Unit 5
The Relational Algebra and Relational Calculus

- Introduction
- Unary Relational Operations: SELECT and PROJECT
- Relational Algebra Operations from Set Theory
- Binary Relational Operations: JOIN and DIVISION
- Additional Relational Operations
- The Tuple Relational Calculus
- The Domain Relational Calculus

Introduction

- The basic set of operations for the formal relational model is the **relational algebra**.
- Relational algebra operations enable a user to specify basic retrieval requests as relational algebra expressions.
- The result of a retrieval query is a new relation, which can be further manipulated using operations of the same relational 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).

Introduction

- The relational algebra is very important for several reasons:
  - First, it provides a formal foundation for relational model operations.
  - Second, and perhaps more important, it is used as a basis for implementing and optimizing queries in the query processing and optimization modules that are integral parts of relational database management systems (RDBMSs)
  - Third, some of its concepts are incorporated into the SQL standard query language for RDBMSs.

Introduction

- Relational algebra operations can be divided into two groups.
  - One group includes set operations from mathematical set theory. These include UNION, INTERSECTION, SET DIFFERENCE, and CARTESIAN PRODUCT (aka CROSS PRODUCT).
  - Other group operations are developed specifically for relational databases – these include SELECT, PROJECT, and JOIN.
- Unary operators, such as SELECT, operate on a single relation and binary operators, such as UNION operate on two relations.
- Additional operations include aggregate functions, which summarize data from tables, as well as additional types of JOIN and UNION operations, known as OUTER JOINs and OUTER UNIONs.

Introduction

- In **relational calculus** expression, there is no order of operations to specify how to retrieve the query result – only what information the result should contain. This is the main distinguishing feature between relational algebra and relational calculus.
- Relational calculus is based on the logic called predicate logic. There are two variations of relational calculus. The **tuple relational calculus** and the **domain relational calculus**. In tuple relational calculus, variables range over tuples, whereas in domain relational calculus, variables range over the domains (values) of attributes.
- The SQL has some of its foundations based on tuple relational calculus.

Database State used in Examples

EMPLOYEE

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
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	Dnumber	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

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Database State used in Examples

WORKS_ON

Essn	Pno	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

Pname	Pnumber	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

Essn	Dependent_name	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

Unary Relational Operations

- **The SELECT Operation (σ):**
  - The SELECT operation is used to choose a subset of the tuples from a relation that satisfies a selection condition.
  - We can consider the SELECT operation to be a filter that keeps only those tuples that satisfy a qualifying condition.
  - Alternatively, we can consider the SELECT operation to restrict the tuples in a relation to only those tuples that satisfy the condition.
  - In general, the SELECT operation is denoted by

    $\sigma_{<\text{selection condition}>}(R)$

    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

The Boolean expression specified in is made up of a number of clauses of the form:

<attribute name><comparison op><constant value>

Or

<attribute name><comparison op><attribute name>

were <attribute name> is the name of an attribute of R, <comparison op> is normally one of the operators {=, <, ≤, >, ≥, ≠}, and <constant value> is a constant value from the attribute domain. Clauses can be connected by the standard Boolean operators **and**, **or**, and **not** to form a general selection condition.

Notice that R is generally a relational algebra expression whose result is a relation – the simplest such expression is just the name of a database relation.

The relation resulting from the SELECT operation has the same attributes as R.

Unary Relational Operations

- **Example 1:** To select the EMPLOYEE tuples whose salary is greater than $30,000, we write,

  $\sigma_{Salary > 30000}(EMPLOYEE)$

- **Example 2:** To select the tuples for all employees who either work in department 4 and make over $25,000 per year, or work in department 5 and make over $30,000, we can specify the following SELECT operation:

  $\sigma_{(Dno = 4 \text{ AND } Salary > 25000) \text{ OR } (Dno = 5 \text{ AND } Salary > 30000)}(EMPLOYEE)$

- SELECT is commutative: $\sigma_{<cond1>}(\sigma_{<cond2>}(R)) = \sigma_{<cond2>}(\sigma_{<cond1>}(R))$
- We can always combine SELECT operations into a single SELECT operation with a conjunctive (AND) condition: $\sigma_{<cond1>}(\sigma_{<cond2>}(R)) = \sigma_{<cond1> \text{ AND } <cond2>}(R))$

Unary Relational Operations

- **The PROJECT Operation (π):**
  - The PROJECT operation selects certain columns from the table and discards the other columns. If we are interested in only certain attributes of a relation, we use the PROJECT operation to project the relation over these attributes only.
  - In general, the PROJECT operation is denoted by

    $\pi_{<\text{attribute list}>}(R)$

    where π (pi) is the symbol used to represent the PROJECT operation, and <attribute list> is the desired sublist of attributes from the attributes of relation R.

    Notice that R is generally a relational algebra expression whose result is a relation – the simplest such expression is just the name of a database relation. The result of the PROJECT operation has only the attributes specified in in the same order as they appear in the list. Hence, its degree is equal to the number of attributes in <attribute list>.

Unary Relational Operations

- If the attribute list includes only nonkey attributes of R, duplicate tuples are likely to occur. The PROJECT operation removes any duplicate tuples.
- If duplicates are not eliminated, the result would be a multiset or bag of tuples rather than a set. This was not permitted in the formal relational model but is allowed in SQL.
- The number of tuples in a relation resulting from a PROJECT operation is always less than or equal to the number of tuples in R.
- For example, to project sex and salary of employees, we write

  $\pi_{\text{Sex, Salary}}$ (EMPLOYEE)
- $\pi_{<list1>}(\pi_{<list2>}(R)) = \pi_{<list1>}(R)$ as long as <list2> contains the attributes in <list1>; otherwise, the left-hand side is an incorrect.
- Note that commutativity does not hold on PROJECT.

Unary Relational Operations

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

(a)

Fname	Minit	Lname	Ssn	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

Unary Relational Operations

- **Sequence of Operations and the RENAME Operation:**
  - For most queries, we need to apply several relational algebra operations one after the other.
  - Either we can write the operations as a single relational algebra expression (aka **in-line 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.

    $\pi_{Fname,\ Lname,\ Salary}\ (\sigma_{Dno=5}(EMPLOYEE))$

    Alternatively, we can explicitly show the sequence of operations, giving a name to each intermediate relation, and using the assignment operation, denoted by $\leftarrow$ (left arrow).

Unary Relational Operations

$DEP5\_EMPS \leftarrow \sigma_{Dno=5}(EMPLOYEE)$

$RESULT \leftarrow \pi_{Fname,\ Lname,\ Salary}(DEP5\_EMPS)$

- It is sometimes simpler to break down a complex sequence of operations by specifying intermediate result relations than to write a single relational algebra expression.
- We can also use this technique to rename the attributes in the intermediate and result relations. To rename the attributes in a relation, we simply list the new attribute names in parentheses, as in the following example

$TEMP \leftarrow \sigma_{Dno=5}(EMPLOYEE)$

$R(First\_name, Last\_name, Salary) \leftarrow \pi_{Fname,\ Lname,\ Salary}(TEMP)$

- If no renaming is applied, the names of the attributes in the resulting relation of a SELECT operation are the same as those in the original relation and in the same order.

Unary Relational Operations

(a)

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

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

(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

Unary Relational Operations

- We can also define a formal **RENAME** operation – which can rename either the relation name or the attribute names, or both – as a **unary operator**. The general RENAME operation when applied to a relation R of degree n is denoted by any of the following three forms:

  $\rho_{S(B1, B2, \ldots, Bn)}(R)$ or $\rho_{S}(R)$ or $\rho_{(B1, B2, \ldots, Bn)}(R)$

  where the symbol ρ (rho) is used to denote the RENAME operator, S is the new relation name, and B_1, B_2, …, B_n are the new attribute names. The first expression renames both the relation and its attributes, the second renames the relation only, and the third renames the attributes only. If the attributes of R are (A_1, A_2, …, A_n) in that order, then each A_i is renamed as B_i.

Relational Algebra Operations form Set Theory

- **The UNION, INTERSECTION, and MINUS Operations:**
  - Several set theoretic operations are used to merge the elements of two sets in various ways, including UNION, INTERSECTION, and SET DIFFERENCE (also called MINUS or EXCEPT).
  - The two relations on which any of these three operations are applied must have the same type of tuples; this condition has been called **union compatibility** or **type compatibility**.
  - Two relations $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_n)$ are said to be union compatible (or type compatible) if they have the same degree n and if $dom(A_i) = dom(B_i)$ for $1 \leq i \leq n$.
  - Hence, the two relations have the same number of attributes and each corresponding pair of attributes has the same domain.

Relational Algebra Operations form Set Theory

- The **UNION** of relations R and S, denoted by R ∪ 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. For 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 write

  $DEP5\_EMPS \leftarrow \sigma_{Dno=5}(EMPLOYEE)$

  $RESULT1 \leftarrow \pi_{Ssn}(DEP5\_EMPS)$

  $RESULT2(Ssn) \leftarrow \pi_{Super\_ssn}(DEP5\_EMPS)$

  $RESULT \leftarrow RESULT1 \cup RESULT2$

- The **INTERSECTION** of relations R and S, denoted by R ∩ S, is a relation that includes all tuples that are in both R and S.
- The **SET DIFFERENCE** between relations R and S, denoted by R – S, is a relation that includes all tuples that are in R but not in S.

Relational Algebra Operations form Set Theory

The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT ∪ INSTRUCTOR. (c) STUDENT ∩ INSTRUCTOR. (d) STUDENT – INSTRUCTOR. (e) INSTRUCTOR – 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 form Set Theory

- Both UNION and INTERSECTION are commutative, that is, $R \cup S = S \cup R$ and $R \cap S = S \cap R$. Both UNION and INTERSECTION can be applicable to any number of relations and both are also associative operations; that is, $R \cup (S \cup T) = (R \cup S) \cup T$ and $(R \cap S) \cap T = R \cap (S \cap T)$.
- The MINUS operation is not commutative; that is, in general, $R - S \neq S - R$. Note that INTERSECTION can be expressed in terms of union and set difference as follows: $R \cap S = ((R \cup S) - (R - S)) - (S - R)$.

- **The CARTESIAN PRODUCT Operation:**
  - CARTESIAN PRODUCT operation – also known as CROSS PRODUCT or CROSS JOIN – is denoted by $\times$.
  - This is also a binary set operation, but the relations on which it is applied do not have to be union compatible.
  - This operation produces a new tuple by combining every tuple from one relation with every tuple from the other relation.

Relational Algebra Operations form Set Theory

- In general, the result of $R(A_1, A_2, \ldots, A_n) \times S(B_1, B_2, \ldots, B_m)$ is a relation Q with degree n + m attributes $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$, 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 × S will have $n_R * n_S$ tuples.
- For example, to retrieve first name, last name, and dependent name names of each female employee, we write

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

$$\text{EMPNAMES} \leftarrow \pi_{\text{Fname, Lname, Ssn}}(\text{FEMALE\_EMPS})$$

$$\text{EMP\_DEPENDENTS} \leftarrow \text{EMPNAMES} \times \text{DEPENDENT}$$

$$\text{ACTUAL\_DEPS} \leftarrow \sigma_{\text{Ssn=Essn}}(\text{EMP\_DEPENDENTS})$$

$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Dependent\_name}}(\text{ACTUAL\_DEPS})$$

Relational Algebra Operations form Set Theory

The CARTESIAN PRODUCT (CROSS PRODUCT) operation.

FEMALE_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1968-07-19	3321Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291Berry, 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

Relational Algebra Operations form Set Theory

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	...

Relational Algebra Operations form Set Theory

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 and DIVISION

- **The JOIN Operation (⋈):**
  - The JOIN operation, denoted by ⋈, is used to combine related tuples from two relations into single "longer" tuples.
  - 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 $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$ is
  - $R \bowtie_{\text{<join condition>}} S$
  - The result of the JOIN is a relation Q with n + m attributes $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ in that order. Q has one tuple for each combination of tuples – one from R and one from S – whenever the combination satisfies the join condition.

Binary Relational Operations: JOIN and DIVISION

- For example, to retrieve the name of the manager of each department, we write

  $DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{Mgr\_ssn = Ssn} EMPLOYEE$

  $RESULT \leftarrow \pi_{Dname, Lname, Fname}(DAPT\_MGR)$

- The JOIN operation can be specified as a CARTESIAN PRODUCT operation followed by a SELECT operation. For example, consider the query with CARTESIAN PRODUCT as given below.

  $EMP\_DEPENDENTS \leftarrow EMPNAMES \times DEPENDENT$

  $ACTUAL\_DEPS \leftarrow \sigma_{Ssn=Essn}(EMP\_DEPENDENTS)$

  These two operations can be replaced with a single JOIN operation as follows:

  $ACTUAL\_DEPS \leftarrow EMPNAMES \bowtie_{Ssn=Essn} DEPENDENT$

Binary Relational Operations: JOIN and DIVISION

Result of the JOIN operation DEPT_MGR ← DEPARTMENT $\bowtie_{Mgr\_ssn=Ssn}$ EMPLOYEE

DEPT_MGR

Dname	Dnumber	Mgr_ssn	...	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	...

- **Difference between CARTESIAN PRODUCT and JOIN.** In JOIN, only combinations of tuples satisfying the join condition appear in the result, whereas in the CARTESIAN PRODUCT all combinations of tuples are included in the result. The join condition is specified on attributes from the two relations R and S and is evaluated for each combination of tuples. Each tuple combination for which the join condition evaluates to TRUE is included in the resulting relation Q as a single combined tuple.

Binary Relational Operations: JOIN and DIVISION

- If no combination of tuples satisfies the join condition, the result of a JOIN is an empty relation with zero tuples. In general, if R has n_R tuples and S has n_S tuples, the result of a JOIN operation $R \bowtie_{<\text{join condition}>} S$ will have between zero and $n_R * n_S$ tuples. The expected size of the join result divided by the maximum size $n_R * n_S$ leads to a ratio called **join selectivity**, which is a property of each join condition. If there is no join condition, all combinations of tuples qualify and the JOIN degenerates into a CARTESIAN PRODUCT.
- A general join condition is of the form <condition> AND <condition> AND … AND <condition> where each <condition> is of the form $A_i \; \theta \; B_j$, A_i is an attribute of R, B_j is an attribute of S, A_i and B_j have the same domain, and θ (theta) is one of the comparison operators $\{=, <, \leq, >, \geq, \neq\}$. A JOIN operation with such a general join condition is called a **THETA JOIN**. Tuples whose join attributes are NULL or for which the join condition is FALSE do not appear in the result.

Binary Relational Operations: JOIN and DIVISION

- **Variations of JOIN: EQUIJOIN**
  - 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. Both previous examples were EQUIJOINs.
  - The result of an EQUIJOIN always have one or more pairs of attributes that have identical values in every tuple.
- **Variations of JOIN: NATURAL JOIN (*)**
  - 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 join attributes) have the same name in both relations. If this is not the case, a renaming operation is applied first.

Binary Relational Operations: JOIN and DIVISION

- To combine each PROJECT tuple with the DEPARTMENT tuple that controls the project, we write

  DEPT ← $\rho_{(Dname, Dnum, Mgr\_ssn, Mgr\_start\_date)}$(DEPARTMENT)

  PROJ_DEPT ← PROJECT * DEPT

- If the attributes on which the natural join is specified already have the same names in both relations, renaming is unnecessary. For example, to apply a natural join on the Dnumber attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write

  DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS

- In general, the join condition for NATURAL JOIN is constructed by equating each pair of join attributes that have the same name in the two relations and combining these conditions with AND.

- JOIN operations are also known as **inner joins**, to distinguish them from a different join variation called **outer joins**.

Binary Relational Operations: JOIN and DIVISION

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

$$DEPT \leftarrow \rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}(DEPARTMENT)$$

$$PROJ\_DEPT \leftarrow PROJECT * DEPT$$

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

$$DEPT\_LOCS \leftarrow DEPARTMENT * DEPT\_LOCATIONS$$

Binary Relational Operations: JOIN and DIVISION

- **Complete Set of Relational Algebra Operations:**
  - The set of relational algebra operations $\{\sigma, \pi, \cup, \rho, -, \times\}$ is a complete set because any of the other original relational algebra operations can be expressed as a sequence of operations from this set. For example,

    $R \cap S \equiv (R \cup S) - ((R - S) \cup (S - R))$

    $R \bowtie_{\text{<join condition>}} S \equiv \sigma_{\text{<join condition>}} (R \times S)$
  - A NATURAL JOIN can also be specified as a CARTESIAN PRODUCT preceded by RENAME and followed by SELECT and PROJECT operations.
  - The complete set of relational algebra operations is sufficient to write any query. However, INTERSCETION and JOIN are important to include because they are convenient to use and are very commonly applied in database applications.

Binary Relational Operations: JOIN and DIVISION

- **The DIVISION Operation (÷):**
  - The DIVISION operation is applied to two relations $R(Z) \div S(X)$, where the attributes of S are a subset of the attributes of R; that is, $X \subseteq Z$. Let Y be the set of attributes of R that are not attributes of S; that is, $Y = Z - X$ (and hence $Z = X \cup Y$).
  - 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. This means that, 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.
  - The DIVISION operation can be expressed as a sequence of π, $\times$, and – operations as follows:

    $T_1 \leftarrow \pi_Y(R)$

    $T_2 \leftarrow \pi_Y((S \times T_1) - R)$

    $T \leftarrow T1 - T2$

Binary Relational Operations: JOIN and DIVISION

- For example, to retrieve the names of employees who work on all the projects that 'John Smith' works on, we write,

  $SMITH \leftarrow \sigma_{Fname='John' \text{ AND } Lname='Smith'}(EMPLOYEE)$

  $SMITH\_PNOS \leftarrow \pi_{Pno}(WORKS\_ON \bowtie_{Essn=Ssn} SMITH)$

  $SSN\_PNOS \leftarrow \pi_{Essn,\ Pno}(WORKS\_ON)$

  $SSNS(Ssn) \leftarrow SSN\_PNOS \div SMITH\_PNOS$

  $RESULT \leftarrow \pi_{Fname,\ Lname}(SSNS * EMPLOYEE)$

Binary Relational Operations: JOIN and DIVISION

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

(a)

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

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

(b)

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

S

A
a1
a2
a3

T

B
b1
b4

Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$
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.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$ 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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Additional Relational Algebra Operations

- Some common database requests cannot be performed with the original relational algebra operations.
- These additional relational algebra operations enhance the expressive power of the original relational algebra.
- **Generalized Projection:**
  - The generalized projection operation extends the projection operation by allowing functions of attributes to be included in the projection list. The generalized form can be expressed as:

    $\pi_{F1, F2, ..., Fn}$ (R)

    where F_1, F_2, …, F_n are functions over the attributes in relation R and may involve arithmetic operations and constant values. For example,

    $\pi_{Fname, Salary * 0.25}$ (EMPLOYEE))

Additional Relational Algebra Operations

- **Aggregate Functions:**
  - Aggregate functions are mathematical functions applied on collections of values. These functions are used in queries that summarize information. These The functions applied to values include **SUM**, **AVERAGE**, **MAXIMUM**, and **MINIMUM**. The **COUNT** function is used for counting tuples or values.
  - We can also group the tuples in a relation by the value of some of their attributes and then applying an aggregate function independently to each group. The general form is:

    <grouping attributes>$\Im$<function list>(R)
  - Where <grouping attributes> is a list of attributes of the relation specified in R, and <function list> is a list of (<function> <attribute>) pairs.
  - Duplicates are not eliminated when an aggregate function is applied. However, NULL values are not considered.

Additional Relational Algebra Operations

The aggregate function operation.

a. $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}({}_{\text{Dno}}\,\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE}))$.

b. ${}_{\text{Dno}}\,\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

c. $\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

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

Additional Relational Algebra Operations

- **Recursive Closure Operation:**
  - Another type of operation that, in general, cannot be specified in the basic original relational algebra is recursive closure.
  - This operation is applied to a recursive relationship, between tuples of same type.
  - Example of a recursive operation is to retrieve all SUPERVISEES of an EMPLOYEE e at all levels – that is, all EMPLOYEE e' directly supervised by e; all employees e'' directly supervised by each employee e'; all employees e''' directly supervised by each employee e''; and so on.
  - Although it is possible to retrieve employees at each level and then take their UNION, we cannot, in general, specify a query at all levels without utilizing a looping mechanism unless we know the maximum number of levels. The SQL3 standard includes syntax for recursive closure.

Additional Relational Algebra Operations

- For example, to specify the Ssns of all employees e′ directly supervised at level one by the employee e whose name is ‘James Borg’, we can apply the following operation:

  $BORG\_SSN \leftarrow \pi_{Ssn}(\sigma_{Fname='James' \text{ AND } Lname='Borg'}(EMPLOYEE))$

  $SUPERVISION(Ssn1, Ssn2) \leftarrow \pi_{Ssn,Super\_ssn}(EMPLOYEE)$

  $RESULT1(Ssn) \leftarrow \pi_{Ssn1}(SUPERVISION \bowtie_{Ssn2=Ssn} BORG\_SSN)$

  To retrieve all employees supervised by Borg at level 2, that is, all employees e″ supervised by some employee e′ who is directly supervised by Borg, we can apply another JOIN to the result of the first query, as follows:

  $RESULT2(Ssn) \leftarrow \pi_{Ssn1}(SUPERVISION \bowtie_{Ssn2=Ssn} RESULT1)$

  To get both sets of employees supervised at levels 1 and 2 by ‘James Borg’, we can apply the UNION as follows:

  $RESULT \leftarrow RESULT2 \cup RESULT1$

Additional Relational Algebra Operations

SUPERVISION

(Borg's Ssn is 888665555)

(Ssn)	(Super_ssn)
Ssn1	Ssn2
123456789	333445555
333445555	888665555
999887777	987654321
987654321	888665555
666884444	333445555
453453453	333445555
987987987	987654321
888665555	null

Figure
A two-level recursive query.

RESULT1

Ssn
333445555
987654321

(Supervised by Borg)

RESULT2

Ssn
123456789
999887777
666884444
453453453
987987987

(Supervised by Borg's subordinates)

RESULT

Ssn
123456789
999887777
666884444
453453453
987987987
333445555
987654321

(RESULT1 ∪ RESULT2)

Additional Relational Algebra Operations

- **OUTER JOIN Operations:**
  - The JOIN operations described earlier match tuples that satisfy the join condition. Hence, tuples without a matching (or related) tuple are eliminated from the JOIN result. Tuples with NULL values in the join attributes are also eliminated. This type of join, where tuples with no match are eliminated, is known as an **inner join**. The join operations we described earlier are all inner joins.
  - A set of operations, called **outer joins**, were developed for the case where the user wants to keep all the tuples in R, or all those in S, or all those in both relations in the result of the JOIN, regardless of whether or not they have matching tuples in the other relation. There are three outer joins: **LEFT OUTER JOIN**, **RIGHT OUTER JOIN**, and **FULL OUTER JOIN**.

Additional Relational Algebra Operations

- The LEFT OUTER JOIN, denoted by R ⟕ S, keeps all the tuples from left relation R in the result of the JOIN, regardless of whether or not they have matching tuples in the other relation. If no matching tuple is found in S, then the attributes of S in the result are filled or padded with NULL values.
- The RIGHT OUTER JOIN, denoted by R ⟖ S, keeps all the tuples from left relation S in the result of the JOIN, regardless of whether or not they have matching tuples in the other relation. If no matching tuple is found in R, then the attributes of R in the result are filled or padded with NULL values.
- The FULL OUTER JOIN, denoted by R ⟗ S, keeps all tuples in both the left and the right relations in the result of the JOIN, regardless of whether or not they have matching tuples in the other relation. If no matching tuple is found, then the attributes in the result are filled or padded with NULL values.

Additional Relational Algebra Operations

- For example, retrieve the name of every employee, and if the employee manages a department, retrieve the name of that department.

  $TEMP \leftarrow (EMPLOYEE \ ⟕_{Ssn=Mgr\_ssn} \ DEPARTMENT)$

  $RESULT \leftarrow \pi_{Fname,\ Minit,\ Lname,\ Dname}(TEMP)$

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

Additional Relational Algebra Operations

- **The OUTER UNION Operations:**
  - The OUTER UNION operation was developed to take the union of tuples from two relations that have some common attributes, but are not union (type) compatible.
  - This operation will take the UNION of tuples in two relations R(X, Y) and S(X, Z) that are partially compatible, meaning that only some of their attributes, say X, are union compatible.
  - The attributes that are union compatible are represented only once in the result, and attributes that are not union compatible from either relation are also kept in the result relation T(X, Y, Z). It is therefore same as a FULL OUTER JOIN on common attributes.
  - Two tuples t_1 in R and t_2 in S are said to match if $t_1[X] = t_2[X]$. These will be combined (unioned) into a single tuple in t. Tuples in either relation that have no matching tuple in the other relation are padded with NULL values.

Additional Relational Algebra Operations

- Example: An outer union of two relations whose schemas are STUDENT(Name, SSN, Department, Advisor) and INSTRUCTOR(Name, SSN, Department, Rank).

  Tuples from the two relations are matched based on having the same combination of values of the shared attributes – Name, SSN, Department.

  If a student is also an instructor, both Advisor and Rank will have a value; otherwise, one of these two attributes will be null.

  All the tuples from both relations are included in the result but tuples with same (name, ssn, department) combination will appear only once in the result.

  The result relation STUDENT_OR_INSTRUCTOR will have the following attributes: STUDENT_OR_INSTRUCTOR (Name, SSN, Department, Advisor, Rank)

Relational Calculus

- Another query language for the formal relational model.
- Based on the branch of mathematics known as first-order predicate logic.
- Two types of relational calculus: **tuple relational calculus** (the SQL language is partially based on this) and **domain relational calculus**.
- Relational calculus is considered to be a **nonprocedural** or declarative language; order of query processing not specified in query. This differs from relational algebra, where we must write a sequence of operations to specify a retrieval request; order of operations is specified; considered to be a **procedural** way of stating a query.

Relational Calculus

- A **relational calculus** expression specifies a query in terms of variables that range over rows (tuples) of the database relations (in **tuple calculus**) or over columns (attributes) of the database relations (in **domain calculus**).
- Query result is a relation.
- No order of operations to specify how to retrieve the query result – specifies only what information the result should contain. This is the main distinguishing feature between relational algebra and relational calculus.

The Tuple Relational Calculus

- Based on specifying a number of tuple variables.
- Each tuple variable usually ranges (loops) over the tuples in a particular database relation (the variable takes as its value any individual tuple from that relation).
- A simple tuple relational calculus query is of the form

  {t | COND(t)}

  where t is a tuple variable and COND (t) is a conditional expression involving t. The result of such a query is the set of all tuples t that satisfy COND (t).
- **Example 1:** To find all employees whose salary is above $50,000, we can write the following tuple calculus expression: **{t | EMPLOYEE(t) AND t.Salary > 50000}**

The Tuple Relational Calculus

The condition EMPLOYEE(t) specifies that the range relation of tuple variable t is EMPLOYEE. Each EMPLOYEE tuple t that satisfies the condition t.Salary>50000 will be retrieved. Notice that t.Salary references attribute Salary of tuple variable t.

- **Example 2:** To retrieve only some of the attributes - say, the first and last names – we write

  {t.Fname, t.Lname | EMPLOYEE(t) AND t.Salary > 50000}

- **Example 3:** Retrieve the department name and manager last name for each department: **{e.Lname, d.Dname | EMPLOYEE(e) AND DEPARTMENT(d) AND d.Mgr_ssn = e.ssn}**

The Tuple Relational Calculus

- Two special symbols called quantifiers can appear in relational calculus formulas (conditions); these are the universal quantifier (∀) and the existential quantifier (∃).
- Informally, a tuple variable t is bound if it is quantified, meaning that it appears in an (∀ t) or (∃ t) clause; otherwise, it is free.
- If F is a formula (boolean condition), then so are (∃ t)(F) and (∀ t)(F), where t is a tuple variable.
  - The formula (∃ t)(F) is true if the formula F evaluates to true for some (at least one) tuple assigned to free occurrences of t in F; otherwise (∃ t)(F) is false.
  - The formula (∀ t)(F) is true if the formula F evaluates to true for every tuple (in the "universe") assigned to free occurrences of t in F; otherwise (∀ t)(F) is false.

The Tuple Relational Calculus

- **Example 4:** Retrieve the name and address of all employees who work for the 'Research' department. The query can be expressed as :

  {t.FNAME, t.LNAME, t.ADDRESS | EMPLOYEE(t) and (∃ d) (DEPARTMENT(d) and d.DNAME='Research' and d.DNUMBER=t.DNO) }
- If a tuple satisfies the conditions specified in the query, the attributes FNAME, LNAME, and ADDRESS are retrieved for each such tuple.
- The conditions EMPLOYEE (t) and DEPARTMENT(d) specify the range relations for t and d.
- Tuple t satisfying the conditions d.DNAME = 'Research' and d.DNUMBER = t.DNO will be retrieved.

The Domain Relational Calculus

- Another variation of relational calculus called the domain relational calculus, or simply, domain calculus is equivalent to tuple calculus and to relational algebra.
- Domain calculus differs from tuple calculus in the type of variables used in formulas:
- Rather than having variables range over tuples, the variables range over single values from domains of attributes.
- To form a relation of degree n for a query result, we must have n of these domain variables— one for each attribute.
- An expression of the domain calculus is of the form: $\{x_1, x_2, \ldots, x_n \mid \text{COND}(x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m})\}$
  where $x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m}$ are domain variables that range over domains (of attributes) and COND is a condition or formula of the domain relational calculus.

The Domain Relational Calculus

- Retrieve the birthdate and address of the employee whose name is ‘John B. Smith’.

  {uv | (∃ q) (∃ r) (∃ s) (∃ t) (∃ w) (∃ x) (∃ y) (∃ z) (EMPLOYEE(qrstuvwxyz) and q=’John’ and r=’B’ and s=’Smith’)}

  Ten variables for the employee relation are needed, one to range over the domain of each attribute in order, EMPLOYEE(qrstuvwxyz).

  Specify the requested attributes, BDATE and ADDRESS, by the domain variables u for BDATE and v for ADDRESS.

  Specify the condition for selecting a tuple following the bar (|) – namely, that the sequence of values assigned to the variables qrstuvwxyz be a tuple of the employee relation and that the values for q (FNAME), r (MINIT), and s (LNAME) be ‘John’, ‘B’, and ‘Smith’, respectively.