied to a relational database I, this relational calculus expression returns the k-ary relation that consists of all k-tuples $(a_1,\ldots,a_k)$ that make the formula "true" on I.

Example: The relational calculus expression

$$\{ (x,y):\ \exists z(E(x,z) \wedge E(z,y)) \}$$

returns the set P of all pairs of nodes (a,b) that are connected via a path of length 2.

Relational Calculus as a Database Query Language

Example: FACULTY(name, dpt, salary), CHAIR(dpt, name)
Give a relational calculus expression for C-SALARY(dpt,salary)
(find the salaries of department chairs).

$$\{(x,y):\ \exists u(\text{FACULTY}(u,x,y) \land \text{CHAIR}(x,u))\}$$

Here is another relational calculus expression for the same task:

$$\{(x,y):\ \exists u \exists v(\text{FACULTY}(u,x,y) \land \text{CHAIR}(x,v) \land (u=v))\}$$

Relational Calculus as a Database Query Language

Example: FACULTY(name, dpt, salary)

Find the names of the highest paid faculty in CS

$\{x: \varphi(x)\}$, where $\varphi(x)$ is the formula:

$$\exists y,z\,(\text{FACULTY}(x,y,z) \wedge y = \text{“CS”} \wedge (\forall u,v,w(\text{FACULTY}(u,v,w) \wedge v = \text{“CS”} \rightarrow z \geq w)))$$

Exercise: Express this query in relational algebra and in SQL.

Abbreviation:

- $\exists x_1,\ldots,x_k$ stands for $\exists x_1,\ldots,\exists x_k$
- $\forall x_1,\ldots,x_k$ stands for $\forall x_1,\ldots,\forall x_k$

Natural Join in Relational Calculus

Example: Let R(A,B,C) and S(B,C,D) be two ternary relation schemas.

- Recall that, in relational algebra, the natural join $R \bowtie S$ is given by

$$\pi_{R.A,R.B,R.C,S.D}\,(\sigma_{R.B = S.B \wedge R.C = S.C}\,(R \times S)).$$

- Give a relational calculus expression for $R \bowtie S$

$$\{(x_1,x_2,x_3,x_4):\ R(x_1,x_2,x_3) \wedge S(x_2,x_3,x_4)\}$$

Note: The natural join is expressible by a quantifier-free formula of relational calculus.

Quotient in Relational Calculus

- Recall that the quotient (or division) $R \div S$ of two relations R and S is the relation of arity $r - s$ consisting of all tuples $(a_1,\ldots,a_{r-s})$ such that for every tuple $(b_1,\ldots,b_s)$ in S, we have that $(a_1,\ldots,a_{r-s}, b_1,\ldots,b_s)$ is in R.

- Assume that R has arity 5 and S has arity 3.

  Express $R \div S$ in relational calculus.

$$\{ (x_1,x_2): (\forall x_3)(\forall x_4)(\forall x_5) (S(x_3,x_4,x_5) \rightarrow R(x_1,x_2,x_3,x_4,x_5)) \}$$

  - Much simpler than the relational algebra expression for $R \div S$

Relational Algebra vs. Relational Calculus

Codd's Theorem (informal statement):
Relational Algebra and Relational Calculus have "essentially" the same expressive power, i.e., they can express the same queries.

Note: It is **not** true that for every relational calculus expression φ, there is an equivalent relational algebra expression E.

Examples:

- $\{ (x_1,\ldots,x_k): \neg R(x_1,\ldots,x_k) \}$
- $\{ x: \forall y,z \text{ ENROLLS}(x,y,z) \}$, where ENROLLS(s-name,course,term)

From Relational Calculus to Relational Algebra

Note: The previous relational calculus expression may produce different answers when we consider different domains over which the variables are interpreted.

Example: If the variables $x_1,\ldots,x_k$ range over a domain D, then

$$\{(x_1,\ldots,x_k): \neg R(x_1,\ldots,x_k)\} = D^k - R.$$

Fact:

- The relational calculus expression $\{ (x_1,\ldots,x_k): \neg R(x_1,\ldots,x_k) \}$ is **not** "domain independent".
- The relational calculus expression $\{(x_1,\ldots,x_k): S(x_1,\ldots,x_k) \wedge \neg R(x_1,\ldots,x_k)\}$ is "domain independent".