Database Design & Applications (SS ZG 518)

Dr.R.Gururaj
CS&IS Dept.

BITS Pilani
Hyderabad Campus

Relational Algebra & Relational Calculus (Ch. 6)

Content

- ❑ *Query languages & Formal query languages for Relational data model*
- ❑ *Introduction to Relational Algebra*
- ❑ *Relational operators*
- ❑ *Set operators*
- ❑ *Join operators*
- ❑ *Aggregate functions*
- ❑ *Grouping operator*
- ❑ *Relational Calculus concepts*

Division (÷)
Used when we want to check the meeting of all the criteria

Let R(A, B) and S(A) $T \leftarrow R \div S$

Selects all values for *B* column in R which contains all values under A in S.
Hence the no. column in T is only *B*.

Join: ($\bowtie$) Used to join tuples from different tables based on some condition. Result is new tuple with different *arity*.

$D \leftarrow DEPT \bowtie_{Mgrssn = ssn} EMP$

Joins tuples from DEPT & EMP where *Mgrssn* in DEPT is equal to *ssn* in EMP and stores the new tuples in relation D.

Theta Join: Joining on some condition with comparison that involves operator like (=, >, ≤, ≥, ≠) etc.

Equijoin is a special type of join where the join condition is '=' (equals operator) only.

Natural Join: is an equijoin on attributes in R and S having the same name.

In the resulting relation only one column is listed.

Ex. D ← DEPT *EMP.

The joining is on common attribute with same name (*Dept Name*).

Employee				Dept			Employee * Dept			
Name	**EmpID**	**Dept Name**		**Dept Name**	**Manager**		**Name**	**EmpID**	**Dept Name**	**Manager**
Harry	3415	Finance		Finance	George		Harry	3415	Finance	George
Sally	2241	Sales		Sales	Harriet		Sally	2241	Sales	Harriet
George	3401	Finance					George	3401	Finance	George
Harriet	2202	Sales		Production	Charles		Hariet	2202	Sales	Harriet

Inner Join (R ⋈ S) An inner join only combines tuples from R and S if they meet the conditions. Tuples that do not meet the conditions are not showed in the final result. (This is the usual type of join).

Outer join: An outer join displays the tuples of one of the relations even if there is no match for the tuple in the other relation.

Left outer join: (R ⟕ S) In the result relations, in addition to all the matching tuples from R and S, all the remaining tuples from left side relation (R) are also showed. For these tuple from R, columns under S will have null values(padding).

Right Outer Join: (R ⟖ S) In the result relation, matching tuples will occur from R & S. In addition all the tuples form S will also appear with null values for the R attributes.

Full Outer join (R ⟗ S) In the result, all tuples from R & S will appear with null values for the other relations attributes.

Additional Relational Operations

Aggregate functions: Sum Average Max Min Count

Grouping:

The tuples of a relation are first grouped by the value of some attribute and then aggregate functions are applied on individual groups.

Symbol used is – £

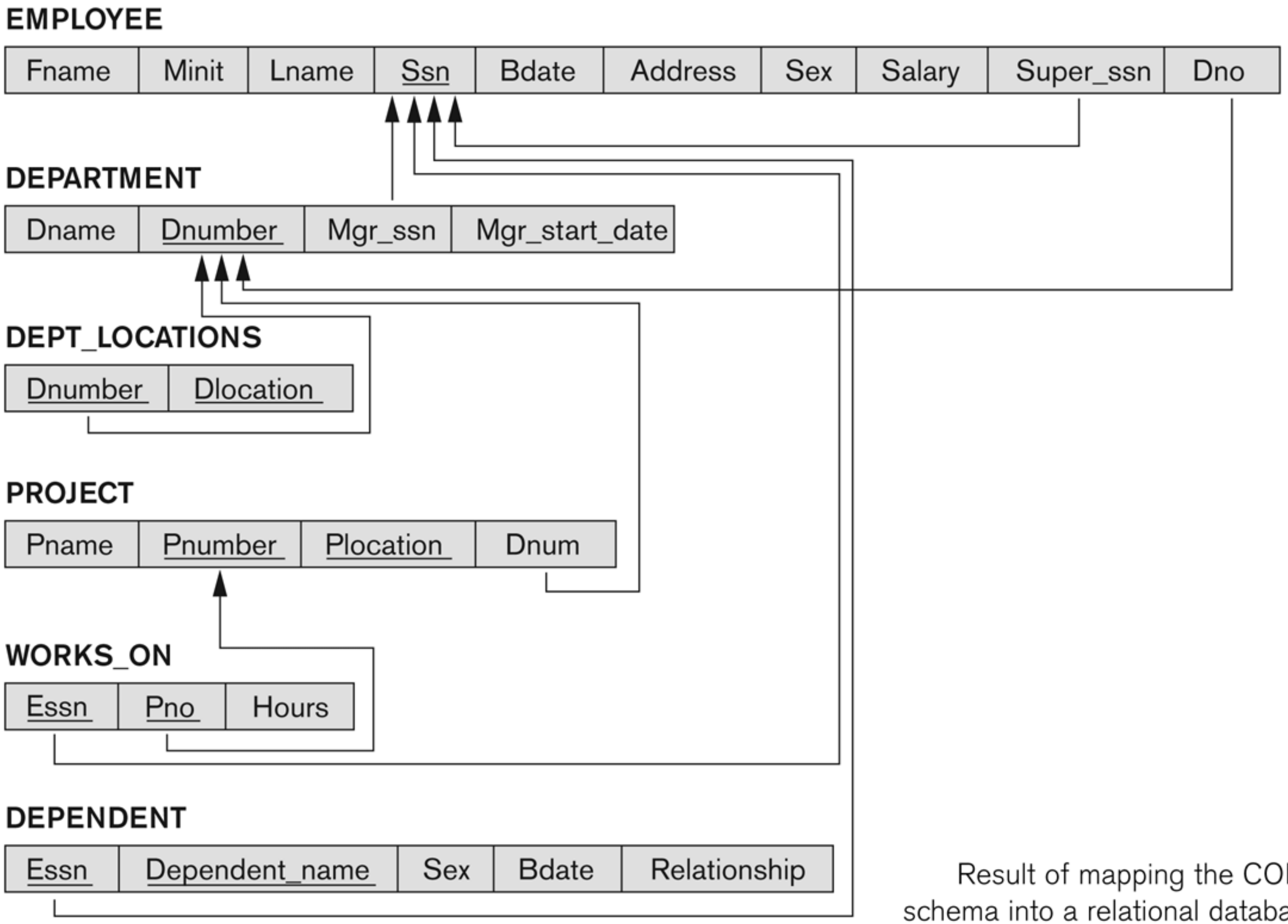
Ex. $Dno\ \pounds_{Count\ (ssn)}\ (EMP)$

The above expression first group the tuples in EMP table based on Dno, and then applies count function on individual groups this will output no. of employees in each department.

Result relation →

Dno	Count (SSn)

Company Database Schema (set of tables/relations)



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 7.2
Result of mapping the COMPANY ER schema into a relational database schema.

1. Get the list of employee IDs who have no dependents.

It is equivalent to:
{ {set of all employees} - {set of employees with Dependents} }

R1 ← Π ssn (Employee)
R2 ← Π essn (Dependent)
Result ← R1- R2

2. Get the list of employee IDs who have more than two dependents.

$R1 \leftarrow essn\ \text{£}_{COUNT\ Dependent\_name}\ (Dependent)$

$Result \leftarrow \Pi_{essn}\ (\sigma_{COUNT\_Depenedent\_name>2}\ (R1))$

R1

essn	Count_Dependent_name
101	3
102	1

3. *Get the list of projects controlled by department with name “ACCOUNTS”.*

$R1 \leftarrow \sigma_{Dname=\text{‘ACCOUNTS’}} (Department)$

$Result \leftarrow \Pi_{pnumber,\ pname} (Project \bowtie_{Dnum=\text{‘}Dnumber} (R1))$

4. Get the list of employee IDs working on all projects

$R1 \leftarrow \Pi_{essn,\ pno}(Works\_on)$

$R2 \leftarrow \Pi_{pnumber}(Project)$

$Result \quad \leftarrow R1 \div R2$

A	B
a1	b1
a1	b2
a1	b3
a2	b2
a2	b3
a3	b1
a3	b2
a3	b3
a4	b3

B
b1
b2
b3

A
a1
a4

5. Find the projects controlled by departments located in Mumbai.

$R1 \leftarrow \Pi_{Dnumber}(\sigma_{\ Dlocation=\text{‘}Mumbai\text{’}}\ (Dept\_locations)$

$R2 \leftarrow \Pi_{\ pnumber,\ pname}\ (Project \bowtie_{Dnum=\text{‘}Dnumber}\ (R1))$

Tuple Relational Calculus

Relational Calculus is a formal query language for relational model where we write one declarative expression to specify a retrieval request and hence there is no description of how to evaluate the query.

A calculus expression specifies what is to be retrieved rather than how to do it.

Hence, relational calculus is *non–procedural* language where as *relational algebra* discussed in the previous section is procedural, where we write sequence of operations to retrieve data.

Any expression for data retrieval written in relational algebra can also be written in relational calculus and vice-versa.

Hence expression power of relational algebra and relational calculus is same.

<u>Tuple Rational Calculus(TRC)</u> is based on specifying a number of *tuple variables*.

Each tuple variable usually ranges over a particular database relation. Variables can take values of individual tuples from the relation.
A simple relational calculus query is in the form-

{t | condition (t)}

t – tuple variable

condition (t) – is a conditional expression involving *t*.

Result is a set of all tuples that satisfy the conditions specified in <u>*condition (t)*</u>.

Ex. Find all employees whose salary is above 50,000

$\{t \mid \text{EMP}(t) \text{ and } t.\text{salary} > 50{,}000\}$

Selects all tuples from EMP such that for each tuple selected, the salary value is > 50,000.

The expression EMP(*t*) specifies from where the tuple *t* must be chosen.

Hence EMP relation in this case is known as a *range relation*.

Note: The above query retrieves all the attributes of relation EMP.

The *universal* (∀), and *existential* (∃) quantifiers can be applied to tuples.

Ex.:

{t.name, t age | EMP (t) and (∃d) (Dept (d)and d.dname = 'Research' and d.dno = t.dno)}

To retrieve the name and age of all employees who work for 'Research' department.

If the tuple variable *t* occurs with ∃ or ∀ quantifiers the variable is known as *bound variable and* otherwise called as *free variable.*

Safe Relation Calculus Expression

Is one that guarantees to yield a finite set of tuples as result.

Ex. *{t | not (EMP (t))}*

Is *unsafe* because it yields all tuples in the universe that are not in EMP relation, which are infinitely numerous.

An expression is safe if all values in its result are from the domain of the expression.

Relational Completeness:

This notion is used to compare high level query languages.

Any relational query language L is considered to be *relationally complete* if we can express in L any query that is expressed in relational algebra (RA) or relational calculus (RC).

Exercise

Student (sid, sname, city, cgpa, branch)
Course(cid, cname, credits)
Student_Course(sid, cid, grade)

1. *Retrieve all course IDs registered by ‘Gopi’ .*
2. *Get the course ids for those registered by all students of CSE branch.*
3. *Get student Id, sname for those who have registered for all courses.*
4. *Get student ID and name for those who are either from Delhi or belong to ‘Civil’ branch.*
5. *Get student ID and name for those who are from Mumbai and do not belong to ‘Civil’ branch.*
6. *Get the course Id and number of Registrations.*

Summary

✓ *What is a query language*

✓ *Formal query languages for Relational data model*

✓ *Basic concepts of Relational Algebra*

✓ *Operations in Relational Algebra*

✓ *Examples and Exercise*