

Johns Hopkins Engineering

Principles of Database Systems

Module 5 / Lecture 1

The Relational Data Model and Relational Algebra



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Relational Model Concept

- The model was first proposed by Dr. E.F. Codd of IBM in 1970 in the following paper - "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970.
- The Relational Model of Data is based on the concept of a Relation.
- A database is represented as a collection of relations by the relational model.

Relational Model Concept (cont.)

- A relation is a table of values (a set of tuples).
- A relation is a mathematical concept based on the ideas of sets.
- Each row (a tuple) in the table represents a collection of related data values that can be interpreted for either an entity or a relationship.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.

The Formal and Informal Relational Terms

Formal Relational Term	Informal Equivalents
Relation	Table
Schema of Relation	Table Definition
Tuple	Row or Record
Cardinality	Number of Rows
Attribute	Column or Field Header
Degree	Number of Columns
Primary Key (PK)	Unique Identifier (UID)
Domain	Pool of Legal Value
Extension	Populated Table

The Formal and Informal Relational Terms (cont.)

- Textbooks present the model and operations on the relational database using the formal terms.
- Programming languages (e.g., SQL) and commercial DBMS world use the informal terms of TABLE, COLUMN, and ROW.

The Formal and Informal Relational Terms (cont.)

- A relation schema R consists of a relational name and a list of attributes.
- Each attribute has its own domain. A domain defines the data type and valid values of an attribute. However, several attributes may have the same domain.

The Formal and Informal Relational Terms (cont.)

- A relation intension represents the ingredient of the relation (a set of attributes.) It does not change very often.
- A relation extension reflects the state of the relation. It may change very often.

Relational Example

■ A relation **SUPPLIER** (a relation name)

<u>s_id</u>	s_name	status	location	phone
1	Queen	Active	North	301-798-2000
2	King	Active	East	410-531-7777
3	Duke	Active	South	202-567-1234
4	Knight	Inactive	West	240-820-7600

Where:

- s_id is primary key (PK);
- the set of headers of all columns: i.e., s_id, s_name, status, location, and phone are attributes;
- the set of values, i.e. (1, Queen, Active, East, 301-798-2000) is a tuple;
- Number of rows (4) is called cardinality;
- legal values for each column are called domain.

The Schema of A Relation

- The Schema of A Relation:
 - $R(A_1, A_2, \dots, A_n)$
 - Relation R is defined over attributes A_1, A_2, \dots, A_n .
 - SUPPLIER is a relation defined over the five attributes s_id, s_name, status, location, phone.
 - Each value of an attribute is derived from an appropriate domain. For example, the domain of status includes 'active' or 'inactive' and s_name is defined over the domain of variable length of string with a maximum of 30 characters.

The Schema of A Relation (cont.)

■ The **Schema** of A Relation:

- Attributes in a relation are also called as columns of the table.
- A **tuple** (row) is an ordered set of values.
- An attribute of a relation is defined as “RelationName.AttributeName” (e.g., student.ssn) that is unique in the relation.

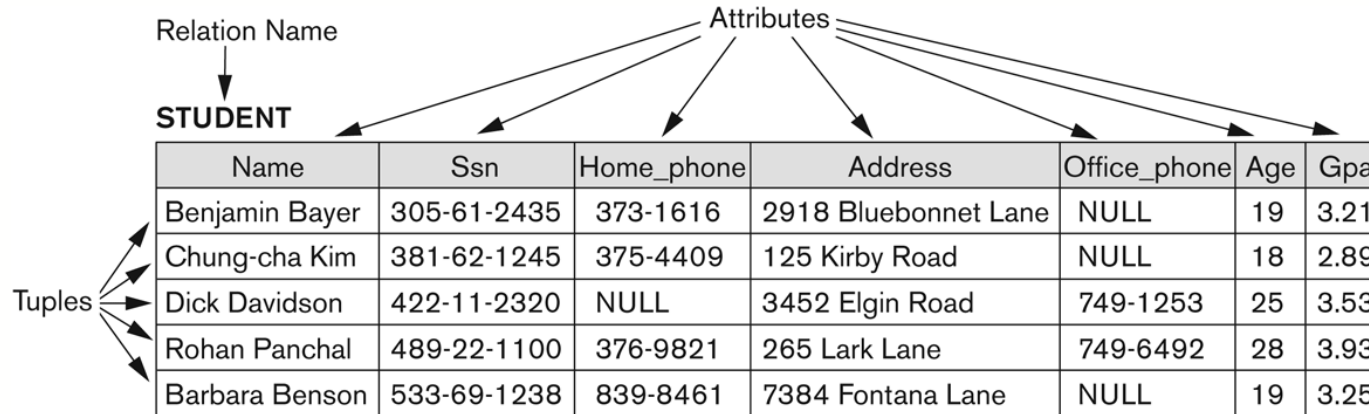


Figure 5.1

The attributes and tuples of a relation STUDENT.

The Schema of A Relation (cont.)

- A relation may include a set of tuples.
- Tuples do not have order, even though they (e.g., employee_id) may have special order in a file at a physical storage or they may be presented in a logical level.
- At a logical level, the order of attributes and their values are not important.
- For performance, the order of attributes may be important regarding the physical storage.

Relational Example (cont.)

- All values in a tuple are considered *atomic* (indivisible).
 - Composite attributes
 - Multivalued attributes
- A value of NULL for an attribute in a tuple represents:
 - No value specified, an unknown value, or may not apply.

Johns Hopkins Engineering

Principles of Database Systems

Module 5 / Lecture 2

The Relational Data Model and Relational Algebra



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

The Relational Constraints

- Constraints are conditions that must hold on all valid relation instances
- Types of constraints:
 - Domain Constraints
 - Key Constraints and Constraints on Null
 - Entity Integrity Constraints
 - Referential Integrity Constraints

Domain Constraints

- A constraint includes data types and ranges associated with a domain.
 - The most fundamental integrity constraints applied to data in a database.
 - Data quality starts with proper data type.

Examples:

Month: 1 to 12

Year: Four digits

DOB: Date or MM/DD/YYYY

Vehicle Identification Number (VIN): 17 characters (digits and capital letters)

Last Name: Char(30) or Varchar(30)

Key Constraints

- A key constraint applies to the field with a unique identifier for each tuple. This field is called the key field. No two tuples can have the same value(s) of the key field(s).

Example: s_id (Supplier ID) in the SUPPLIER relation

- Superkey of a relation is a set of attributes SK such that no two tuples in any valid relation instance will have the same value for SK.
- A key constraint is a minimal set of superkey by removing all non-key attributes.

Key Constraints and Constraints on Null

■ Key Constraints

- If more than one key field in a relation, each of the keys is called candidate key, i.e. ssn, emp_id in the EMP relation.
- If a relation has several **candidate keys**, one can be chosen arbitrarily to be the **primary key**.

■ Constraints on Null

- Constraint on an attribute that may or may not accept a Null value.
(e.g., NOT NULL for required key attributes, FKs with total constraints)

Entity Integrity Constraint

- No primary key value can be null.
- A tuple cannot be identified if the primary key is null.

Note: Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity Constraint

- A constraint is specified between two relations and is used to maintain the consistency among tuples of the two relations.
- The primary key of a parent relation migrates to a child relation as a foreign key to maintain the referential integrity between two relations.
- Tuples in the *referencing relation* have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation*.

Constraints in Practice

- All constraints are defined in a relational database schema.
- In a relational database, the data definition language (DDL) specifies all constraints.
- DBMSs maintain and enforce these constraints for all relations and all tuples of constraint attributes.

Other Ways to Enforce Constraints or Business Rules

■ Triggers

- Code stored in the database and invoked by events that occur in the application
- Before/after insert/update/delete per row/table

■ Stored Procedures or Functions

- Code stored in the database and externally invoked by application code

■ Packages

- A collection of procedures, functions, variables, and SQL statements that are grouped together and stored as a single program unit

Relational Database Schema Example: COMPANY

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Figure 5.5 Schema diagram for the COMPANY relational database schema.

One Possible Database State for COMPANY Relational Database Schema

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Figure 5.6 One possible database state for the COMPANY relational database schema.

Relational Database Schema Example: COMPANY (cont.)

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

<u>Pname</u>	<u>Pnumber</u>	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Figure 5.6 (continued) One possible database state for the COMPANY relational database schema.

Displayed Referential Integrity Constraints

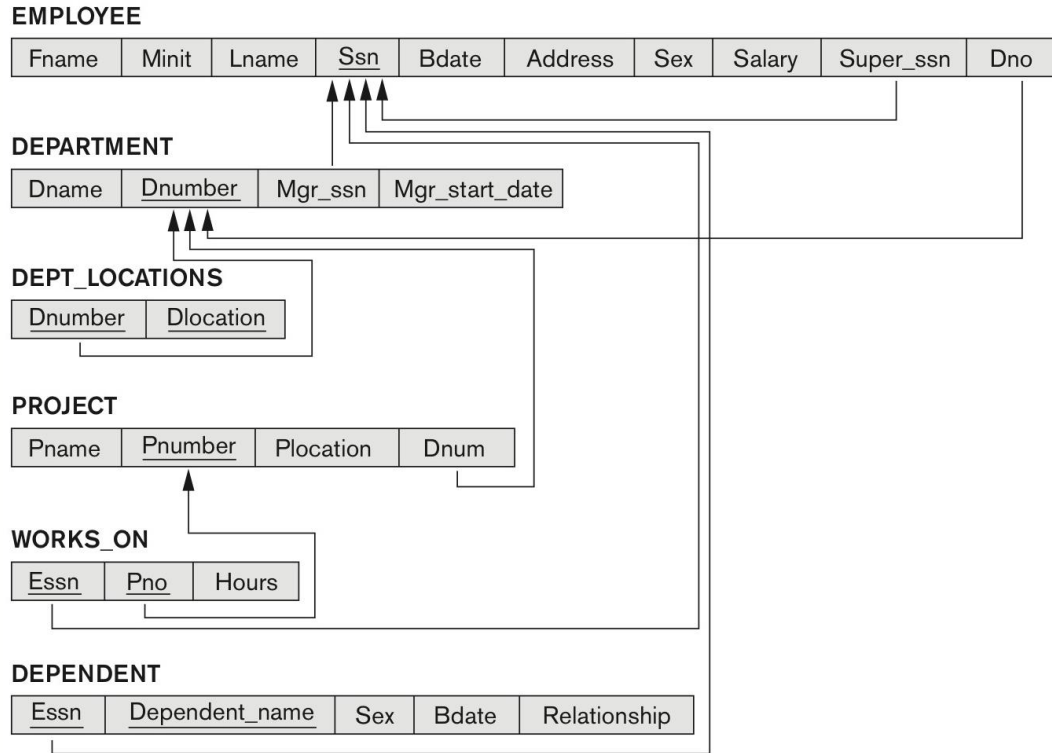


Figure 5.7 Referential integrity constraints displayed on the COMPANY relational database schema.

Operations on Relations

- Basic operations on relations include a data retrieval statement such as SELECT; and data manipulation statements such as INSERT, UPDATE, and DELETE.
- Data retrieval statements do not change the state of a database, data manipulation statements change the state of a database – always keep in a consistent state.

Operations on Relations (cont.)

- All data manipulation statements will be rejected if they violate the integrity constraints specified in the database schema implemented by data definition language.
- RDBMS enforces all constraints specified in the database schema.

Operations on Relations (cont.)

- Several update operations may have to be grouped together as a transaction and any integrity violation may result in all operations being rejected.
 - Possible actions for deleting a PK value while it has a FK referential integrity constraints:
 - Reject the operation(s) / Not allowed or RESTRICT
 - Cascade
 - Set Null
 - Set Default
 - How about updating a PK value?

Johns Hopkins Engineering

Principles of Database Systems

Module 5 / Lecture 3

The Relational Data Model and Relational Algebra



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

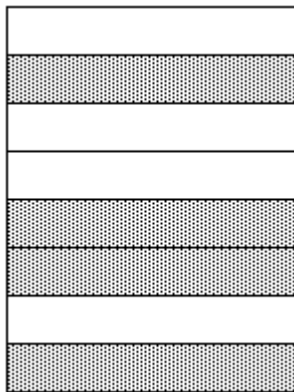
Relational Algebra

- The special relational operations are *select* (or *restrict*), *project*, *join*, and *divide*.
- The result of each operation on a relation(s) returns a new relation for further data manipulation if necessary.
- What is the algebra for?
 - In general, the expression serves as a high-level, symbolic representation of the user's intent to retrieve information from relations.
 - The algebra serves as a convenient basis for optimization based on commutative and associative properties for operators and operands.

Relational Operator SELECT

■ SELECT (σ)

- Returns a relation consisting of all tuples from a specified relation that satisfy a specified condition.



Relational Operator SELECT (cont.)

■ SELECT

- A select operation is used to choose a subset of the tuples in a relation that satisfy a selection condition.

$\sigma_{\langle \text{selection condition} \rangle} (\langle \text{relation name} \rangle)$

- The symbol σ is used to denote the SELECT operator, and selection condition is a Boolean expression specified on the relation attributes.

Relational Operator SELECT (cont.)

■ SELECT

- The selection condition is made of a number of clauses of the form.
 <attribute name> <comparison op> <constant value>, or
 <attribute name> <comparison op> <attribute name>
- The *comparison operator* can be (>, >=, =, <, <=). Clauses can be arbitrarily connected by the logical operators *AND*, *OR*, and *NOT* for form compound conditions.

Oracle provide additional {*BETWEEN ...AND ...*; *NOT BETWEEN ...AND ...*; *IN*; *NOT IN*; *LIKE*; *IS NULL*; *IS NOT NULL*}.

Relational Operator SELECT (cont.)

■ SELECT

- The SELECT operation is commutative.

$$\sigma_{\langle \text{cond1} \rangle} (\sigma_{\langle \text{cond2} \rangle} (R)) = \sigma_{\langle \text{cond2} \rangle} (\sigma_{\langle \text{cond1} \rangle} (R))$$

$$\sigma_{\langle \text{cond1} \rangle} (\sigma_{\langle \text{cond2} \rangle} (R)) = \sigma_{\langle \text{cond1} \rangle \text{ AND } \langle \text{cond2} \rangle} (R)$$

Example:

$$\sigma_{\text{DNO}=4} (\text{EMPLOYEE})$$

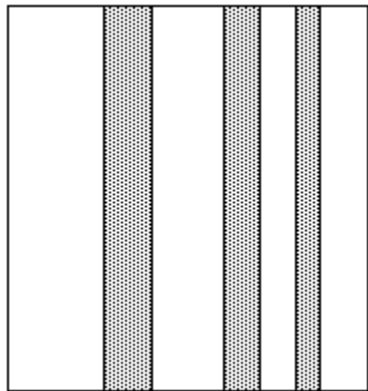
$$\sigma_{\text{SALARY}>30000} (\text{EMPLOYEE})$$

$$\sigma_{(\text{DNO}=4 \text{ AND } \text{SALARY}>25000) \text{ OR } \text{DNO}=5} (\text{EMPLOYEE})$$

Relational Operator PROJECT

■ PROJECT (π)

- Returns a relation consisting of specified attributes and eliminating duplicates as a vertical subset of the original relation.



Relational Operator Interpretation (cont.)

■ PROJECT

- A project operation is used to choose columns in a relation or keeps only certain columns.

$\pi_{\langle \text{attribute list} \rangle} (\langle \text{relation name} \rangle)$

- The resulting relation has only the attributes specified in $\langle \text{attribute list} \rangle$ and in the same order as they appear in the list.

Relational Operator Interpretation (cont.)

■ PROJECT

- The PROJECT operation *eliminates duplicate tuples* in the resulting relation so that it remains a mathematical set.
- The PROJECT is not commutative.

Example:

$\pi_{\text{FNAME,LNAME,SALARY}}(\text{EMPLOYEE})$

$\pi_{\text{ADDRESS}}(\text{EMPLOYEE})$

Relational Operator σ , π Interpretations

(a)

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

(b)

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

(c)

Sex	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

Figure 8.1 Results of SELECT and PROJECT operations. (a) $\sigma_{(Dno=4 \text{ AND } Salary>25000) \text{ OR } (Dno=5 \text{ AND } Salary>30000)}(EMPLOYEE)$. (b) $\pi_{Lname, Fname, Salary}(EMPLOYEE)$. (c) $\pi_{Sex, Salary}(EMPLOYEE)$.

Relational Operator σ , π Interpretations (cont.)

- A *relational algebra expression* (query) may consist of several operations.

Example: Retrieve the first names, last names and salaries of employees who work in department 4:

$\pi_{\text{FNAME,LNAME,SALARY}}(\sigma_{\text{DNO}=4}(\text{EMPLOYEE}))$

Alternatively, explicit intermediate relations for each step
(e.g., $\sigma_{\text{DNO}=4}(\text{EMPLOYEE})$)

Relational Operator σ , π Interpretations (cont.)

(a)

Fname	Lname	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

(b)

TEMP

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston,TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston,TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble,TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

R

First_name	Last_name	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

Figure 8.2 Results of a sequence of operations. (a) $\pi_{\text{Fname, Lname, Salary}}(\sigma_{\text{Dno}=5}(\text{EMPLOYEE}))$.
(b) Using intermediate relations and renaming of attributes.

Johns Hopkins Engineering

Principles of Database Systems

Module 5 / Lecture 4

The Relational Data Model and Relational Algebra

Relational Operator Set Operations

- Set Operations

UNION (\cup),

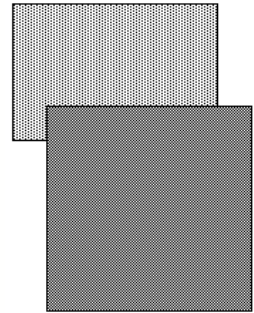
INTERSECTION (\cap),

DIFFERENCE ($-$), and

CARTESIAN PRODUCT (\times)

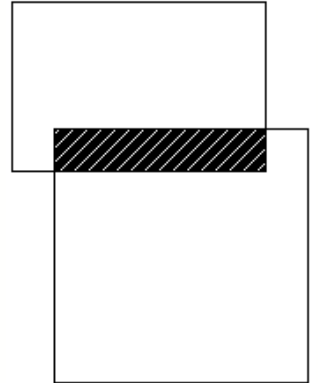
Relational Operator UNION

- UNION (\cup)
 - Returns a relation consisting of all tuples appearing in either or both of two specified relations.
 - Both relations are union compatible
 - have the same number of attributes
 - have same domains of each attributes
 - UNION is a commutative operation.
 - The UNION operation “ \cup ” eliminates the duplicate tuples.



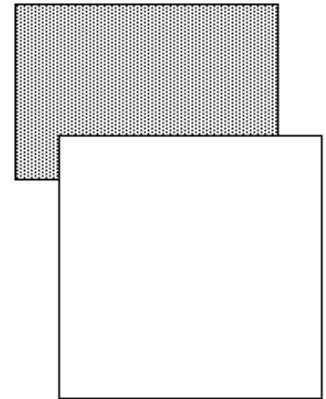
Relational Operator INTERSECT

- INTERSECT (\cap)
 - Returns a relation consisting of all tuples appearing in both of two specified relations.
 - The INTERSECTION operation “ \cap ” include all tuples that in both relations.
 - INTERSECTION is a commutative operation.



Relational Operator DIFFERENCE

- DIFFERENCE or MINUS ($-$)
 - Returns a relation consisting of all tuples appearing in the first and not the second of two specified relations.
 - DIFFERENCE operation is not a commutative operation.



Relational SET Operators Interpretation

(a) STUDENT		INSTRUCTOR		(b)	
Fn	Ln	Fname	Lname	Fn	Ln
Susan	Yao	John	Smith	Susan	Yao
Ramesh	Shah	Ricardo	Browne	Ramesh	Shah
Johnny	Kohler	Susan	Yao	Johnny	Kohler
Barbara	Jones	Francis	Johnson	Barbara	Jones
Amy	Ford	Ramesh	Shah	Amy	Ford
Jimmy	Wang			Jimmy	Wang
Ernest	Gilbert			Ernest	Gilbert

(c)		(d)		(e)	
Fn	Ln	Fn	Ln	Fname	Lname
Susan	Yao	Johnny	Kohler	John	Smith
Ramesh	Shah	Barbara	Jones	Ricardo	Browne
		Amy	Ford	Francis	Johnson
		Jimmy	Wang		
		Ernest	Gilbert		

Figure 8.4 The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT \cup INSTRUCTOR. (c) STUDENT \cap INSTRUCTOR. (d) STUDENT $-$ INSTRUCTOR. (e) INSTRUCTOR $-$ STUDENT.

Relational Operator CARTESIAN PRODUCT

- CARTESIAN PRODUCT (X)
 - Produces a relation that has all possible combinations of tuples of attributes of participated relations.
 - This operation “X” usually generates a big relation without much meaning. However, this operation works with the SELECT operation that can create a meaningful and desired relation.

R x S	
1	a
1	b
2	a
2	b
3	a
3	b

Relational Operator CARTESIAN PRODUCT (cont.)

FEMALE_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

EMPNames

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

Example: Retrieve a list of names of each female employee's dependents:

Steps 1 and 2:

$\text{FEMALE_EMPS} \leftarrow \sigma_{\text{Sex}='F'}(\text{EMPLOYEE})$

$\text{EMPNames} \leftarrow \pi_{\text{Fname}, \text{Lname}, \text{Ssn}}(\text{FEMALE_EMPS})$

Figure 8.5 The CARTESIAN PRODUCT (CROSS PRODUCT) operation.

Relational Operator CARTESIAN PRODUCT (cont.)

EMP_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	...
Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	...
Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	...
Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	...
Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	...
Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	...
Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	...
Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	...
Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	...
Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...
Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	...
Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	...
Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	...
Joyce	English	453453453	333445555	Alice	F	1986-04-05	...
Joyce	English	453453453	333445555	Theodore	M	1983-10-25	...
Joyce	English	453453453	333445555	Joy	F	1958-05-03	...
Joyce	English	453453453	987654321	Abner	M	1942-02-28	...
Joyce	English	453453453	123456789	Michael	M	1988-01-04	...
Joyce	English	453453453	123456789	Alice	F	1988-12-30	...
Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	...

ACTUAL_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

RESULT

Fname	Lname	Dependent_name
Jennifer	Wallace	Abner

Example: Retrieve a list of names of each female employee's dependents:
Steps 3, 4 and 5:

$\text{EMP_DEPENDENTS} \leftarrow \text{EMP_NAMES} \times \text{DEPENDENT}$

$\text{ACTUAL_DEPENDENTS} \leftarrow \sigma_{\text{Ssn}=\text{Essn}}(\text{EMP_DEPENDENTS})$

$\text{RESULT} \leftarrow \pi_{\text{Fname}, \text{Lname}, \text{Dependent_name}}(\text{ACTUAL_DEPENDENTS})$

Figure 8.5 (continued) The CARTESIAN PRODUCT (CROSS PRODUCT) operation.

Johns Hopkins Engineering

Principles of Database Systems

Module 5 / Lecture 5

The Relational Data Model and Relational Algebra



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Relational Operator JOIN

- JOIN
 - Returns a relation consisting of all possible tuples that are a combination of two tuples, one from each of two specified relations, such that the two tuples contributing to any given combination have a common value for the common attribute(s) of the two relations.

Relational Operator JOIN (cont.)

- JOIN

- This operation merges two or more relations into one relation.
- The JOIN operation combined with other operations for relations can process entities and relationships in a relational database.

- JOIN Types

- THETA JOIN, EQUIJOIN, NATURAL JOIN

Relational Operator THETA JOIN

- THETA JOIN (\bowtie)
 - Similar to a CARTESIAN PRODUCT followed by a SELECT. It has a *join condition*.

$$R(A_1, A_2, \dots, A_m, B_1, B_2, \dots, B_n) \\ \leftarrow R_1(A_1, A_2, \dots, A_m) \bowtie_{\langle \text{join condition} \rangle} R_2(B_1, B_2, \dots, B_n)$$

Join condition is one of the comparison operators $\{=, <, <=, >, >=, <>\}$

Relational Operator EQUIJOIN

- EQUIJOIN (e.g., $R \leftarrow R1 \bowtie_{\langle join\ condition \rangle} R2$)
 - The most common join operation is called EQUIJOIN “ \bowtie ” that is to generate a relation having one or more pairs of attributes that have identical values in every tuple.
 - The *join condition* includes one or more equality comparisons involving attributes from R1 and R2. (A special case of THETA JOIN.)
 - In an EQUIJOIN, the join contains a redundant attribute(s) in the result relation R.

Relational Operator EQUIJOIN (cont.)

■ EQUIJOIN

○ Example:

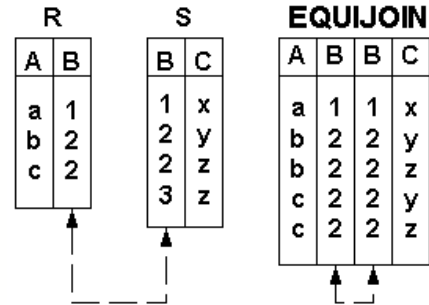
Retrieve all EMPLOYEES and their DEPENDENT information:

$\text{ACTUAL_DEPENDENT} \leftarrow \text{EMP} \bowtie_{\text{SSN}=\text{ESSN}} \text{DEPENDENT}$

Retrieve each DEPARTMENT's name and its manager's name:

$\text{TEMP} \leftarrow \text{DEPARTMENT} \bowtie_{\text{MGRSSN}=\text{SSN}} \text{EMPLOYEE}$

$\text{RESULT} \leftarrow \pi_{\text{DNAME}, \text{FNAME}, \text{LNAME}} (\text{TEMP})$



Relational Operator EQUIJOIN (cont.)

- EQUIJOIN

- Example: Retrieve each EMPLOYEE and his/her DEPARTMENT's name and its location:

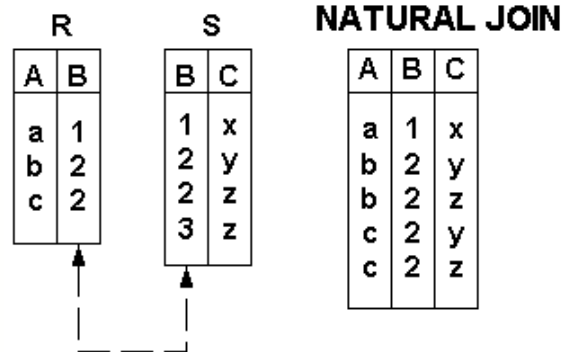
EMPLOYEE |X| DEPTNO=DEPTNO DEPARTMENT

EMPNO	ENAME	MGR	DEPTNO	DEPTNO	DNAME	LOC
7369	SMITH	7902	20	20	RESEARCH	DALLAS
7499	ALLEN	7698	30	30	SALES	CHICAGO
7521	WARD	7698	30	30	SALES	CHICAGO
7566	JONES	7839	20	20	RESEARCH	DALLAS
7654	MARTIN	7698	30	30	SALES	CHICAGO
7698	BLAKE	7839	30	30	SALES	CHICAGO
7782	CLARK	7839	10	10	ACCOUNTING	NEW YORK
7788	SCOTT	7566	20	20	RESEARCH	DALLAS
7839	KING		10	10	ACCOUNTING	NEW YORK
7844	TURNER	7698	30	30	SALES	CHICAGO
7876	ADAMS	7788	20	20	RESEARCH	DALLAS
7900	JAMES	7698	30	30	SALES	CHICAGO
7902	FORD	7566	20	20	RESEARCH	DALLAS
7934	MILLER	7782	10	10	ACCOUNTING	NEW YORK

Relational Operator NATURAL JOIN

- NATURAL JOIN

- The same operation as EQUIJOIN except that the redundant attribute(s) of the second relation is excluded in the resulting relation. In fact, this is the most preferable/desirable relation for a join operation.



Relational Operator NATURAL JOIN (cont.)

- The degree of a NATURAL JOIN is the sum of the degrees of the two relations minus the number of attributes in the EQUIJOIN condition.
- Example: Retrieve each EMPLOYEE's name and the name of the DEPARTMENT he/she works for:

Using NATURAL JOIN

```
TEMP ← EMPLOYEE * $\rho(\dots, Dno)$  DEPARTMENT (Note:  $\rho$  “rename” Dnumber)  
RESULT ←  $\pi_{FNAME, LNAME, DNAME}$  (TEMP)
```

Using EQUIJOIN

```
TEMP ← EMPLOYEE ⋈ $(Dno)=(Dnumber)$  DEPARTMENT  
RESULT ←  $\pi_{FNAME, LNAME, DNAME}$  (TEMP)
```

Relational Operator NATURAL JOIN (cont.)

(a)

PROJ_DEPT

Pname	Pnumber	Plocation	Dnum	Dname	Mgr_ssn	Mgr_start_date
ProductX	1	Bellaire	5	Research	333445555	1988-05-22
ProductY	2	Sugarland	5	Research	333445555	1988-05-22
ProductZ	3	Houston	5	Research	333445555	1988-05-22
Computerization	10	Stafford	4	Administration	987654321	1995-01-01
Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

(b)

DEPT_LOCS

Dname	Dnumber	Mgr_ssn	Mgr_start_date	Location
Headquarters	1	888665555	1981-06-19	Houston
Administration	4	987654321	1995-01-01	Stafford
Research	5	333445555	1988-05-22	Bellaire
Research	5	333445555	1988-05-22	Sugarland
Research	5	333445555	1988-05-22	Houston

Figure 8.7 Results of two natural join operations. (a) $\text{proj_dept} \leftarrow \text{project} * \text{dept}$. (b) $\text{dept_locs} \leftarrow \text{department} * \text{dept_locations}$.

Relational Operator DIVISION

■ DIVISION

- Defines a relation over the attributes that takes two relations, and a first relation match the combination of every tuple in the second relation.

1	a
1	b
2	a
2	b
3	a
3	c

/

a
b

=

1
2

Relational Operator DIVISION (cont.)

- The complete set of relational algebra operations $\{\sigma, \pi, \cup, -, \times\}$. Any other relation algebra operations can be represented as a sequence of operations from this set.
- The DIVISION operator can be expressed as a sequence of π , \times , and $-$ operations as follows:
 - $T_1 \leftarrow \pi_Y(R)$
 - $T_2 \leftarrow \pi_Y((S \times T_1) - R)$
 - $T \leftarrow T_1 - T_2$

Relational Operator DIVISION (cont.)

(a)				(b)	
SSN_PNOS		SMITH_PNOS		R	
Essn	Pno	Pno		A	B
123456789	1	1		a1	b1
123456789	2	2		a2	b1
666884444	3			a3	b1
453453453	1			a4	b1
453453453	2			a1	b2
333445555	2			a3	b2
333445555	3			a2	b3
333445555	10			a3	b3
333445555	20			a4	b3
999887777	30			a1	b4
999887777	10			a2	b4
987987987	10			a3	b4
987987987	30				
987654321	30				
987654321	20				
888665555	20				

		SSNS		S	
		Ssn		A	
		123456789		a1	
		453453453		a2	

		T	
		B	
		b1	
		b4	

Figure 8.8 The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) $T \leftarrow R \div S$.

Johns Hopkins Engineering

Principles of Database Systems

Module 5 / Lecture 6

The Relational Data Model and Relational Algebra



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Operations of Relational Algebra

- The bottom line to form a complex relational algebra condition is to break the list into pieces and test each one in sequence.
- The relational algebra uses one declarative expression to specify a retrieval request while a sequence of operations may be needed for an equivalent relation algebra.
- The relational algebra expresses what to be retrieved instead of how to retrieve.

Operations of Relational Algebra (cont.)

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R .	$\sigma_{\langle \text{selection condition} \rangle}(R)$
PROJECT	Produces a new relation with only some of the attributes of R , and removes duplicate tuples.	$\pi_{\langle \text{attribute list} \rangle}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$, OR $R_1 \bowtie_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 \star_{\langle \text{join condition} \rangle} R_2$, OR $R_1 \star_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$ OR $R_1 \star R_2$

Operations of Relational Algebra (cont.)

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2 ; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2 ; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2 ; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2 .	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Notation for Query Trees

- A query can be expressed in a tree structure (a query tree) with various steps of data retrieval using relation algebra.

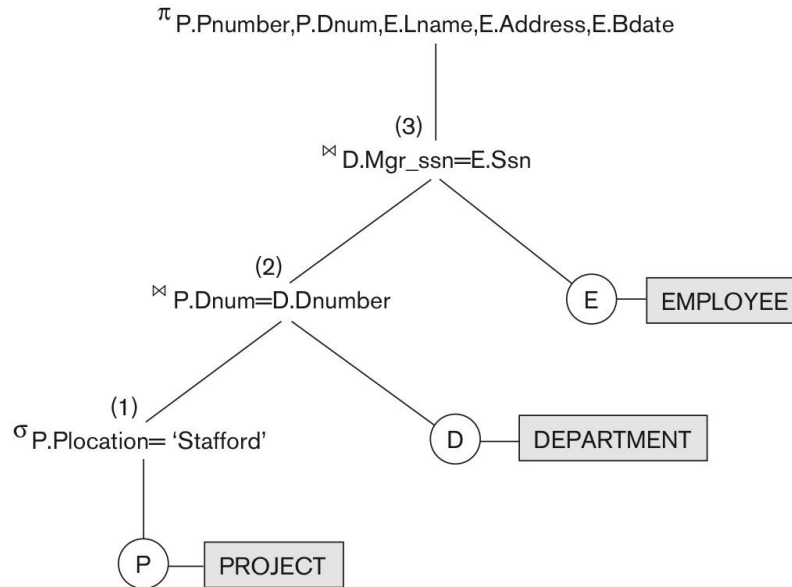


Figure 8.9 Query tree corresponding to the relational algebra expression for Q2.

Additional Relational Operations

- For developing database applications, additional operations are needed that were not part of the *original* relational algebra. These include:
 - Aggregate functions and grouping
 - Recursive Closure Operations
 - OUTER JOIN and OUTER UNION

Aggregate Functions

- Functions such as SUM, COUNT, AVERAGE, MINIMUM, MAXIMUM are often applied to sets of values or sets of tuples in database applications.
- The grouping attributes may be applied as needed.
<grouping attributes> **F** *<function list>* (R)

Aggregate Functions (cont.)

- Aggregate Function (\exists or F) Example:
Retrieve the average salary of all employees
 $R(\text{AVGSAL}) \leftarrow F_{\text{AVERAGE SALARY}}(\text{EMPLOYEE})$

For each department, retrieve the department number, the number of employees, and the average salary in the department. The grouping is needed and DNO is called the grouping attribute:

$$R(\text{DNO}, \text{NUMEMPS}, \text{AVGSAL}) \leftarrow$$
$$F_{\text{DNO } \underline{\text{COUNT SSN}}, \underline{\text{AVERAGE SALARY}}}(\text{EMPLOYEE})$$

Aggregate Functions (cont.)

R

(a)

Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

(b)

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

(c)

Count_ssn	Average_salary
8	35125

Figure 8.8 The aggregate function operation. a. $\rho_{R(Dno, No_of_employees, Average_sal)}(Dno \Join COUNT Ssn, AVERAGE Salary (EMPLOYEE))$. b. $Dno \Join COUNT Ssn, AVERAGE Salary (EMPLOYEE)$. c. $\Join COUNT Ssn, AVERAGE Salary (EMPLOYEE)$.

Recursive Closure Operations

- Recursive closure operations apply to a recursive relationship (employee and supervisor).
 - Retrieve all employees (directly or indirectly) supervised by an employee
 - Can be implemented in SQL as hierarchical queries

OUTER JOIN

- In a regular EQUIJOIN or NATURAL JOIN operation, tuples in R_1 or R_2 that do not have matching tuples in the other relation do *NOT* appear in the result.
- Some queries require all tuples in R_1 (or R_2 or both) to appear in the result.
- When no matching tuples are found, **nulls** are placed for the missing attributes.

OUTER JOIN (cont.)

- LEFT OUTER JOIN:

$R1 \bowtie R2$ lets every tuple in $R1$ appear in the result.

- RIGHT OUTER JOIN:

$R1 \bowtie R2$ lets every tuple in $R2$ appear in the result.

- FULL OUTER JOIN:

$R1 \bowtie R2$ lets every tuple in $R1$ and $R2$ appear in the result, padding tuples with nulls when no matching tuples are found.

OUTER JOIN (cont.)

■ Examples:

- LEFT (NATURAL) OUTER JOIN: $R1 \Join_L R2$
- RIGHT (NATURAL) OUTER JOIN: $R1 \Join_R R2$
- FULL (NATURAL) OUTER JOIN: $R1 \Join_F R2$

R	
A	B
a	1
b	2
c	2
d	5

S	
B	C
1	x
2	y
2	z
3	z
4	q

**LEFT NATURAL
OUTER JOIN**

A	B	C
a	1	x
b	2	y
b	2	z
c	2	y
c	2	z
d	5	

**RIGHT NATURAL
OUTER JOIN**

A	B	C
a	1	x
b	2	y
b	2	z
c	2	y
c	2	z
	3	z
	4	q

**FULL NATURAL
OUTER JOIN**

A	B	C
a	1	x
b	2	y
b	2	z
c	2	y
c	2	z
d	5	
	3	z
	4	q

OUTER JOIN (cont.)

■ Example:

$$\text{RESULT} \leftarrow \pi_{\text{Fname, Minit, Lname, Dname}} (\text{EMPLOYEE} \bowtie_{\text{Ssn=Mgr_ssn}} \text{DEPARTMENT})$$

RESULT

Fname	Minit	Lname	Dname
John	B	Smith	NULL
Franklin	T	Wong	Research
Alicia	J	Zelaya	NULL
Jennifer	S	Wallace	Administration
Ramesh	K	Narayan	NULL
Joyce	A	English	NULL
Ahmad	V	Jabbar	NULL
James	E	Borg	Headquarters

Figure 8.12 The result of a LEFT OUTER JOIN operation.