Relational Calculus

Database Design

Department of Computer Engineering

Sharif University of Technology

Maryam Ramezani maryam.ramezani@sharif.edu

Introduction

Relational Calculus

- Relational Calculus is an alternative way for expressing queries
  - Main feature: specify what you want, not how to get it
  - Many equivalent algebra "implementations" possible for a given calculus expression
- In short: SQL query without aggregation = relational calculus expression
- Relational algebra expression is similar to program, describing what operations to perform in what order
- It is a formal language based upon predicate calculus
- It allows you to express the set of tuples you want from a database using a formula.

Relational Calculus

- A **relational calculus** expression creates a new **relation**, which is specified in terms of variables that range **over rows of the stored database relations** (in **Tuple Relational Calculus (TRC)**) or **over columns of the stored relations** (in **Domain Relational Calculus (DRC)**).

- TRC: Variables range over tuples.
- DRC: Variables range over domain elements (= attribute values)
- Both TRC and DRC are subsets of first-order logic
  - We use some short-hand notation to simplify formulas

Relational Calculus

- **Relational calculus is descriptive, while relational algebra is prospective.**
- In a calculus expression, there is no order of operations to specify how to retrieve the query result—a calculus expression specifies only what information the result should contain.
  - This is the main distinguishing feature between relational algebra and relational calculus.
  - Relational calculus is considered to be a **nonprocedural** language. This differs from relational algebra, where we must write a sequence of operations to specify a retrieval request; hence relational algebra can be considered as a **procedural** way of stating a query.

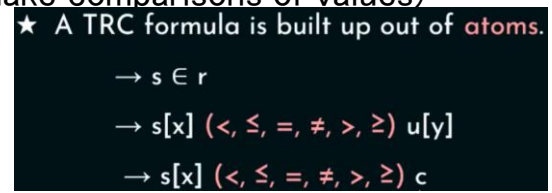
Tuple Relational Calculus

Tuple Relational Calculus

- The tuple relational calculus is based on specifying a number of tuple variables.
- Each tuple variable usually ranges over a particular database relation, meaning that the variable may take as its value any individual tuple from that relation.
- A simple tuple relational calculus query is of the form

$$\{t \mid p(t)\}$$

  - where t is a tuple variable and p(t) is a conditional expression involving t.
  - The result of such a query is the set of all tuples t that satisfy p(t).
  - p(t) denotes a formula in which tuple variable t appears.
- Answer: is the set of all tuples t for which the formula p(t) evaluates to true.
- Formula is recursively defined:
  - start with simple atomic formulas (get tuples from relations or make comparisons of values)
  - build bigger and better formulas using the logical connectives.



Tuple Relational Calculus

- A simple tuple relational calculus query is of the form

$$\{t \mid p(t)\}$$

- t.A or t[A]: the value of tuple t on attribute A
- R(t) or $t \in R$: tuple t is in relation R
- Example: Find the ID, name, dept_name, and salary for instructors whose salary is greater than $80,000.

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

notation1: $\{t | Instructor(t) \wedge t.salary > 80000\}$

notation2: $\{t | t \in Instructor \wedge t[salary] > 80000\}$

Domain Relational Calculus

Domain Relational Calculus

- **Query** has the form:

  {<x1, x2,⋯, xn> | p(<x1, x2,⋯, xn>)}

- Domain Variable – ranges over the values in the domain of some attribute or is a constant
- Example: If the domain variable x1 maps to attribute – Name char(20) then x1 ranges over all strings that are 20 characters long.
  - Not just the strings values in the relation's attribute
- **Answer** includes all tuples <x1, x2,⋯, xn> that make the formula p(<x1, x2,⋯, xn>) true.
- Last Example with DRC:

$$\{< i, n, d, s > \mid < i, n, d, s > \in Instructor(t) \wedge s > 80000\}$$

Quantifiers

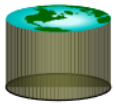
Free and Bound Variables

- **The use of quantifiers $\exists X$ and $\forall X$ in a formula is said to *bind* X in the formula.**
  - A variable that is not bound is free.

- **Let us revisit the definition of a query:**
  - $\{T \mid p(T)\}$

- **There is an important restriction**
  - the variable T that appears to the left of `|' must be the *only* free variable in the formula $p(T)$.
  - in other words, all other tuple variables must be bound using a quantifier.

Review



$a \Rightarrow b$ is the same as $\neg a \vee b$

a \ b	T	F
T	T	F
F	T	T

- **If a is true, b must be true for the implication to be true. If a is true and b is false, the implication evaluates to false.**
- **If a is not true, we don't care about b, the expression is always true.**

Review

- $P_1 \wedge P_2$ is equivalent to $\neg(\neg(P_1) \vee \neg(P_2))$.
- $\forall t \in r\ (P_1(t))$ is equivalent to $\neg\exists t \in r\ (\neg P_1(t))$.
- $P_1 \Rightarrow P_2$ is equivalent to $\neg(P_1) \vee P_2$.

The Existential and Universal Quantifiers

- ∀ is called the universal or "for all" quantifier because every tuple in "the universe of" tuples must make F true to make the quantified formula true.
- ∃ is called the existential or "there exists" quantifier because any tuple that exists in "the universe of" tuples may make F true to make the quantified formula true.
- Informally, a tuple variable t is bound if it is quantified, meaning that it appears in an (∀ t) or (∃ t) clause; otherwise, it is free.

  - FOR ALL X (F) = NOT EXISTS X (NOT F)
  - EXISTS X (F) = NOT (FORALL X (NOT F))
  - FORALL X (F) ⇒ EXISTS X (F)
  - NOT EXISTS X (F) ⇒ NOT FORALL X (F)
  - FORALL X (F AND G) = NOT EXISTS X (NOT(F) OR NOT(G))
  - FORALL X (F OR G) = NOT EXISTS X (NOT(F) AND NOT(G))
  - EXISTS X (F OR G) = NOT FORALL X (NOT(F) AND NOT(G))
  - EXISTS X (F AND G) = NOT FORALL X (NOT(F) OR NOT(G))

Examples

DRC

- Find all instructor name for instructors with salary greater than 80,000.

$$\{< n > | \exists i, d, s (< i, n, d, s > \in Instructor \wedge s > 80000)\}$$

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

- Example: To find the first and last names of all employees whose salary is above $50,000, we can write the following tuple calculus expression:

```
{t.FNAME, t.LNAME | EMPLOYEE(t) AND t.SALARY>50000}
```

- Retrieve the name and address of all employees who work for the 'Research' department. The query can be expressed as :

```
{t.FNAME, t.LNAME, t.ADDRESS | EMPLOYEE(t) and (∃ d)
(DEPARTMENT(d) and d.DNAME='Research' and d.DNUMBER=t.DNO)}
```

Another Relational Calculus Notation

(target-items) [WHERE F]

- STX.STID شماره تمام دانشجویان در رابطه STT
- STX.STID **WHERE** STX.STDEID='D11' شماره دانشجویان گروه آموزشی D11
- (STX.STID, STX.STL) **WHERE EXISTS** STCOX (STX.STID=STCOX.STID **AND** STCOX.COID='COM11')

شماره دانشجویی و مقطع تحصیلی آنهایی که درس COM11 را انتخاب کرده‌اند.

Another Relational Calculus Notation

- شماره همه تهیه کنندگان

SX.S#

- نام تهیه کنندگان شهرستان C2 که وضعیت آنها بزرگتر از 15 باشد.

SX.SNAME **WHERE** SX.CITY='C2' **AND** SX.STATUS> 15

- نام تهیه کنندگانی که حداقل یک قطعه آبی‌رنگ تهیه کرده‌اند.

SX.SNAME **WHERE EXISTS** SPX **(**SPX.S#=SX.S# **AND**

EXISTS PX (PX.P#=SPX.P# **AND** PX.COLOR='Blue') **)**

- نام جفت تهیه کنندگانی که در یک شهر بوده و حداقل یک قطعه مشترک تولید کرده‌اند.

SX.SNAME, SY.SNAME **WHERE** SX.CITY=SY.CITY **AND NOT** (SX.S#=SY.S#)

AND EXISTS SPX **(EXISTS** SPY SPX.S#=SX.S# **AND**

SPY.S#=SY.S# **AND** SPX.P#=SPY.P#**)**

Another Relational Calculus Notation

- عنوان درسهایی را بدهید که که تمام دانشجویان رشته کامپیوتر در ترم دوم ۹۹-۹۸ در آنها قبول شده‌اند.

STT(STID, Name, DEID, ...)

COT(COID, Title, Credit, ...)

STCOT(STID, COID, YR, TR, Grade)

p -> q = ~p v q

اگر کامپیوتر باشد، آن‌گاه در ترم دوم ۹۹-۹۸ قبول شده باشد!

COX.TITLE **WHERE FORALL** STX **(NOT** STX.STJ='CE' **OR**

EXISTS STCOX (STCOX.STID=STX.STID **AND** STCOX.COID=COX.COID **AND**

STCOX.YR='98-99' **AND** STCOX.TR='2' **AND** STCOX.GRADE>=10) **)**