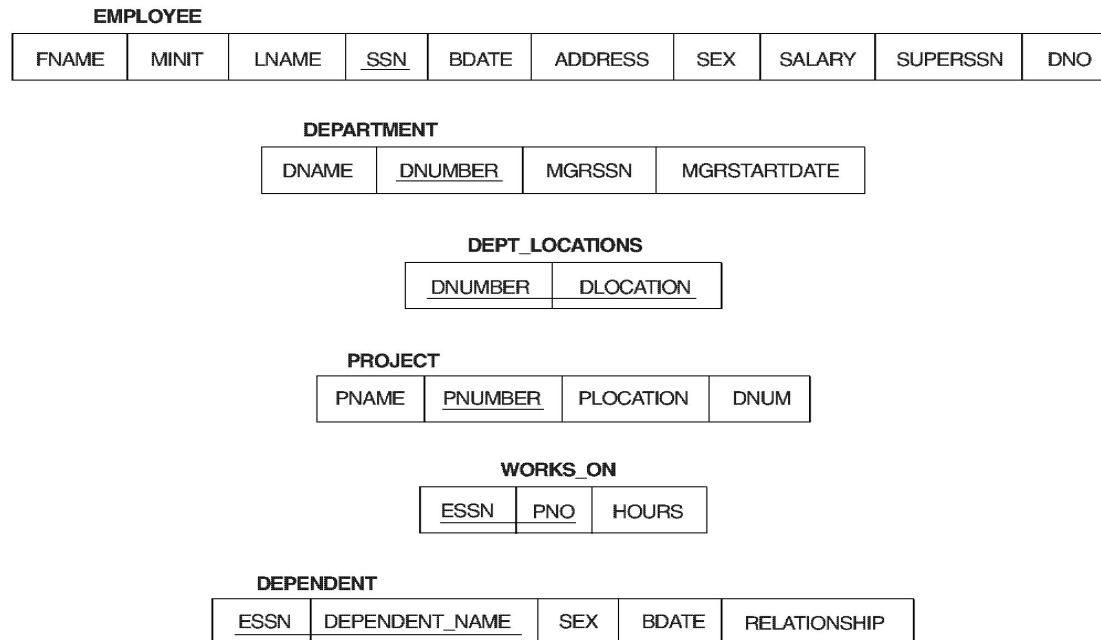


The Relational Model and Relational Algebra

Database State for COMPANY

Figure 7.5 Schema diagram for the COMPANY relational database schema; the primary keys are underlined.



Relational Model

- The relational Model of Data is based on the concept of a Relation.
- A Relation is a mathematical concept based on the ideas of sets.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations
- 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 above paper caused a major revolution in the field of Database management and earned Ted Codd the coveted ACM Turing Award.

Relational Model-Informal Definition

- RELATION: A table of values
 - A relation may be thought of as a **set of rows**.
 - A relation may alternately be thought of as a **set of columns**.
 - Each row represents a fact that corresponds to a real-world **entity** or **relationship**.
 - Each row has a value of an item or set of items that uniquely identifies that row in the table.
 - Sometimes row-ids or sequential numbers are assigned to identify the rows in the table.
 - Each column typically is called by its column name or column header or attribute name.

Relational Model-Formal Definition

- A **Relation** may be defined in multiple ways.
- The **Schema** of a Relation: $R (A_1, A_2, \dots, A_n)$

Relation schema R is defined over **attributes** A_1, A_2, \dots, A_n

For Example -

CUSTOMER (Cust-id, Cust-name, Address, Phone#)

Here, CUSTOMER is a relation defined over the four attributes Cust-id, Cust-name, Address, Phone#, each of which has a **domain** or a set of valid values. For example, the domain of Cust-id is 6 digit numbers.

Relational Model

- A **tuple** is an ordered set of values
- Each value is derived from an appropriate domain.
- Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four values.
- **<632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">** is a tuple belonging to the CUSTOMER relation.
- A relation may be regarded as a *set of tuples* (rows).
- Columns in a table are also called attributes of the relation.

Relational Model

- A **domain** has a logical definition: e.g.,
“USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.
- A domain may have a data-type or a format defined for it. The USA_phone_numbers may have a format: (ddd)-ddd-dddd where each d is a decimal digit. E.g., Dates have various formats such as monthname, date, year or yyyy-mm-dd, or dd mm,yyyy etc.
- An attribute designates the **role** played by the domain. E.g., the domain Date may be used to define attributes “Invoice-date” and “Payment-date”.
- The relation is formed over the cartesian product of the sets; each set has values from a domain; that domain is used in a specific role which is conveyed by the attribute name.
- For example, attribute Cust-name is defined over the domain of strings of 25 characters. The role these strings play in the CUSTOMER relation is that of the name of customers.
- Formally,

Given $R(A_1, A_2, \dots, A_n)$

$$r(R) \subset \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$$

- R: schema of the relation
- r of R: a specific "value" or population of R.
- R is also called the **intension** of a relation
- r is also called the **extension** of a relation

Relational Model

Let $S1 = \{0,1\}$

Let $S2 = \{a,b,c\}$

Let $R \subset S1 \times S2$

Then for example: $r(R) = \{ \langle 0,a \rangle, \langle 0,b \rangle, \langle 1,c \rangle \}$

is one possible “state” or “population” or “extension” r of the relation R , defined over domains $S1$ and $S2$. It has three tuples.

Informal Terms

Table

Column

Row

Values in a column

Table Definition

Populated Table

Formal Terms

Relation

Attribute/Domain

Tuple

Domain

Schema of a Relation

Extension

Relational Model

- **Ordering of tuples in a relation $r(R)$:** The tuples are *not* considered to be ordered, even though they appear to be in the tabular form.
- **Ordering of attributes in a relation schema R** (and of values within each tuple): We will consider the attributes in $R(A_1, A_2, \dots, A_n)$ and the values in $t = \langle v_1, v_2, \dots, v_n \rangle$ to be *ordered*. (However, a more general *alternative definition* of relation does not require this ordering).
- **Values in a tuple:** All values are considered *atomic* (indivisible). A special **null** value is used to represent values that are unknown or inapplicable to certain tuples.
- Notation:
 - We refer to **component values** of a tuple t by $t[A_i] = v_i$ (the value of attribute A_i for tuple t).
Similarly, $t[A_u, A_v, \dots, A_w]$ refers to the subtuple of t containing the values of attributes A_u, A_v, \dots, A_w , respectively.

Relational Integrity Constraints

- Constraints are *conditions* that must hold on *all* valid relation instances. There are three main types of constraints:
 1. **Key** constraints
 2. **Entity integrity** constraints
 3. **Referential integrity** constraints

Key Constraint

- **Superkey of R:** A set of attributes SK of R such that no two tuples *in any valid relation instance* $r(R)$ will have the same value for SK. That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$.
- **Key of R:** A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

Example: The CAR relation schema:

CAR(State, Reg#, SerialNo, Make, Model, Year)

has two keys $\text{Key1} = \{\text{State}, \text{Reg}\# \}$, $\text{Key2} = \{\text{SerialNo}\}$, which are also superkeys.

$\{\text{SerialNo}, \text{Make}\}$ is a superkey but *not* a key.

- If a relation has *several candidate keys*, one is chosen arbitrarily to be the **primary key**. The primary key attributes are *underlined*.

Figure 7.4 The CAR relation with two candidate keys:
LicenseNumber and EngineSerialNumber.

CAR	<u>LicenseNumber</u>	EngineSerialNumber	Make	Model	Year
	Texas ABC-739	A69352	Ford	Mustang	96
	Florida TVP-347	B43696	Oldsmobile	Cutlass	99
	New York MPO-22	X83554	Oldsmobile	Delta	95
	California 432-TFY	C43742	Mercedes	190-D	93
	California RSK-629	Y82935	Toyota	Camry	98
	Texas RSK-629	U028365	Jaguar	XJS	98

Entity Integrity

- **Relational Database Schema:** A set S of relation schemas that belong to the same database. S is the *name* of the **database**.

$$S = \{R_1, R_2, \dots, R_n\}$$

- **Entity Integrity:** The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of $r(R)$. This is because primary key values are used to *identify* the individual tuples.

$$t[\text{PK}] \neq \text{null for any tuple } t \text{ in } r(R)$$

- 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

- A constraint involving *two* relations (the previous constraints involve a *single* relation).
- Used to specify a *relationship* among tuples in two relations: the **referencing relation** and the **referenced relation**.
- Tuples in the *referencing relation* R_1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation* R_2 . A tuple t_1 in R_1 is said to **reference** a tuple t_2 in R_2 if $t_1[\text{FK}] = t_2[\text{PK}]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1.\text{FK}$ to R_2 .

Statement of the constraint

The value in the foreign key column (or columns) FK of the the **referencing relation** R_1 can be either:

- (1) a value of an existing primary key value of the corresponding primary key PK in the **referenced relation** R_2 , or..
- (2) a null.

In case (2), the FK in R_1 should not be a part of its own primary key.

Others Constraints

Semantic Integrity Constraints:

- based on application semantics and cannot be expressed by the model per se
- E.g., “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- A *constraint specification language* may have to be used to express these
- SQL-99 allows triggers and ASSERTIONS to allow for some of these

Operation:

There are three basic operations that can change the states of relations in the database: Insert, Delete, and Update (or Modify). They insert new data, delete old data, or modify existing data records. Insert is used to insert one or more new tuples in a relation, Delete is used to delete tuples, and Update (or Modify) is used to change the values of some attributes in existing tuples. Whenever these operations are applied, the integrity constraints specified on the relational database schema should not be violated

Insert Operation

The Insert operation provides a list of attribute values for a new tuple t that is to be inserted into a relation R . Insert can violate any of the four types of constraints discussed in the previous section. Domain constraints can be violated if an attribute value is given that does not appear in the corresponding domain or is not of the appropriate data type.

Key constraints can be violated if a key value in the new tuple t already exists in another tuple in the relation $r(R)$. Entity integrity can be violated if any part of the primary key of the new tuple t is NULL. Referential integrity can be violated if the value of any foreign key in t refers to a tuple that does not exist in the referenced relation.

Update Operation

- Integrity constraints should not be violated by the update operations.
- Several update operations may have to be grouped together.
- Updates may *propagate* to cause other updates automatically. This may be necessary to maintain integrity constraints.
- In case of integrity violation, several actions can be taken:
 - Cancel the operation that causes the violation (REJECT option)
 - Perform the operation but inform the user of the violation
 - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
 - Execute a user-specified error-correction routine

Delete Operation

The Delete operation can violate only referential integrity. This occurs if the tuple being deleted is referenced by foreign keys from other tuples in the database. To specify deletion, a condition on the attributes of the relation selects the tuple (or tuples) to be deleted.

1. Delete the WORKS_ON tuple with Essn = '999887777' and Pno = 10.

Result: This deletion is acceptable and deletes exactly one tuple.

1. Delete the EMPLOYEE tuple with Ssn = '999887777'.

Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

1. Delete the EMPLOYEE tuple with Ssn = '333445555'.

Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations.

Transaction Concept

A database application program running against a relational database typically executes one or more transactions. A transaction is an executing program that includes some database operations, such as reading from the database, or applying insertions, deletions, or updates to the database. At the end of the transaction, it must leave the database in a valid or consistent state that satisfies all the constraints specified on the database schema.

A large number of commercial applications running against relational databases in online transaction processing (OLTP) systems are executing transactions at rates that reach several hundred per second.

Relational Algebra

- The basic set of operations for the relational model is known as the relational algebra. These operations enable a user to specify basic retrieval requests.
- The result of a retrieval is a new relation, which may have been formed from one or more relations. The **algebra operations** thus produce new relations, which can be further manipulated using operations of the same algebra.
- A sequence of relational algebra operations forms a **relational algebra expression**, whose result will also be a relation that represents the result of a database query (or retrieval request).

Unary Relational Operations

- **SELECT Operation**

SELECT operation is used to select a *subset* of the tuples from a relation that satisfy a **selection condition**. It is a filter that keeps only those tuples that satisfy a qualifying condition – those satisfying the condition are selected while others are discarded.

Example: To select the EMPLOYEE tuples whose department number is four or those whose salary is greater than \$30,000 the following notation is used:

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

$\sigma_{SALARY > 30,000}$ (EMPLOYEE)

In general, the select operation is denoted by $\sigma_{\langle \text{selection condition} \rangle}(R)$ where the symbol σ (sigma) is used to denote the select operator, and the selection condition is a Boolean expression specified on the attributes of relation R

Unary Relational Operations

SELECT Operation Properties

The SELECT operation $\sigma_{\langle \text{selection condition} \rangle}(R)$ produces a relation S that has the same schema as R

The SELECT operation σ is **commutative**; i.e.,

$$\sigma_{\langle \text{condition1} \rangle}(\sigma_{\langle \text{condition2} \rangle}(R)) = \sigma_{\langle \text{condition2} \rangle}(\sigma_{\langle \text{condition1} \rangle}(R))$$

A cascaded SELECT operation **may be applied in any order**; i.e.,

$$\sigma_{\langle \text{condition1} \rangle}(\sigma_{\langle \text{condition2} \rangle}(\sigma_{\langle \text{condition3} \rangle}(R))) = \sigma_{\langle \text{condition2} \rangle}(\sigma_{\langle \text{condition3} \rangle}(\sigma_{\langle \text{condition1} \rangle}(R)))$$

A cascaded SELECT operation may be replaced by a single selection with a conjunction of all the conditions; i.e.,

$$\sigma_{\langle \text{condition1} \rangle}(\sigma_{\langle \text{condition2} \rangle}(\sigma_{\langle \text{condition3} \rangle}(R))) = \sigma_{\langle \text{condition1} \rangle \text{ AND } \langle \text{condition2} \rangle \text{ AND } \langle \text{condition3} \rangle}(R)$$

Unary Relational Operations (cont.)

Figure 7.8 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) $\pi_{SEX, SALARY}(EMPLOYEE)$

(a)

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
Franklin	T	Wong	333445555	1955-12-08	638 Voss,Houston,TX	M	40000	888665555	5
Jennifer		Wallace	987654321	1941-06-20	291 Berry,Bellaire,TX	F	43000	888665555	4
Ramesh		Narayan	666884444	1962-09-15	975 FireOak,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

Unary Relational Operations (cont.)

- **PROJECT Operation**

This operation selects certain *columns* from the table and discards the other columns. The PROJECT creates a vertical partitioning – one with the needed columns (attributes) containing results of the operation and other containing the discarded Columns.

Example: To list each employee's first and last name and salary, the following is used:

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

The general form of the project operation is $\pi\langle\text{attribute list}\rangle(R)$ where π (pi) is the symbol used to represent the project operation and $\langle\text{attribute list}\rangle$ is the desired list of attributes from the attributes of relation R.

The project operation *removes any duplicate tuples*, so the result of the project operation is a set of tuples and hence a valid relation.

Unary Relational Operations (cont.)

PROJECT Operation Properties

- The number of tuples in the result of projection $\pi_{\langle \text{list} \rangle}(R)$ is always less or equal to the number of tuples in R .
- If the list of attributes includes a key of R , then the number of tuples is equal to the number of tuples in R .
- $\pi_{\langle \text{list1} \rangle}(\pi_{\langle \text{list2} \rangle}(R)) = \pi_{\langle \text{list1} \rangle}(R)$ as long as $\langle \text{list2} \rangle$ contains the attributes in $\langle \text{list1} \rangle$

Figure 7.8 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) $\pi_{SEX, SALARY}(EMPLOYEE)$

(a)

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
Franklin	T	Wong	333445555	1955-12-08	638 Voss,Houston,TX	M	40000	888665555	5
Jennifer		Wallace	987654321	1941-06-20	291 Berry,Bellaire,TX	F	43000	888665555	4
Ramesh		Narayan	666884444	1962-09-15	975 FireOak,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

Unary Relational Operations (cont.)

- **Rename Operation**

We may want to apply several relational algebra operations one after the other. Either we can write the operations as a single **relational algebra expression** by nesting the operations, or we can apply one operation at a time and create **intermediate result relations**. In the latter case, we must give names to the relations that hold the intermediate results.

Example: To retrieve the first name, last name, and salary of all employees who work in department number 5, we must apply a select and a project operation. We can write a single relational algebra expression as follows: $\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO}=5}(\text{EMPLOYEE}))$

OR We can explicitly show the sequence of operations, giving a name to each intermediate relation:

DEP5_EMPS $\leftarrow \sigma_{\text{DNO}=5}(\text{EMPLOYEE})$

RESULT $\leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5_EMPS})$

Unary Relational Operations (cont.)

- **Rename Operation**

The rename operator is ρ

The general Rename operation can be expressed by any of the following forms:

- $\rho_{S(B_1, B_2, \dots, B_n)}(R)$ is a renamed relation S based on R with column names B_1, B_1, \dots, B_n .
- $\rho_S(R)$ is a renamed relation S based on R (which does not specify column names).
- $\rho_{(B_1, B_2, \dots, B_n)}(R)$ is a renamed relation with column names B_1, B_1, \dots, B_n which does not specify a new relation name.

Unary Relational Operations (cont.)

Figure 7.9 Results of relational algebra expressions.

(a) $\pi_{\text{LNAME, FNAME, SALARY}}(\sigma_{\text{DNO}=5}(\text{EMPLOYEE}))$. (b) The same expression using intermediate relations and renaming of attributes.

(a)

FNAME	LNAME	SALARY
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

(b)

TEMP	FNAME	MINIT	LNAME	<u>SSN</u>	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	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

	FIRSTNAME	LASTNAME	SALARY
	John	Smith	30000
	Franklin	Wong	40000
	Ramesh	Narayan	38000
	Joyce	English	25000

Relational Algebra Operations From Set Theory

UNION Operation

The result of this operation, denoted by $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R and S . Duplicate tuples are eliminated.

Example: To retrieve the social security numbers of all employees who either work in department 5 or directly supervise an employee who works in department 5, we can use the union operation as follows:

DEP5_EMPS $\leftarrow \sigma_{\text{DNO}=5}(\text{EMPLOYEE})$

RESULT1 $\leftarrow \pi_{\text{SSN}}(\text{DEP5_EMPS})$

RESULT2(SSN) $\leftarrow \pi_{\text{SUPERSSN}}(\text{DEP5_EMPS})$

RESULT $\leftarrow \text{RESULT1} \cup \text{RESULT2}$

The union operation produces the tuples that are in either **RESULT1** or **RESULT2** or both. The two operands must be “type compatible”.

Relational Algebra Operations From Set Theory

- **Type Compatibility**

- The operand relations $R_1(A_1, A_2, \dots, A_n)$ and $R_2(B_1, B_2, \dots, B_n)$ must have the same number of attributes, and the domains of corresponding attributes must be compatible; that is, $\text{dom}(A_i) = \text{dom}(B_i)$ for $i=1, 2, \dots, n$.
- The resulting relation for $R_1 \cup R_2$, $R_1 \cap R_2$, or $R_1 - R_2$ has the same attribute names as the *first* operand relation R_1 (by convention).

Relational Algebra Operations From Set Theory

- UNION Example**

STUDENT	FN	LN
	Susan	Yao
	Ramesh	Shah
	Johnny	Kohler
	Barbara	Jones
	Amy	Ford
	Jimmy	Wang
	Ernest	Gilbert

INSTRUCTOR	FNAME	LNAME
	John	Smith
	Ricardo	Browne
	Susan	Yao
	Francis	Johnson
	Ramesh	Shah

(b)

FN	LN
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

FN	LN
Susan	Yao
Ramesh	Shah

(d)

FN	LN
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

FNAME	LNAME
John	Smith
Ricardo	Browne
Francis	Johnson

Relational Algebra Operations From Set Theory (cont.) – use Fig. 6.4

Figure 7.11 Illustrating the set operations union, intersection, and difference. (a) Two union compatible relations. (b) $\text{STUDENT} \cup \text{INSTRUCTOR}$. (c) $\text{STUDENT} \cap \text{INSTRUCTOR}$. (d) $\text{STUDENT} - \text{INSTRUCTOR}$. (e) $\text{INSTRUCTOR} - \text{STUDENT}$.

(a)

STUDENT	FN	LN
	Susan	Yao
	Ramesh	Shah
	Johnny	Kohler
	Barbara	Jones
	Amy	Ford
	Jimmy	Wang
	Ernest	Gilbert

INSTRUCTOR	FNAME	LNAME
	John	Smith
	Ricardo	Browne
	Susan	Yao
	Francis	Johnson
	Ramesh	Shah

(b)

FN	LN
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

(c)

FN	LN
Susan	Yao
Ramesh	Shah

(d)

FN	LN
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

FNAME	LNAME
John	Smith
Ricardo	Browne
Francis	Johnson

Relational Algebra Operations From Set Theory (cont.)

- INTERSECTION OPERATION

The result of this operation, denoted by $R \cap S$, is a relation that includes all tuples that are in both R and S . The two operands must be "type compatible"

Example: The result of the intersection operation (figure below) includes only those who are both students and instructors.

FN	LN
Susan	Yao
Ramesh	Shah

(d)

FN	LN
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

STUDENT \cap INSTRUCTOR

Relational Algebra Operations From Set Theory (cont.)

- Set Difference (or MINUS) Operation

The result of this operation, denoted by $R - S$, is a relation that includes all tuples that are in R but not in S . The two operands must be "type compatible".

Example: The figure shows the names of students who are not instructors, and the names of instructors who are not students.

STUDENT	FN	LN
	Susan	Yao
	Ramesh	Shah
	Johnny	Kohler
	Barbara	Jones
	Amy	Ford
	Jimmy	Wang
	Ernest	Gilbert

FN	LN
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

STUDENT-INSTRUCTOR

FNAME	LNAME
John	Smith
Ricardo	Browne
Francis	Johnson

INSTRUCTOR-STUDENT

Relational Algebra Operations From Set Theory

- Notice that both union and intersection are *commutative operations*; that is

$$\mathbf{R \cup S = S \cup R, \text{ and } R \cap S = S \cap R}$$

- Both union and intersection can be treated as n-ary operations applicable to any number of relations as both are *associative operations*; that is

$$\mathbf{R \cup (S \cup T) = (R \cup S) \cup T, \text{ and } (R \cap S) \cap T = R \cap (S \cap T)}$$

- The minus operation is *not commutative*; that is, in general

$$\mathbf{R - S \neq S - R}$$

Relational Algebra Operations From Set Theory

- **CARTESIAN (or cross product) Operation**

- This operation is used to combine tuples from two relations in a combinatorial fashion. In general, the result of $R(A_1, A_2, \dots, A_n) \times S(B_1, B_2, \dots, B_m)$ is a relation Q with degree $n + m$ attributes $Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$, in that order. The resulting relation Q has one tuple for each combination of tuples—one from R and one from S .
- Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then
 $|R \times S|$ will have $n_R * n_S$ tuples.
- The two operands do NOT have to be "type compatible"

Example:

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

EMPNAMES $\leftarrow \pi_{FNAME, LNAME, SSN}(\text{FEMALE_EMPS})$

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

Figure 7.12 An illustration of the CARTESIAN PRODUCT operation.

FEMALE_EMPS	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	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

EMP_NAMES	FNAME	LNAME	SSN
	Alicia	Zelaya	999887777
	Jennifer	Wallace	987654321
	Joyce	English	453453453

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

Binary Relational Operations

- **JOIN Operation**

- The sequence of cartesian product followed by select is used quite commonly to identify and select related tuples from two relations, a special operation, called **JOIN**. It is denoted by a
- This operation is very important for any relational database with more than a single relation, because it allows us to process relationships among relations.
- The general form of a join operation on two relations $R(A_1, A_2, \dots, A_n)$ and $S(B_1, B_2, \dots, B_m)$ is:

$$R \underset{\langle \text{join condition} \rangle}{\bowtie} S$$

where R and S can be any relations that result from general *relational algebra expressions*.



Binary Relational Operations (cont.)

Example: Suppose that we want to retrieve the name of the manager of each department. To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple. We do this by using the join operation.

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

DEPT_MGR	DNAME	DNUMBER	MGRSSN	...	FNAME	MINIT	LNAME	SSN	...
	Research	5	333445555	...	Franklin	T	Wong	333445555	...
	Administration	4	987654321	...	Jennifer	S	Wallace	987654321	...
	Headquarters	1	888665555	...	James	E	Borg	888665555	...

FIGURE 6.6 Result of the JOIN operation $\text{DEPT_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{MGRSSN=SSN}} \text{EMPLOYEE}$.

Binary Relational Operations (cont.)

- **EQUIJOIN Operation**

The most common use of join involves join conditions with equality comparisons only. Such a join, where the only comparison operator used is =, is called an EQUIJOIN. In the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have *identical values* in every tuple.

The JOIN seen in the previous example was EQUIJOIN.

- **NATURAL JOIN Operation**

Because one of each pair of attributes with identical values is superfluous, a new operation called natural join—denoted by *—was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition.

The standard definition of natural join requires that the two join attributes, or each pair of corresponding join attributes, have the **same name** in both relations. If this is not the case, a renaming operation is applied first.

Example: To apply a natural join on the DNUMBER attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write:

DEPT_LOCS \leftarrow DEPARTMENT * DEPT_LOCATIONS

(a)

PROJ_DEPT	PNAME	<u>PNUMBER</u>	PLOCATION	DNUM	DNAME	MGRSSN	MGRSTARTDATE
	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	MGRSSN	MGRSTARTDATE	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 6.7 Results of two NATURAL JOIN operations. (a) PROJ_DEPT \leftarrow PROJECT * DEPT. (b) DEPT_LOCS \leftarrow DEPARTMENT * DEPT_LOCATIONS.

Complete Set of Relational Operations

- The set of operations including **select** σ , **project** π , **union** \cup , **set difference** $-$, and **cartesian product** \times is called a complete set because any other relational algebra expression can be expressed by a combination of these five operations.
- For example:

$$\mathbf{R} \cap \mathbf{S} = (\mathbf{R} \cup \mathbf{S}) - ((\mathbf{R} - \mathbf{S}) \cup (\mathbf{S} - \mathbf{R}))$$

$$\mathbf{R} \underset{\text{<join condition>}}{\bowtie} \mathbf{S} = \sigma_{\text{<join condition>}}(\mathbf{R} \times \mathbf{S})$$



Binary Relational Operations (cont.)

- **DIVISION Operation**

- The division operation is applied to two relations

$R(Z) \div S(X)$, where X subset Z . Let $Y = Z - X$ (and hence $Z = X \cup Y$); that is, let Y be the set of attributes of R that are not attributes of S .

- The result of DIVISION is a relation $T(Y)$ that includes a tuple t if tuples t_R appear in R with $t_R[Y] = t$, and with $t_R[X] = t_s$ for every tuple t_s in S .
- For a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with *every* tuple in S .

Binary Relational Operations (cont.)

SSN_PNOS	ESSN	PNO
	123456789	1
	123456789	2
	666884444	3
	453453453	1
	453453453	2
	333445555	2
	333445555	3
	333445555	10
	333445555	20
	999887777	30
	999887777	10
	987987987	10
	987987987	30
	987654321	30
	987654321	20
	888665555	20

R	A	B
	a1	b1
	a2	b1
	a3	b1
	a4	b1
	a1	b2
	a3	b2
	a2	b3
	a3	b3
	a4	b3
	a1	b4
	a2	b4
	a3	b4

TABLE 6.1 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.	$R_1^*_{\langle \text{JOIN CONDITION} \rangle} R_2$, OR $R_1^*_{(\langle \text{JOIN ATTRIBUTES 1} \rangle, \langle \text{JOIN ATTRIBUTES 2} \rangle)} R_2$ OR $R_1 * R_2$
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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$