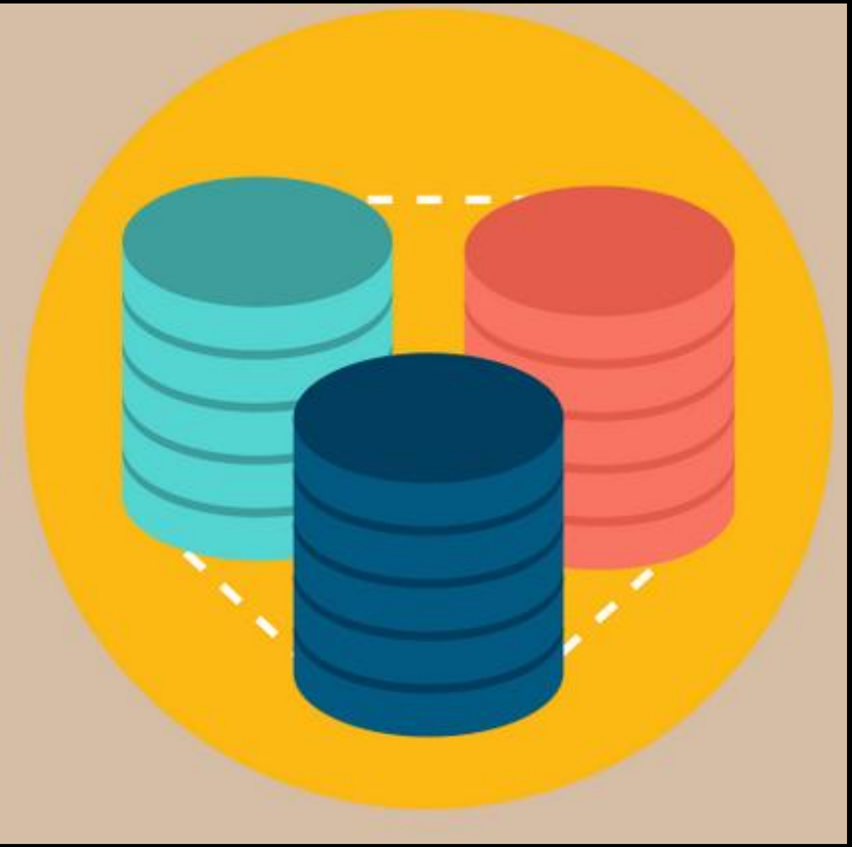

Relational Calculus

CE384: Database Design
Maryam Ramezani
Sharif University of Technology
maryam.ramezani@sharif.edu

01

Introduction

Relational Calculus

- Relational algebra is prescriptive/imperative.

- Relational calculus is declarative/descriptive.

- Similarity: Both use the concept of relation

- Difference: In Relational Calculus, unlike Relational Algebra, there are no operators. Instead, we use universal and existential quantifiers.

Relational Calculus

- 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.

- 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

Relational Calculus

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

- Both TRC and DRC are subsets of first-order logic
  - We use some short-hand notation to simplify formulas

02

Tuple Relational Calculus

Tuple Relational Calculus

- The tuple relational calculus is based on specifying a number of tuple variables.
- 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.

Tuple Relational Calculus

- A simple tuple relational calculus query is of the form

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

With two different notations (red and blue):

- 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: $\{t \mid Instructor(t) \wedge t.salary > 80000\}$

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

Example

- ID and Name of B.Sc. students of physics:

$$\{< t.STID, t.STNAME > | STUD(t) \wedge t.STLEV =' bs' \wedge STMJR = 'phys'\}$$

- ID of students who are classmate in the same class:

$$\{< x.STID, y.STID > | SCR(x) \wedge SCR(y) \wedge x.TR = y.TR \wedge x.YR = y.YR \wedge x.COID = y.COID \wedge x.STID < y.STID\}$$

STUD

STID	STNAME	STLEV	STMJR
888	st8	ms	math
444	st4	ms	phys
⋮	⋮	⋮	⋮

SCR

STID	COID	TR	YR	GRADE
888	co2	1	87	19
888	co3	1	87	10
444	co2	1	87	13
⋮	⋮	⋮	⋮	⋮

Example

- IDs of students who have taken at least one of the courses taken by the student with STID = '1024'.
  - at least one of = exists = ∃

$$\{< t.STID > | STUD(t) \wedge (\exists r | STUD(r) \wedge r.COID = t.COID \wedge r.STID = '1024')\}$$

STUD

STID	STNAME	STLEV	STMJR
888	st8	ms	math
444	st4	ms	phys
⋮	⋮	⋮	⋮

SCR

STID	COID	TR	YR	GRADE
888	co2	1	87	19
888	co3	1	87	10
444	co2	1	87	13
⋮	⋮	⋮	⋮	⋮

Example

- IDs of students who have taken <u>all</u> the courses taken by the student with STID = '1024'.
  - All = ∀

$$\{< t.STID > | STUD(t) \wedge \forall (r | (SCR(r) \wedge r.STID =' 1024') \Rightarrow \exists (s | (SCR(s) \wedge s.STID = t.STID \wedge r.COID = s.COID)\}$$

STUD

STID	STNAME	STLEV	STMJR
888	st8	ms	math
444	st4	ms	phys
⋮	⋮	⋮	⋮

SCR

STID	COID	TR	YR	GRADE
888	co2	1	87	19
888	co3	1	87	10
444	co2	1	87	13
⋮	⋮	⋮	⋮	⋮

Example

- IDs of students whose grade in at least one course is greater than from all the grades of the student with STID = '1024'.

$$\{< t.STID > | SCR(t) \wedge \forall (r | (SCR(r) \wedge r.STID =' 1024') \Rightarrow t.GRADE = r.GRADE\}$$

$$P_1 \Rightarrow P_2 \equiv \neg P_1 \vee P_2$$

STUD

STID	STNAME	STLEV	STMJR
888	st8	ms	math
444	st4	ms	phys
⋮	⋮	⋮	⋮

SCR

STID	COID	TR	YR	GRADE
888	co2	1	87	19
888	co3	1	87	10
444	co2	1	87	13
⋮	⋮	⋮	⋮	⋮

Example

- IDs of students who has the maximum in all taken courses:

$$\{< t.STID > | STUD(t) \wedge \forall (r | SCR(r) \wedge t.STID = r.STID \Rightarrow \forall (s | SCR(s) \wedge r.COID = s.COID \wedge r.TR = s.TR \wedge s.YR = r.YR) \Rightarrow r.GRADE \geq s.GRADE\}$$

STUD

STID	STNAME	STLEV	STMJR
888	st8	ms	math
444	st4	ms	phys
⋮	⋮	⋮	⋮

SCR

STID	COID	TR	YR	GRADE
888	co2	1	87	19
888	co3	1	87	10
444	co2	1	87	13
⋮	⋮	⋮	⋮	⋮

03

Domain Relational Calculus

Domain Relational Calculus

- **Query** has the form:

$$\{< x_1, x_2, ..., x_n > \mid p(< x_1, x_2, ..., x_n >)\}$$

- 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 $< x_1, x_2, ..., x_n >$ that make the formula $p(< x_1, x_2, ..., x_n >)$ true.
- Last Example with DRC:

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

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

Free and Bound Variables

- **The use of quantifiers ∃X and ∀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 | 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.

04

Quantifiers

Review

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

		b	
		T	F
a	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.**

$$P_1 \wedge P_2 \equiv \neg(\neg P_1 \vee \neg P_2)$$

$$\forall t \in r(P_1(t)) \equiv \neg \exists t \in r(\neg(P_1(t)))$$

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))

05

More Practice

TRC

- 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 **WHERE** STX.STDEID=‘D11’   STID of students in department ‘D11’
- (STX.STID, STX.STL) **WHERE EXISTS** STCOX (STX.STID=STCOX.STID **AND** STCOX.COID=‘COM11’) STID and STL of who has enrolled in course =‘COM11’

Another Relational Calculus Notation

S(S#, SNAME, CITY, STATUS,…)
P(P#, PNAME, COLOR,…)
SP(S#, P#, QTY)

- Supplier numbers of all suppliers: SX.S#
- Names of suppliers in city **C2** whose status is greater than **15**.
  - SX.SNAME **WHERE** SX.CITY=‘C2’ **AND** SX.STATUS> 15

- Names of suppliers who have supplied at least one blue part.
  - SX.SNAME **WHERE EXISTS** SPX (SPX.S#=SX.S# **AND** **EXISTS** PX (PX.P#=SPX.P# **AND** PX.COLOR=‘Blue’))

- Names of supplier pairs who are from the same city and have produced at least one common part.
  - 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

STT(STID, NAME, DEID, …)
COT(COID, TITLE, CREDIT, …)
STCOT(STID, COID, YR, TR, GRADE, …)

- Give the titles of the courses in which all computer engineering students passed in the second semester of the academic year 98-99.
- If a computer major student, then passed in second semester of 98-99: $P \Rightarrow Q \equiv \neg P \vee Q$
  - 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))