Relational Algebra 1

Unit 3.3

Relational Algebra

Relational Algebra is :

- the formal description of how a relational database operates
- the mathematics which underpin SQL operations.

Operators in relational algebra are not necessarily the same as SQL operators, even if they have the same name.

Terminology

- Relation - a set of tuples.
- Tuple - a collection of attributes which describe some real world entity.
- Attribute - a real world role played by a named domain.
- Domain - a set of atomic values.
- Set - a mathematical definition for a collection of objects which contains no duplicates.

Operators - Write

- INSERT - provides a list of attribute values for a new tuple in a relation. This operator is the same as SQL.
- DELETE - provides a condition on the attributes of a relation to determine which tuple(s) to remove from the relation. This operator is the same as SQL.
- MODIFY - changes the values of one or more attributes in one or more tuples of a relation, as identified by a condition operating on the attributes of the relation. This is equivalent to SQL UPDATE.

Operators - Retrieval

There are two groups of operations:

- Mathematical set theory based relations:
  UNION, INTERSECTION, DIFFERENCE, and CARTESIAN PRODUCT.
- Special database operations:
  SELECT (not the same as SQL SELECT), PROJECT, and JOIN.

Relational SELECT

SELECT is used to obtain a subset of the tuples of a relation that satisfy a *select condition*.

For example, find all employees born after 1st Jan 1950:

$\text{SELECT}_{\text{dob} > \text{'01/JAN/1950'}}\text{(employee)}$

Relational PROJECT

The PROJECT operation is used to select a subset of the attributes of a relation by specifying the names of the required attributes.

For example, to get a list of all employees surnames and employee numbers:

$$\text{PROJECT}_{\text{surname,empno}}\ (\text{employee})$$

SELECT and PROJECT

SELECT and PROJECT can be combined together. For example, to get a list of employee numbers for employees in department number 1:

$$\text{PROJECT}_{\text{empno}} \; (\text{SELECT}_{\text{depno} = 1} \; (\text{employee}))$$

Mapping this back to SQL gives:

```
SELECT empno
FROM   employee
WHERE  depno = 1;
```

Set Operations - semantics

Consider two relations R and S.

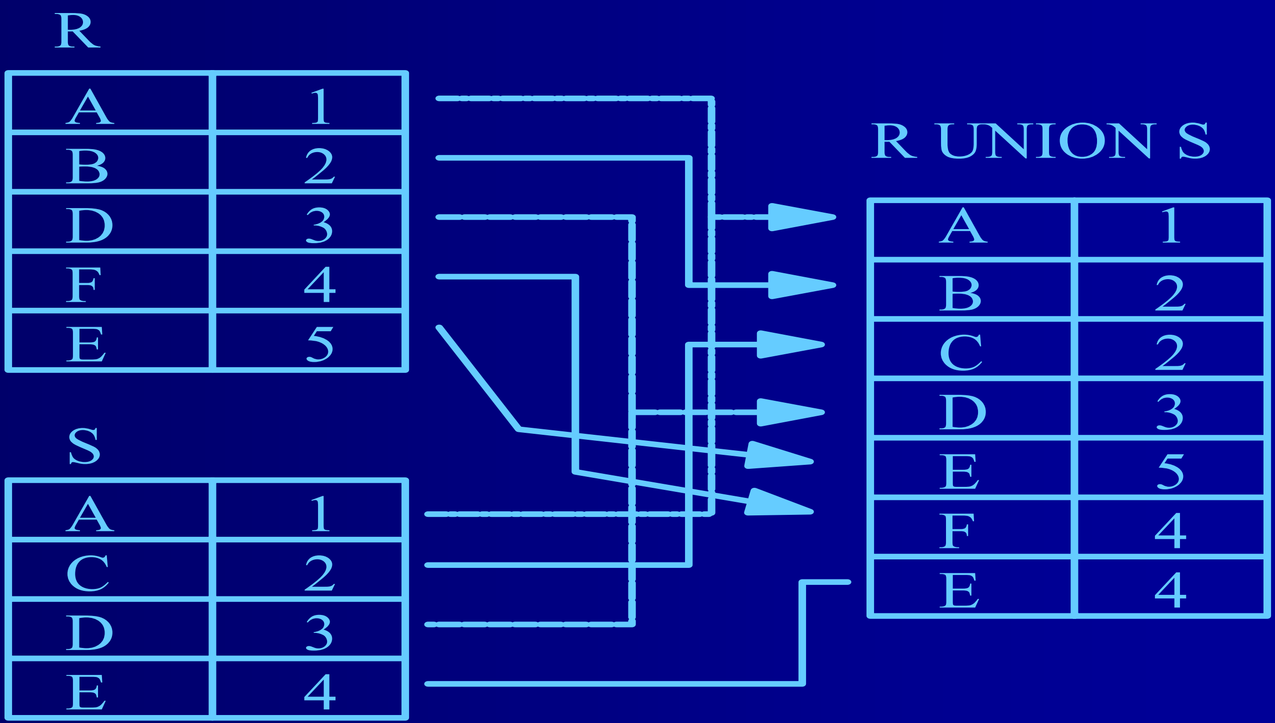
- UNION of R and S
  the union of two relations is a relation that includes all the tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.
- INTERSECTION of R and S
  the intersection of R and S is a relation that includes all tuples that are both in R and S.
- DIFFERENCE of R and S
  the difference of R and S is the relation that contains all the tuples that are in R but that are not in S.

SET Operations - requirements

For set operations to function correctly the relations R and S must be union compatible. Two relations are union compatible if

- they have the same number of attributes
- the domain of each attribute in column order is the same in both R and S.

UNION Example


R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R UNION S

A	1
B	2
C	2
D	3
E	5
F	4
E	4

INTERSECTION Example

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R |NTERSECTION S

A	1
D	3

DIFFERENCE Example

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R DIFFERENCE S

B	2
F	4
E	5

S DIFFERENCE R

C	2
E	4

CARTESIAN PRODUCT

The Cartesian Product is also an operator which works on two sets. It is sometimes called the CROSS PRODUCT or CROSS JOIN.

It combines the tuples of one relation with all the tuples of the other relation.

CARTESIAN PRODUCT Example

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R CROSS S

A	1	A	1
A	1	C	2
A	1	D	3
A	1	E	4
B	2	A	1
B	2	C	2
B	2	D	3
B	2	E	4
D	3	A	1
D	3	C	2
D	3	D	3
D	3	E	4
F	4	A	1
F	4	C	2
F	4	D	3
F	4	E	4
E	5	A	1
E	5	C	2
E	5	D	3
E	5	E	4

JOIN Operator

JOIN is used to combine related tuples from two relations:

- In its simplest form the JOIN operator is just the cross product of the two relations.
- As the join becomes more complex, tuples are removed within the cross product to make the result of the join more meaningful.
- JOIN allows you to evaluate a join condition between the attributes of the relations on which the join is undertaken.

The notation used is

$R \text{ JOIN}_{\text{join condition}} S$

JOIN Example

R

ColA	*ColB*
A	1
B	2
D	3
F	4
E	5

S

SColA	*SColB*
A	1
C	2
D	3
E	4

$R\ \mathrm{JOIN}_{R.ColA\ =\ S.SColA}\ S$

A	1	A	1
D	3	D	3
E	5	E	4

$R\ \mathrm{JOIN}_{R.ColB\ =\ S.SColB}\ S$

A	1	A	1
B	2	C	2
D	3	D	3
F	4	E	4

Natural Join

Invariably the JOIN involves an equality test, and thus is often described as an equi-join. Such joins result in two attributes in the resulting relation having exactly the same value. A ‘natural join’ will remove the duplicate attribute(s).

- In most systems a natural join will require that the attributes have the same name to identify the attribute(s) to be used in the join. This may require a renaming mechanism.
- If you do use natural joins make sure that the relations do not have two attributes with the same name by accident.

OUTER JOINs

Notice that much of the data is lost when applying a join to two relations. In some cases this lost data might hold useful information. An outer join retains the information that would have been lost from the tables, replacing missing data with nulls.

There are three forms of the outer join, depending on which data is to be kept.

- LEFT OUTER JOIN - keep data from the left-hand table
- RIGHT OUTER JOIN - keep data from the right-hand table
- FULL OUTER JOIN - keep data from both tables

OUTER JOIN Example 1

R

ColA	*ColB*
A	1
B	2
D	3
F	4
E	5

R LEFT OUTER JOIN $_{\text{R.ColA = S.SColA}}$ S

A	1	A	1
D	3	D	3
E	5	E	4
B	2	-	-
F	4	-	-

S

SColA	*SColB*
A	1
C	2
D	3
E	4

R RIGHT OUTER JOIN $_{\text{R.ColA = S.SColA}}$ S

A	1	A	1
D	3	D	3
E	5	E	4
-	-	C	2

OUTER JOIN Example 2

R

ColA	*ColB*
A	1
B	2
D	3
F	4
E	5

S

SColA	*SColB*
A	1
C	2
D	3
E	4

R FULL OUTER JOIN$_{R.ColA = S.SColA}$ S

A	1	A	1
D	3	D	3
E	5	E	4
B	2	-	-
F	4	-	-
-	-	C	2