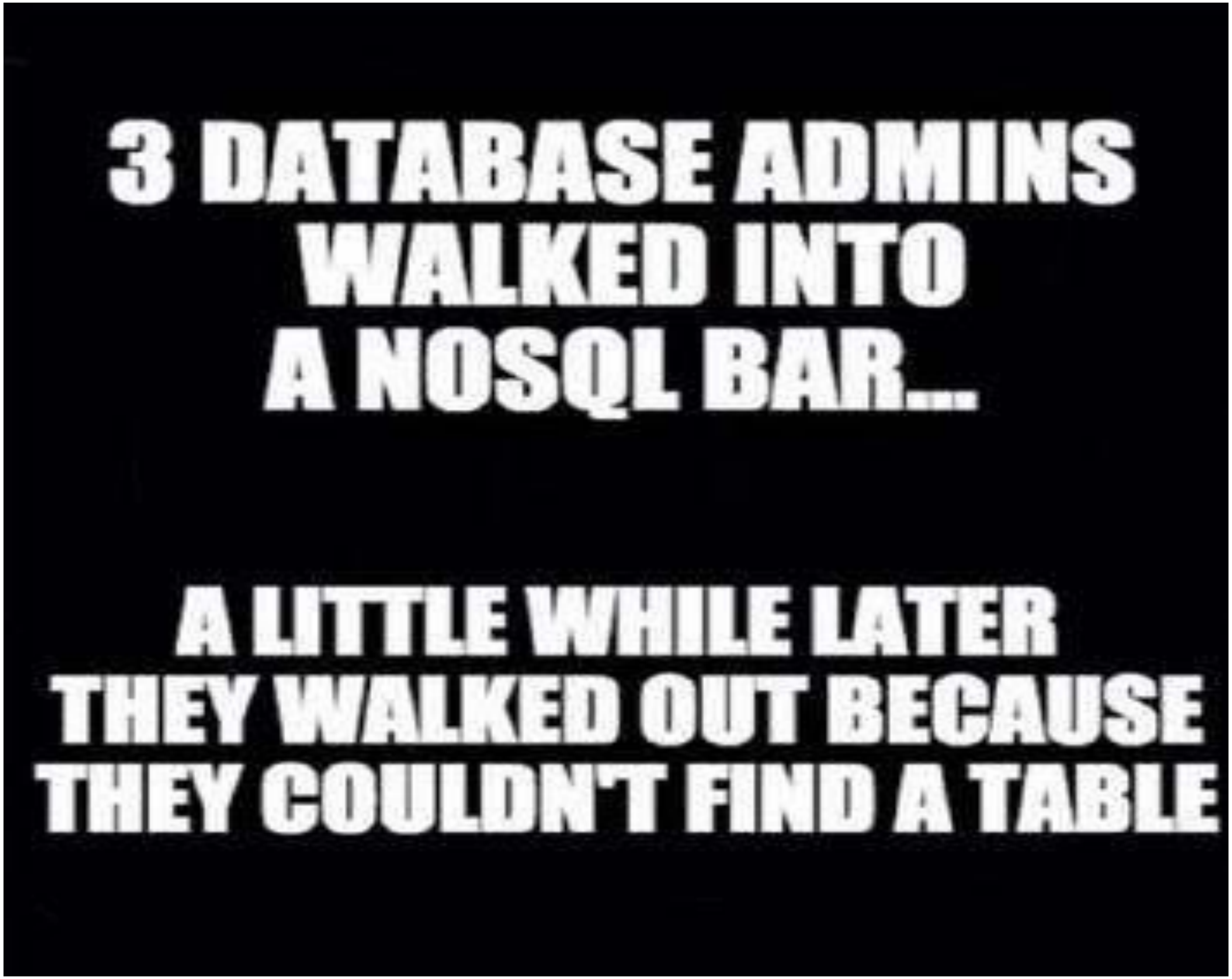
Formal Relational Query Languages

Instructor : Nitesh Kumar Jha

niteshjha@soa.ac.in

ITER,S'O'A(DEEMED TO BE UNIVERSITY)

Sept 2018


3 DATABASE ADMINS
WALKED INTO
A NOSQL BAR...
A LITTLE WHILE LATER
THEY WALKED OUT BECAUSE
THEY COULDN'T FIND A TABLE

Formal Relational query language

- Relational Algebra
- Relational Calculus
  - Tuple Relational Calculus
  - Domain Relational Calculus

Relational Algebra

- Procedural formal query language
- It forms a basis of mathematical foundation on which SQL is developed
- The operators take one or two relations as inputs and produce a new relation as a result.

Relational Algebra

- Six basic operators
  - select: σ --------Unary Operator
  - project: $\prod$ ---------Unary Operator
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ ----------Unary operator

Binary Operator

- Other operators include
  - Natural Join : $\bowtie$
  - Left Outer Join : ⟕
  - Right Outer Join : ⟖
  - Full Outer Join : ⟗

Which I have discussed in my Lab lecture on Subquery and joins.

More operators include Division(÷) and Assignment($\leftarrow$)

Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P is a predicate on attributes in E_1
  - $\prod_S(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x (E_1)$, x is the new name for the result of E_1

Two Databases

Consider the following tables for Bank Database:

- Customer(C_id, name, ph_no, address)
- Depositor(C_id, A_no)
- Account(A_no, Balance, B_name)
- Borrower(C_id, l_no)
- Loan(l_no, l_amt, B_name)
- Branch(B_name, B_city, B_assets)

Consider the following table for University Database:

- Instructor(<u>ID</u>, Name, Dept_name, Salary)
- Course(<u>Course_id</u>, Title, Dept_name, credits)
- Prereq(<u>Course_id</u>, <u>Prereq_id</u>)
- Department(<u>Dept_name</u>, Building, Budget)
- Teaches(<u>ID</u>, <u>Course_id</u>, <u>sec_id</u>, <u>sem</u>, <u>year</u>)
- Student(<u>id</u>, Name, dept_name, tot_credit)
- Takes(<u>id</u>, <u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, grade)
- Section(<u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, building, room_no, timeslot_id)

Select Operation

- Notation: $\sigma_p(r)$
- *p* is called the **selection predicate**
- Defined as:

$$\sigma_p(\mathbf{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Where *p* is a formula in propositional calculus consisting of **terms** connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
Each **term** is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: $=, \neq, >, \geq. <. \leq$

- Example of selection:

$$\sigma_{dept\_name=\text{“Physics”}}(instructor)$$

Project Operation

- Notation:

$$\Pi_{A_1, A_2, \ldots, A_k}(r)$$

  where A_1, A_2 are attribute names and r is a relation name.
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the *dept_name* attribute of *instructor*

$$\Pi_{ID,\ name,\ salary}\ (instructor)$$

Union Operation

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid.
  1. *r, s* must have the *same* **arity** (same number of attributes)
  2. The attribute domains must be **compatible** (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)
- Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

$$\Pi_{course\_id}(\sigma_{semester=\text{“Fall”}\ \wedge\ year=2009}(Section)) \cup$$

$$\Pi_{course\_id}(\sigma_{semester=\text{“Spring”}\ \wedge\ year=2010}(Section))$$

Set Difference Operation

- Notation $r - s$
- Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- Set differences must be taken between **compatible** relations.
  - r and s must have the same arity
  - attribute domains of r and s must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\Pi_{course\_id}\,(\sigma_{semester=\text{“Fall”}\ \land\ year=2009}\,(Section))\ -$$
$$\Pi_{course\_id}\,(\sigma_{semester=\text{“Spring”}\ \land\ year=2010}\,(Section))$$

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
  - *r*, *s* have the *same arity*
  - attributes of *r* and *s* are compatible
- Note: $r \cap s = r - (r - s)$

Cartesian-Product Operation

- Notation $r \text{ x } s$
- Defined as:

$$r \text{ x } s = \{t\,q \mid t \in r \textbf{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes of *r(R)* and *s(S)* are not disjoint, then renaming must be used.

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

$$\rho_X(E)$$

returns the expression E under the name X

- If a relational-algebra expression E has arity n, then

$$\rho_{x(A_1, A_2, \ldots, A_n)}(E)$$

returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

More Operators:Assignment Operation

- The assignment operation (←) provides a convenient way to express **complex queries**.

  - **Write query as a sequential program** consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.

  - Assignment **must always** be made to a temporary relation variable.
  - Iter_Faculty←$\prod_{ID,Name}$(Instructor)

Division Operation

- Suited to queries that include the phrase **"for all"**.
- Let *r* and *s* be relations on schemas *R* and *S* respectively $r \div s$

  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  The result of $r \div s$ is a relation on schema

  $R - S = (A_1, \ldots, A_m)$

$$r \div s = \{\, t \mid t \in \Pi_{R\text{-}S}(r) \wedge \forall\, \mathbf{u} \in \mathbf{s}\, (\mathbf{tu} \in \mathbf{r}) \,\}$$

* ***u*** *representing any tuple in s*

Where *tu* means the concatenation of a tuple *t* and *u* to produce a single tuple

* *for every tuple in R-S (called t), there are a set of tuples in R, such that for all tuples (such as u) in s, the tu is a tuple in R.*

Division Operation – Example

- Relations *r*, *s*:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
$\in$	6
$\in$	1
β	2

r

B
1
2

s

- $r \div s$:

A
α
β

e.g.

A is customer name

B is branch-name

1and 2 here show two specific branch-names

(Find customers who have an account in all branches of the bank)

Another Division Example

- Relations *r*, *s*:

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

- *r* ÷ *s*:

A	*B*	*C*
α	a	γ
γ	a	γ

e.g.
Students who have taken both "a" and "b" courses, with instructor "1"

(Find students who have taken all courses given by instructor 1)

Assignment Operation

- Example of writing division with set difference, projection, and assignments: r ÷ s

$$temp1 \leftarrow \prod_{R-S} (r)$$

$$temp2 \leftarrow \prod_{R-S} ((temp1 \times s) - \prod_{R-S,S}(r))$$

$$result = temp1 - temp2$$

- The result to the right of the $\leftarrow$ is assigned to relation variable on the left of the $\leftarrow$.
- May use variables in subsequent expressions

** Try executing the above query at home on the previous example, to convince yourself about its equivalence to the division operation*

Tuple Relational Calculus

Tuple Relational Calculus

- A nonprocedural query language, where each query is of the form

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

- It is the set of all tuples *t* such that predicate *P* is true for *t*
- *t* is a *tuple variable*, $t[A]$ denotes the value of tuple *t* on attribute *A*
- $t \in r$ denotes that tuple *t* is in relation *r*
- *P* is a *formula* similar to that of the predicate calculus

Predicate Calculus Formula

1. Set of attributes and constants
2. Set of comparison operators: (e.g., $<$, $\leq$, $=$, $\neq$, $>$, $\geq$)
3. Set of connectives: and ($\wedge$), or (v), not ($\neg$)
4. Implication ($\Rightarrow$): x $\Rightarrow$ y, if x if true, then y is true

$$x \Rightarrow y \equiv \neg x \vee y$$

5. Set of quantifiers:
   - $\exists\, t \in r\, (Q\,(t\,)) \equiv$ "there exists" a tuple in *t* in relation *r* such that predicate *Q* (*t*) is true
   - $\forall t \in r\, (Q\,(t\,)) \equiv Q$ is true "for all" tuples *t* in relation *r*

Example Queries

- Find the *ID, name, dept_name, salary* for instructors whose salary is greater than $80,000

$$\{t \mid t \in instructor \land t\,[salary\,] > 80000\}$$

Notice that a relation on schema (*ID, name, dept_name, salary*) is implicitly defined by the query

- As in the previous query, but output only the *ID* attribute value

$$\{t \mid \exists\, s \in \text{instructor}\ (t\,[ID\,] = s\,[ID\,] \land s\,[salary\,] > 80000)\}$$

Notice that a relation on schema (*ID*) is implicitly defined by the query

Example Queries

- Find the names of all instructors whose department is in the Watson building

$$\{t \mid \exists s \in instructor\ (t\,[name\,] = s\,[name\,] \\ \land \exists u \in department\ (u\,[dept\_name\,] = s[dept\_name] \text{ “} \\ \land\ u\,[building] = \text{“Watson”}\ ))\}$$

- Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

$$\{t \mid \exists s \in teaches\ (t\,[course\_id\,] = s\,[course\_id\,] \land \\ s\,[semester] = \text{“Fall”} \land s\,[\text{year}] = 2009 \\ \text{v } \exists u \in teaches\ (t\ \ [course\_id\,] = u\,[course\_id\,] \land \\ u\,[semester] = \text{“Spring”} \land u\,[\text{year}] = 2010\ )\}$$

Example Queries

- Find the set of all courses taught in the Fall 2009 semester, and in the Spring 2010 semester

$$\{t \mid \exists s \in \textit{teaches}\ (t\,[\textit{course\_id}\,] = s\,[\textit{course\_id}\,] \wedge s\,[\textit{semester}] = \text{“Fall”} \wedge s\,[\text{year}] = 2009$$
$$\wedge\ \exists u \in \textit{teaches}\ (t\ \ [\textit{course\_id}\,] = u\,[\textit{course\_id}\,] \wedge u\,[\textit{semester}] = \text{“Spring”} \wedge u\,[\text{year}] = 2010\ )\}$$

- Find the set of all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\{t \mid \exists s \in \textit{teaches}\ (t\,[\textit{course\_id}\,] = s\,[\textit{course\_id}\,] \wedge s\,[\textit{semester}] = \text{“Fall”} \wedge s\,[\text{year}] = 2009$$
$$\wedge \neg\, \exists u \in \textit{teaches}\ (t\ \ [\textit{course\_id}\,] = u\,[\textit{course\_id}\,] \wedge u\,[\textit{semester}] = \text{“Spring”} \wedge u\,[\text{year}] = 2010\ )\}$$

Universal Quantification

- Find all students who have taken all courses offered in the Biology department
  - $\{t \mid \exists\, r \in student\ (t\,[ID] = r\,[ID]) \wedge$
    $(\forall\, u \in course\ (u\,[dept\_name] = \text{“Biology”} \Rightarrow$
    $\exists\, s \in takes\ (t\,[ID] = s\,[ID\,] \wedge$
    $s\,[course\_id] = u\,[course\_id]))\}$

Safety of Expressions

- It is possible to write tuple calculus expressions that generate infinite relations.
- For example, $\{ t \mid \neg t \in r \}$ results in an infinite relation if the domain of any attribute of relation *r* is infinite
- To guard against the problem, we restrict the set of allowable expressions to safe expressions.
- An expression $\{t \mid P(t)\}$ in the tuple relational calculus is *safe* if every component of *t* appears in one of the relations, tuples, or constants that appear in *P*
  - NOTE: this is more than just a syntax condition.
    - E.g. $\{ t \mid t[A] = 5 \vee \textbf{true} \}$ is not safe --- it defines an infinite set with attribute values that do not appear in any relation or tuples or constants in *P*.

Safety of Expressions (Cont.)

- Consider again that query to find all students who have taken all courses offered in the Biology department
  - $\{t \mid \exists\ r \in student\ (t[ID] = r[ID]) \wedge$
    $(\forall\ u \in course\ (u[dept\_name] = \text{“Biology”} \Rightarrow$
    $\exists\ s \in takes\ (t[ID] = s[ID] \wedge$
    $s[course\_id] = u[course\_id]))\}$
- Without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.

Domain Relational Calculus

Domain Relational Calculus

- A nonprocedural query language equivalent in power to the tuple relational calculus
- Each query is an expression of the form:

$$\{ < x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n)\}$$

  - $x_1, x_2, \ldots, x_n$ represent domain variables
  - P represents a formula similar to that of the predicate calculus

Example Queries

- Find the *ID, name, dept_name, salary* for instructors whose salary is greater than $80,000
  - $\{< i, n, d, s> \mid < i, n, d, s> \in instructor \land s > 80000\}$
- As in the previous query, but output only the *ID* attribute value
  - $\{< i> \mid < i, n, d, s> \in instructor \land s > 80000\}$
- Find the names of all instructors whose department is in the Watson building

$$\{< n > \mid \exists\, i, d, s\, (< i, n, d, s > \in instructor \land \exists\, \text{b, a}\, (< d, b, a> \in department \land b = \text{“Watson”}))\}$$

Example Queries

- Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

$$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t> \in Teaches\ \wedge$$
$$s = \text{“Fall”} \wedge y = 2009\ )$$
$$\text{v}\ \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t> \in Teaches\ ] \wedge$$
$$s = \text{“Spring”} \wedge y = 2010)\}$$

This case can also be written as

$$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t> \in Teaches\ \wedge$$
$$(\ (s = \text{“Fall”} \wedge y = 2009\ )\ \ \text{v}\ (s = \text{“Spring”} \wedge y = 2010))\}$$

- Find the set of all courses taught in the Fall 2009 semester, and in the Spring 2010 semester

$$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t> \in Teaches\ \wedge$$
$$s = \text{“Fall”} \wedge y = 2009\ )$$
$$\wedge \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t> \in Teaches\ ] \wedge$$
$$s = \text{“Spring”} \wedge y = 2010)\}$$

Safety of Expressions

The expression:

$$\{ < x_1, x_2, \ldots, x_n > \mid P\ (x_1, x_2, \ldots, x_n)\}$$

is safe if all of the following hold:

1. All values that appear in tuples of the expression are values from *dom* (P) (that is, the values appear either in P or in a tuple of a relation mentioned in P).
2. For every "there exists" subformula of the form $\exists\, x\ (P_1(x))$, the subformula is true if and only if there is a value of x in *dom* (P_1) such that $P_1(x)$ is true.
3. For every "for all" subformula of the form $\forall_x\ (P_1\ (x))$, the subformula is true if and only if $P_1(x)$ is true for all values x from *dom* (P_1).

Universal Quantification

- Find all students who have taken all courses offered in the Biology department
  - $\{< i > \mid \exists\ n, d, tc\ (< i, n, d, tc > \in student\ \wedge$
    $(\forall\ ci, ti, dn, cr\ (< ci, ti, dn, cr > \in course \wedge dn = \text{“Biology”}$
    $\Rightarrow\ \exists\ si, se, y, g\ (<i, ci, si, se, y, g> \in takes\ ))\}$
  - Note that without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.

End of Chapter 6