Logic, Data, and Incomplete Information

Phokion G. Kolaitis

UC Santa Cruz & IBM Research

Logic and Databases are inextricably intertwined.

C.J. Date -- 2007

Logic and Databases

- Extensive interaction between logic and databases during the past 50 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

Two main uses of logic in databases:

- Logic is used as a database query language to express questions asked against databases.

- Logic is used as a specification language to express integrity constraints in databases.

We will discuss both these uses of logic in databases with emphasis on the interaction of logic and databases in the presence of incomplete information.

Thematic Roadmap

- Logic and Database Query Languages
  - Relational Algebra, Relational calculus, Codd's Theorem
- Overview of computational complexity
  - Main complexity classes, reductions, and complete problems
- Query Evaluation on a single database
  - Data complexity and combined complexity
- Query Evaluation on a collection of databases
  - Modeling incomplete information
  - Certain answers and possible answers

On the Modeling of Incomplete Information

- Traditionally, a query is evaluated on a single database in which there is no ambiguity or uncertainty about the data.
- There are several different settings, however, in which (some of) the data may be “uncertain” or “incomplete”.
- In such cases, we have an “incomplete” database that, in effect, is a compact representation of several different “complete” databases.
- Evaluating a query on an “incomplete” database gives rise to the notions of the certain answers and the possible answers.
- We will discuss four different such settings of “incomplete” databases and the associated notions of the certain/possible answers.

Thematic Roadmap - Continued

- Inconsistent Databases
  - Logic and integrity constraints in databases
  - Query answering: Semantics and Complexity
- Probabilistic Databases
  - Query answering: Semantics and Complexity
- Data Exchange and Data Integration
  - Schema mappings and schema-mapping languages
  - Query answering: Semantics and Complexity
- Computational Social Choice Meets Databases
  - Incomplete information in social choice, election databases
  - Query answering: Semantics and Complexity

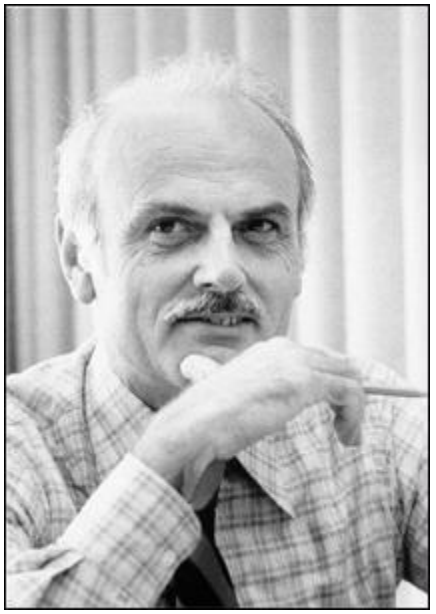
Tentative Outline by Lecture

- Lecture 1:
  - Relational Algebra and Relational Calculus
  - Overview of Computational Complexity
- Lecture 2:
  - Complexity of Query Evaluation
  - Integrity Constraints in databases, Inconsistent databases
- Lecture 3:
  - Repairs, Consistent answers, Complexity of Consistent Answers
  - Data transformations, Schema Mappings
- Lecture 4:
  - Schema Mappings and Data Exchange, Probabilistic Databases
- Lecture 5:
  - Computational Social Choice
  - Election databases, Necessary and Possible Answers

Relational Databases: How it all got started

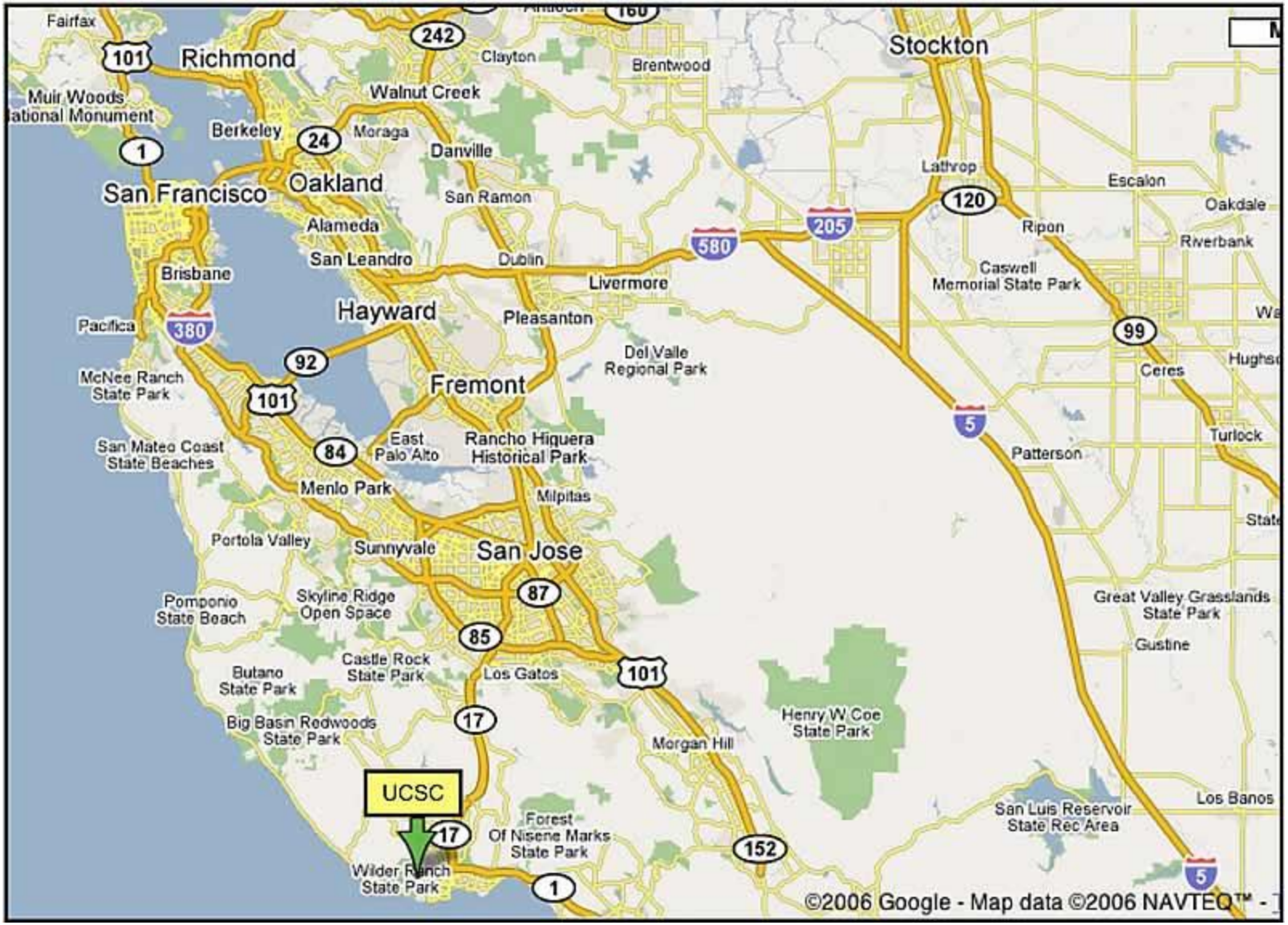
- The history of relational databases is the history of a scientific and technological revolution.
- The scientific revolution started in 1970 by Edgar (Ted) F. Codd at the IBM San Jose Research Laboratory (now the IBM Almaden Research Center)
- Codd introduced the relational data model and two database 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
National Monument
Berkeley
Oakland
San Francisco
Walnut Creek
Clayton
Brentwood
Stockton
Moraga
Danville
Lathrop
Escalon
San Ramon
Alameda
San Leandro
Dublin
Livermore
Ripon
Oakdale
Riverbank
Caswell
Memorial State Park
Brisbane
Hayward
Pleasanton
Pacifica
Del Valle
Regional Park
Ceres
McNee Ranch
State Park
Fremont
Turlock
San Mateo Coast
State Beaches
East
Palo Alto
Rancho Higuera
Historical Park
Patterson
Menlo Park
Milpitas
Portola Valley
Sunnyvale
San Jose
Pomponio
State Beach
Skyline Ridge
Open Space
Great Valley Grasslands
State Park
Gustine
Butano
State Park
Castle Rock
State Park
Los Gatos
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
    $R \subseteq B_1 \times B_2 \times \ldots \times B_k$.
  - Informally, a relation is a "table" with rows and columns.

CHECKING Table

branch-name	**account-no**	**customer-name**	**balance**
Orsay	10991-06284	Abiteboul	$13,567.53
Hawthorne	10992-35671	Hull	$21,245.75
…	…	…	…

Relational Database Schemas

- A k-ary relation schema **R**(A_1,A_2,…,A_K) is a set { A_1,A_2,…,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.
  - k is called the arity of the relation schema
- An instance of a relation schema is a relation conforming to the schema (arities match; 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 or, simply, a database over 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 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 …

Database Queries

- Users interact with databases by posing questions and receiving answers to their questions.

- The questions posed against databases are called database queries or, simply, queries.

- To make a formal study of queries we need to
  1. Give a precise definition of the notion of a query.
  2. Describe a language or languages for formulating queries.

Database Queries

Definition: Let **S** be a relational database schema and let $k \geq 1$.

A k-ary query on **S** is a function q defined on databases over **S** such that if D is a database over **S**, then q(D) is a k-ary relation with entries from the relations in D that is invariant under isomorphisms

(i.e., if h: D $\rightarrow$ F is an isomorphism, then q(F) = h(q(D)).

Example 1: TEACHES(instructor, course), ENROLLS(student,course)

- Find all students enrolled in some course taught by Miller.
- Find all students who are enrolled in every course taught by Miller.

  Unary queries (1-ary queries)

Example 2: FLIGHT(from, to)

- Find all pairs of cities such that one can fly to the other with at most one stop.

  Binary query (2-ary query)

Database Queries

Definition: Let **S** be a relational database schema.
A Boolean query on **S** is a function q defined on database instances over S such that if D is a database over S, then q(D) = 1 or q(D) = 0, and q(D) is invariant under isomorphisms.

Example 1:

CHECKING(branch-name, account-no, customer-name, balance)
Is there a customer at the San Jose branch whose account has a balance of more than 1 Million dollars?

Example 2: EDGE(node1,node2)

- Does the graph have diameter at most 3?
- Is the graph connected?

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.

Set-Theoretic Operations

- Union: R and S two k-ary relations

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

- Difference: R and S two k-ary relations

$R - S = \{ (a_1,\ldots,a_k): (a_1,\ldots,a_k) \text{ is in R and } (a_1,\ldots,a_k) \text{ is not in S } \}$

- Cartesian Product: R is a k-ary relation, S is a m-ary relation

$R \times S = \{ (a_1,\ldots,a_{k,}\, b_1,\ldots,b_m): (a_1,\ldots,a_k) \text{ is in R and } (b_1,\ldots,b_m) \text{ is in S } \}$

The Projection Operation

- Projection Operation:
  - Syntax: $\pi_{i_1,\ldots,i_m}(R)$, where R is of arity k, and $i_1, \ldots i_m$ 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 $a_1 = b_{i_1}, \ldots, a_m = b_{i_m}\}$

  - Names of attributes can also be used, instead of indices

- Example: If R is R(A,B,C,D), then

  $\pi_{3,1}(R) = \{(c,a)$: there are b,d such that $(a,b,c,d) \in R\} =$

  $\pi_{C,A}(R)$

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

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

- A condition in the selection operation is an expression built up from:
  - Comparison operators =, <, >, ≠, ≤, ≥ applied to operands that are constants or attribute names or component numbers.
    - These are the basic (atomic) clauses of the conditions.
  - Boolean combinations ($\wedge$, $\vee$, $\neg$) of basic clauses.

The Selection Operation

Example:

CHECKING(branch-name, account-no, customer-name, balance)

- Find the records at the Hawthorne branch in which the balance is less than $10000

- σ (branch-name='Hawthorne') $\wedge$ (balance < 10000) (**CHECKING**)

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, … | $(E_1 \cup E_2)$ | $(E_1 - E_2)$ | $(E_1 \times E_2)$ | $\pi_L(E)$ | $\sigma_\Theta(E)$,

  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

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

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

  where <list> contains all attributes of $R \times S$, except for S.A1, …, S.Ak
  (in other words, duplicate columns are eliminated).

- 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)
  Then
  TAUGHT-BY = ENROLLS $\bowtie$ TEACHES

Quotient (Division)

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

  The quotient (or division) R ÷ 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

  - 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)

R ÷ S is the relation of arity r – s 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.

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 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 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 it is not possessed by any relational algebra expression that involves only the other four operations.

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.

Propositional (Boolean) Logic - Reminder

- Propositional variables: x, y, z, …
  - They take values 0 (True) and 1 (False).

- Propositional connectives: $\wedge$, $\vee$, $\neg$, $\rightarrow$

- Propositional formulas: expressions built from propositional variables and propositional connectives
  - Syntax: φ := x, y, z, … | $(\psi \wedge \chi)$ | $(\psi \vee \chi)$ | $\neg \psi$ | $(\psi \rightarrow \chi)$
  - Semantics: Truth-table semantics

- Application: Propositional formulas express Boolean functions
  - $(x \vee y) \wedge (\neg x \vee \neg y)$ XOR-Gate
  - $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ Majority Gate

In fact, every Boolean function can be expressed by a propositional formula.

First-Order Logic

- Question: What is First-Order Logic?
- Answer: Informally,

" First-Order Logic = Propositional Logic + (∃ and ∀)",

where

∃ and ∀ range over *possible values occurring in relations*.

Relational Calculus (FO Logic for Databases)

- First-order variables: x, y, z, …, x_1, …,x_k,…
  - 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 op y), where op is one of =, ≠, <, >, ≤, ≥
  - (x op c), where c is a constant and op is one of =, ≠, <, >, ≤, ≥.
- 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 - Examples

- Examples: Assume E is a binary relation symbol
  - $(\exists\, x)E(x,x)$
  - $(\forall\, x)(\forall\, y)(\exists\, z)(E(x,z) \wedge E(z,y))$
  - $(\exists\, z_1)(\exists\, z_2)(E(x,z_1) \wedge E(z_1,z_2) \wedge E(z_2,y))$
  - $(\exists\, y)(\exists\, z)(E(x,y) \wedge E(x,z) \wedge (y \neq z))$

- Free and bound variables:
  - In the first two formulas above, no variable is free.
  - In the third formula above, the free variables are x and y.
  - In the fourth formula above, the only free variable is x.
  - Intuitively, a variable is free in a formula if the variable must be assigned a value in order to tell if the formula is true or false.

Free and Bound Variables

- A sentence is a first-order formula ψ with no free variables.
  - $(\exists x)E(x,x)$
  - $(\forall x)(\forall y)(\exists z)(E(x,z) \wedge E(z,y))$
  - $(\forall x)(\forall y)(x < y \rightarrow (\exists z)(x < z \wedge z < y))$
  - On every relational database D, a sentence is either true or false.
    - Either $D \models \psi$ or $D \models \neg \psi$

- If a first-order formula has at least one free variable, then it makes no sense to tell whether it is true or false on a relational database D. Instead, we need to also assign values to its free variables
  - $D, 3, 5 \models \exists z\ (x < z \wedge z < y)$
  - $D, 3, 4 \models \neg \exists z\ (x < z \wedge z < y)$,

  where D is the linear order < on the natural numbers 1, 2, 3, …

Relational Calculus as a 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 database D, 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 D.

Fact:

- Every relational calculus expression $\{ (x_1,\ldots,x_k) : \varphi(x_1,\ldots x_k) \}$ defines a k-ary query.

Relational Calculus as a Query Language

Example 1: TEACHES(instructor, course), ENROLLS(student,course)

- Find all students enrolled in some course taught by Miller.

  {s: ∃ t ∃ c (TEACHES(t, c) ∧ t = 'Miller' ∧ ENROLLS(s,c)) }

- Find all students who are enrolled in every course taught by Miller.

  {s: ∀ t ∀ c (TEACHES(t, c) ∧ t = 'Miller' → ENROLLS(s,c)) }

Example 2: FLIGHT(from, to)

- Find all pairs of cities such that one can fly to the other with at most one stop.

  {(x,y): FLIGHT(x,y) ∨ ∃ z (FLIGHT(x,z) ∧ FLIGHT(z,y)) }

Relational Calculus as a Query Language

Example 3: FACULTY(name, dpt, salary)

Find the names of the highest paid faculty in CS

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

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

Optional Exercise: Express this query in relational algebra.

Abbreviation:

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

Relational Calculus and Boolean Queries

Note:

Let ψ be a relational calculus sentence (i.e., ψ has no free variables).

Then ψ express a Boolean query:

- If $D \models \psi$, then the query returns 1 ("true"/"yes")
- If $D \models \neg \psi$, then the query returns 0 ("false"/"no")

Example: E(node1,node2)

Does the graph with edge relation E have diameter at most 3?

$\forall x \forall y \exists z \exists w (E(x,y) \vee (E(x,z) \wedge E(z,y)) \vee (E(x,z) \wedge E(z,w) \wedge E(w,y))$

Fact: The query "is the graph with edge relation E connected?"

is not expressible in relational calculus.

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 C, then

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

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 $\{(x_1,\ldots,x_k): S(x_1,..,x_k) \wedge \neg R(x_1,\ldots,x_k)\}$ is "domain independent".

Active Domain and Active Domain Interpretation

Definition:

- The active domain adom(D) of a relational database instance D is the set of all values that occur in the relations of D.

- Let $\varphi(x_1,\ldots,x_k)$ be a relational calculus formula and let D be a relational database instance. Then

$$\varphi^{adom}(D)$$

is the result of evaluating $\varphi(x_1,\ldots,x_k)$ over adom(D) and D, i.e.,

  - all variables and quantifiers are assumed to range over adom(D);
  - the relation symbols in φ are interpreted by the relations in D.

More on the Active Domain

Example: Let φ be $\forall y R(x,y)$ and D = {(1,2), (1,1)}.

- adom(D) = {1,2}
- φ^{adom} (D) = {1}
- Relational structure **A** = ({1,2,3}, {(1,2), (1,1)}
  Then, $\varphi(\mathbf{A}) = \emptyset$
  (1 is not returned because (1,3) is not in {(1,2), (1,1)}.

Note: This example also shows the importance that the universe of discourse has on the semantics of first-order logic

Equivalence of Relational Algebra and Calculus

Theorem: If q is a k-ary query, then the following statements are equivalent:

1. There is a relational algebra expression E such that
   q(D) = E(D), for every database instance D
   (in other words, q is expressible in relational algebra).

2. There is a relational calculus formula ψ such that
   q(D) = ψ^{adom} (D)
   (in other words, q is expressible in relational calculus under the active domain interpretation).

From Relational Algebra to Relational Calculus

Theorem: For every relational algebra expression E, there is an equivalent relational calculus expression $\{ (x_1,\ldots,x_k): \varphi(x_1,\ldots x_k) \}$.

Proof: By induction on the construction of rel. algebra expressions.

- If E is a relation R of arity k, then we take $\{ (x_1,\ldots,x_k): R(x_1,\ldots,x_k) \}$.
- Assume E_1 and E_2 are expressible by $\{ (x_1,\ldots,x_k): \varphi_1(x_1,\ldots,x_k) \}$ and $\{ (x_1,\ldots,x_k): \varphi_2(x_1,\ldots,x_k) \}$. Then
  - $E_1 \cup E_2$ is expressible by
    $\{ (x_1,\ldots,x_k): \varphi_1(x_1,\ldots,x_k) \vee \varphi_2(x_1,\ldots,x_k) \}$.
  - $E_1 - E_2$ is expressible by
    $\{ (x_1,\ldots,x_k): \varphi_1(x_1,\ldots,x_k) \wedge \neg\varphi_2(x_1,\ldots,x_k) \}$.
  - $E_1 \times E_2$ is expressible by
    $\{ (x_1,\ldots,x_k,y_1,\ldots,y_m): \varphi_1(x_1,\ldots,x_k) \wedge \varphi_2(y_1,\ldots,y_m) \}$

From Relational Algebra to Relational Calculus

Theorem: For every relational expression E, there is an equivalent relational calculus expression $\{(x_1,\ldots,x_k): \varphi(x_1,\ldots x_k)\}$.

Proof: (continued)

- Assume that E is expressible by $\{ (x_1,\ldots,x_k): \varphi(x_1,\ldots,x_k) \}$.
  Then
  - $\pi_{1,3}(E)$ is expressible by
    $\{ (x_1,x_3): (\exists\, x_2)(\exists\, x_4) \ldots(\exists\, x_k)\ \varphi(x_1,\ldots,x_k) \}$
  - $\sigma_{\Theta}(E)$ is expressible by
    $\{ (x_1,\ldots,x_k): \Theta^* \wedge \varphi(x_1,\ldots,x_k) \}$, where Θ^* is the rewriting of Θ as a formula of relational calculus.

From Relational Algebra to Relational Calculus

Example: R(A,B), S(C,D)

Translate $\pi_{1,4}(\sigma_{R.B=S.C}(R \times S))$ to relational calculus

1. R translates to R(x,y)
2. S translates to S(z,w)
3. $R \times S$ translates to $R(x,y) \wedge S(z,w)$
4. $\sigma_{R.B=S.C}(R \times S)$ translates to $(y=z) \wedge R(x,y) \wedge S(z,w)$
5. $\pi_{1,4}(\sigma_{R.B=S.C}(R \times S))$ translates to

   $\exists y \exists z ((y=z) \wedge R(x,y) \wedge S(z,w))$

   or, simply, to

   $\exists y (R(x,y) \wedge S(y,w))$

Equivalence of Relational Algebra and Calculus

Proof (Sketch):

1. $\Rightarrow$ 2. We showed this by induction on the construction of relational algebra expressions.

Key idea: Projection π is simulated using $\exists$

2. $\Rightarrow$ 1.

- Show first that for every relational database schema **S**, there is a relational algebra expression E such that for every database instance D, we have that adom(D) = E(D).

- Use the above fact and induction on the construction of relational calculus formulas to obtain a translation of relational calculus under the active domain interpretation to relational algebra.

Equivalence of Relational Algebra and Calculus

- In this translation, the most interesting part is the simulation of the universal quantifier $\forall$ in relational algebra.
  - It uses the logical equivalence $\forall y\psi \equiv \neg\exists y\neg\psi$
- As an illustration, consider $\forall y R(x,y)$.
  - $\forall y R(x,y) \equiv \neg\exists y\neg R(x,y)$
  - adom(D) = $\pi_1(R) \cup \pi_2(R)$

Rel.Calc. formula φ	Relational Algebra Expression for φ^{adom}
$\neg R(x,y)$	$(\pi_1(R) \cup \pi_2(R)) \times (\pi_1(R) \cup \pi_2(R)) - R$
$\exists y \neg R(x,y)$	$\pi_1((\pi_1(R) \cup \pi_2(R)) \times (\pi_1(R) \cup \pi_2(R)) - R)$
$\neg\exists y \neg R(x,y)$	$(\pi_1(R) \cup \pi_2(R)) - (\pi_1((\pi_1(R) \cup \pi_2(R)) \times (\pi_1(R) \cup \pi_2(R)) - R))$

Sublanguages of Relational Calculus

- Formulas of the relational calculus may have an arbitrarily large number of alternations of universal ∀ and existential ∃ quantifiers.

- In practice, many frequently asked queries can be expressed using relational calculus formulas that have simple syntactic structure.

- We will focus on two sublanguages of relational calculus:
  1. The sublanguage of conjunctive queries.
  2. The sublanguage of unions of conjunctive queries.

Conjunctive Queries

Definition:

- A conjunctive query (CQ) is a query expressible by a relational calculus formula in prenex normal form built from atomic formulas $R(y_1,\ldots,y_n)$, and $\wedge$ and $\exists$ only.

$$\{ (x_1,\ldots,x_k): \exists z_1 \ldots \exists z_m \chi(x_1, \ldots,x_k, z_1,\ldots,z_m) \}$$

- A union of conjunctive queries (UCQ) is a query expressible by a disjunction of relational calculus formulas each expressing a conjunctive query.

Examples:

- Path of length 2: (CQ)

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

- Path of length at most 3: (UCQ)

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

Conjunctive Queries

Example: TEACHES(instructor, course), ENROLLS(student, course)

TAUGHT-BY(student, instructor) is a conjunctive query

$$\{ (s,t) : \exists c \, (\text{TEACHES}(t,c) \wedge \text{ENROLLS}(s,c)) \}$$

Example: Every relational join is a quantifier-free conjunctive query:

P(A,B,C), R(B,C,D) two relation symbol

Relational Join $P \bowtie R = \{ (x,y,z,w): P(x,y,z) \wedge R(y,z,w) \}$

Example: Is there a triangle? (Boolean conjunctive query)

$$\exists x \exists y \exists z (E(x,y) \wedge E(y,z) \wedge E(z,x))$$

Note: In general, every Boolean conjunctive query express the existence of a "pattern".

Conjunctive Queries and SQL

- Conjunctive queries are among the most frequently asked queries

- SQL, the main commercial database query language, directly supports conjunctive queries using the construct

  SELECT … FROM … WHERE

Example: TEACHES(instructor, course), ENROLLS(student, course)

- TAUGHT-BY(student, instructor) is a conjunctive query

  $\{ (s,t) : \exists c \, (TEACHES(t,c) \wedge ENROLLS(s,c)) \}$

- SELECT ENROLLS.student, TEACHES.instructor
  FROM TEACHES, ENROLLS
  WHERE TEACHES.course = ENROLLS.course

Conjunctive Queries and SQL

Example: Every relational join is a conjunctive query:

P(A,B,C), R(B,C,D) two relation symbols

- $P \bowtie R = \{ (x,y,z,w): P(x,y,z) \wedge R(y,z,w) \}$

- SELECT P.A, P.B, P.C, R.D
  FROM P, R
  WHERE P.B = R.B AND P.C = R.C

Note: Conjunctive queries are also known as SELECT-PROJECT-JOIN queries or SPJ-queries.

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, …, R_{im}.A.m
FROM R_1, …,R_K
WHERE Ψ

$$= \pi_{R_{i1}.A1, \ldots, R_{im}.A.m} (\sigma_{\Psi} (R_1 \times \ldots \times R_K))$$