Department of UIC

Database Management Systems

Pawandeep Sharma

Assistant Prof., UIC

Syllabus

Unit-I

Introduction: Overview of Database Management System: Various views of data Models, Schemes and Introduction to database Languages & Environments, Advantages of DBMS over file processing systems, Responsibility of Database Administrator. Three level architecture of Database Systems: Introduction to client/Server architecture.

Syllabus

Unit-II

Data Models: E-R Diagram (Entity Relationship), mapping Constraints, keys, Reduction of E-R diagram into tables. Network & Hierarchical Models, File Organization: Sequential File, index sequential files, direct files, Hashing, B-trees Index files, Inverted Lists., Relational Models, Relational Algebra & various operations (set operations, select, project, join, division), Order, Relational calculus: Domain, Tuple, Well Formed Formula, specification, quantifiers, Introduction to Query Language, QBE.

Syllabus

Unit-III

Integrity constrains, functional dependencies & Normalization, 1st, 2nd, 3rd and BCNF.

Introduction to Distributed Data processing, Concurrency control: Transactions, Time stamping, Lock-based Protocols.

Reference Books

- Fundamentals of Database Systems by R.Elmasri and S.B.Navathe, 3rd Edition, Pearson Education, New Delhi.
- An Introduction to Database Systems by C.J. Date, 7th Edition, Pearson Education, New Delhi.
- A Guide to the SQL Standard, Data, C. and Darwen, H.3rd Edition, Reading, Addison-Wesley Publications, New Delhi.
- Introduction to Database Management system by Bipin Desai, Galgotia Pub, New Delhi.
- Database System Concepts by A. Silberschatz, H.F.Korth and S.Sudarshan, 3rd Edition, McGraw-Hill, International Edition.
- SQL / PL/SQL, by Ivan Bayross, BPB Publications.

Relational Algebra

Contents

What is an "Algebra"

- Mathematical system consisting of:
  - *Operands* --- variables or values from which new values can be constructed.
  - *Operators* --- symbols denoting procedures that construct new values from given values.

What is Relational Algebra?

- An algebra whose operands are relations or variables that represent relations.
- Operators are used to perform operations given by users in the form of retrieval requests on relations in the database.

Basic terms in Relational Algebra

STUDENT

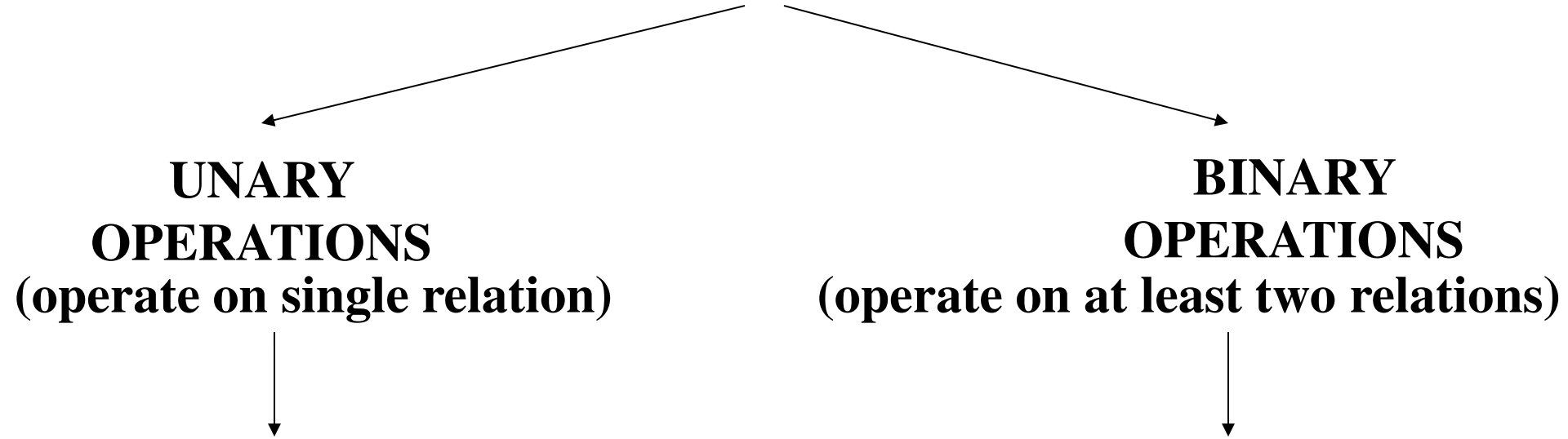
Roll No.	Name	Class	Year
1029	Komal	MCA	2
1059	Ajay	MCA	2
1019	Brijesh	MCA	3

ATTRIBUTE

TUPLE

COLUMN

Relational Algebra Operations



UNARY OPERATIONS
(operate on single relation)

* SELECT
* PROJECT

BINARY OPERATIONS
(operate on at least two relations)

- UNION
- INTERSACTION
- SET DIFFERENCE
- CROSS PRODUCT
- JOIN

Selection

- SELECT operation is use to select a subset of the tuples from a relation that satisfy a selection condition.

Syntax:

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

Selection Symbol

Relation Name

Example

Student

Roll No.	Name	Age	Marks
1	A	20	75
2	B	22	55
3	C	21	70
4	D	22	70

$$\text{Result} \leftarrow \sigma_{\text{age} = 22}(\text{Student})$$

Result

Roll No.	Name	Age	Marks
2	B	22	55
4	D	22	70

Features of Selection

- ✓ Selection operation is applied tuple individually.
- ✓ Degree of relation resulting from SELECT operation is same as degree of relation R.
- ✓ Number of tuples in result relation is less or equal to relation R.
- ✓ SELECT operation is Commutative.
- ✓ Sequence of SELECTs can be applied in any order.

Projection

- PROJECT operation is use to selects certain columns from the relation and discard the unselected columns.

Syntax:

Example

Student

RollNo	Name	Age	Marks
1	A	20	75
2	B	22	55
3	C	21	70
4	D	22	70

$$\text{Result} \leftarrow \prod_{\text{RollNo,Name,Marks}}(\text{Student})$$

Result

RollNo	Name	Marks
1	A	75
2	B	55
3	C	70
4	D	70

Features of Projection

- ✓ Resultant relation has attributes specified in <attribute list> and in the same order.
- ✓ Resultant relation has degree equal to no. of attributes in <attribute list>
- ✓ PROJECT operation removes duplicate tuples.
- ✓ No. of tuples in Result is always less than or equal to the no. of tuples in R.
- ✓ Commutativity does not hold on PROJECT.

Mathematical Set Theory Operation

- Mathematical set theory operations include UNION, INTERSECTION, SET difference (MINUS) and CARTESIAN PRODUCT.

- All the above operations are Binary operations and applied on two relations.

- UNION,INTERSECTION,MINUS operations required two Union Compatible sets.

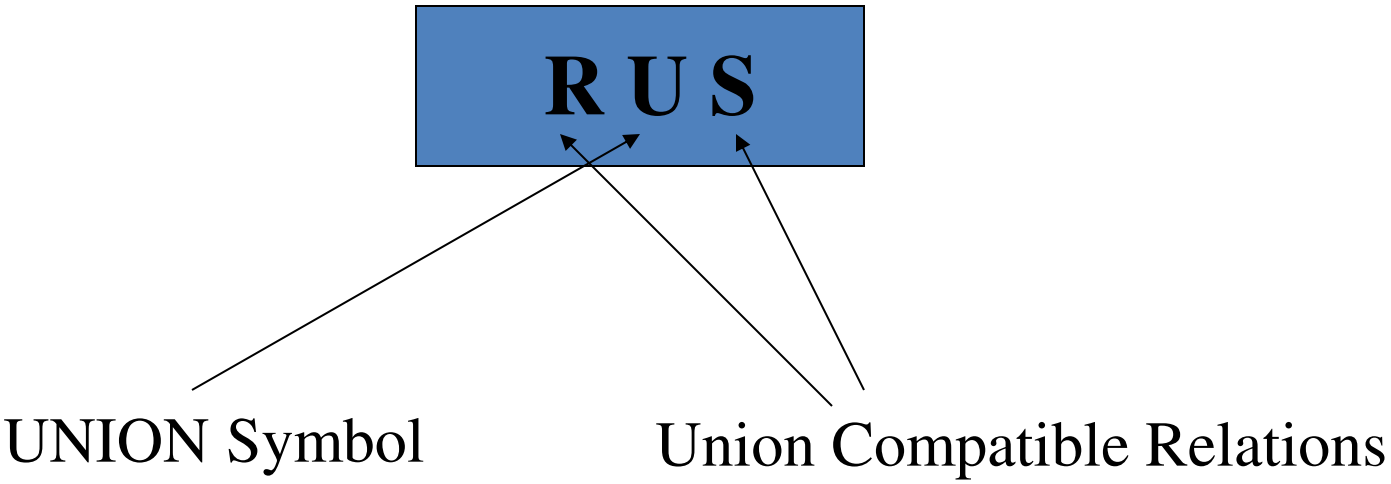
What is Union Compatibility?

- Relations R and S has same type of tuples.
- Relations R and S has same degree I.e. same no. of attributes.
- Each corresponding attributes has same domain.

UNION

- The result of union operation is a relation that includes all tuples that are either in R or in S or in both R & S.

Syntax:



Example

A

Name	Age
Neeraj	23
Sandeep	23
Gaurav	22
Ajay	22

B

Sname	Age
Sandeep	23
Ajay	22
Rahul	23

RESULT ← A U B

RESULT

Name	Age
Neeraj	23
Sandeep	23
Gaurav	22
Ajay	22
Rahul	23

Features of UNION

- ✓ Duplicate tuples are eliminated.
- ✓ By Default resulting relation has same attribute names as in the first relation.
- ✓ UNION is Commutative operation.

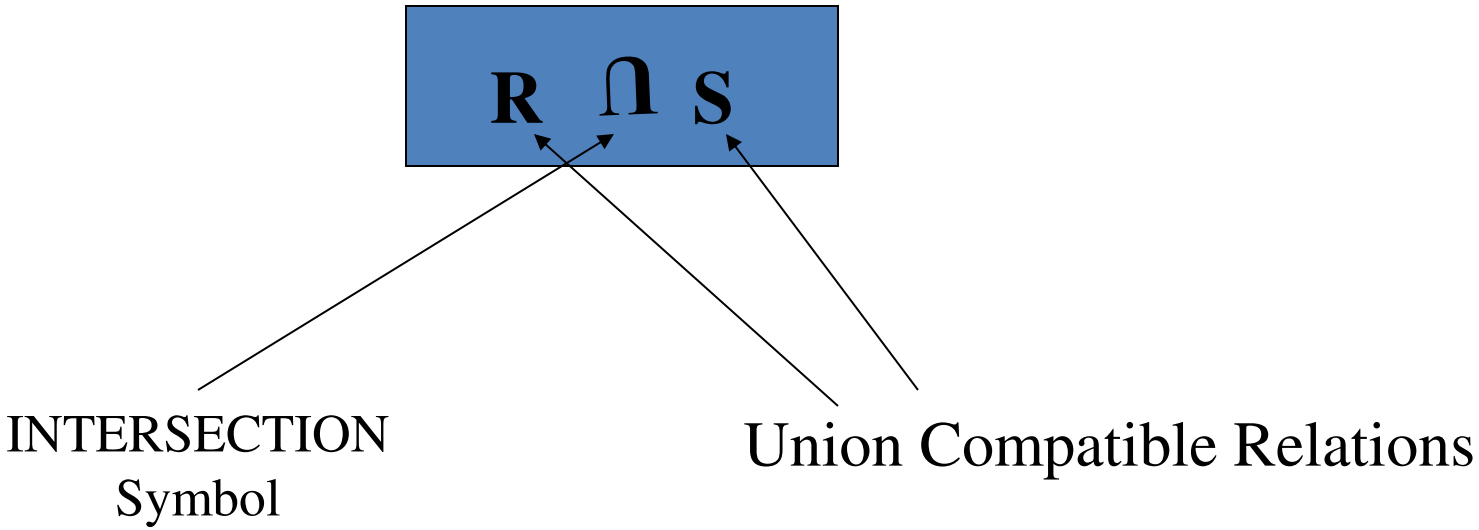
  R U S = S U R
- ✓ UNION is also Associative operation.

  R U (S U T) = (R U S) U T

INTERSECTION

- The result of intersection operation is a relation that includes all tuples that are in both R and S.

Syntax:



Example

A

Name	Age
Neeraj	23
Sandeep	23
Gaurav	22
Ajay	22

B

Sname	Age
Sandeep	23
Ajay	22
Rahul	23

$$RES \leftarrow A \cap B$$

RES

Name	Age
Sandeep	23
Ajay	22

Features of INTERSECTION

- ✓ If both relations are not even having a single similar tuple than resulting relation is empty relation.

- ✓ INTERSECTION is Commutative operation.

$$R \cap S = S \cap R$$

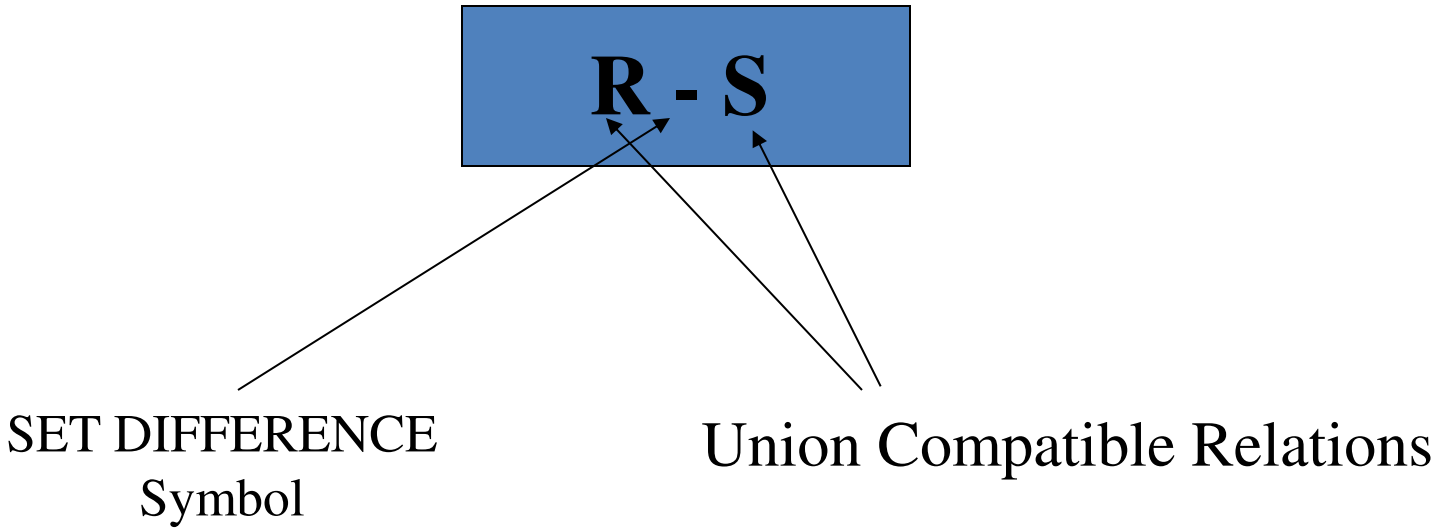
- ✓ INTERSECTION is also Associative operation.

$$(R \cap S) \cap T = R \cap (S \cap T)$$

SET DIFFERENCE

- The result of MINUS operation is a relation that includes all tuples that are in R but not in S.

Syntax:



Example

STU1

Name	Age
Neeraj	23
Sandeep	23
Gaurav	22
Ajay	22

STU2

Sname	Age
Sandeep	23
Ajay	22
Rahul	23

R ← STU1 - STU2

S ← STU2 – STU1

R

Name	Age
Neeraj	23
Gaurav	22

S

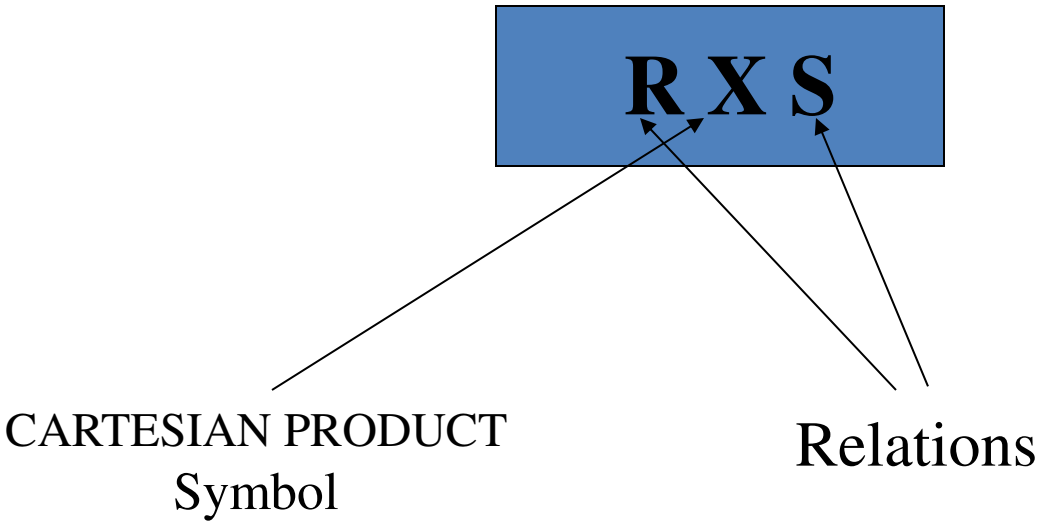
Name	Age
Rahul	23

✓ MINUS operation is not Commutative I.e. $R \neq S$

CARTESIAN PRODUCT

- Cartesian Product is used to combine the tuples from two relations in a combine fashion.
- The resulting relation has one tuple for each combination of tuples – one from R and one from S.

Syntax:



Example

MCASTU

Name	Class
Neeraj	MCA
Gaurav	MCA

SUB

Subject	Faculty
DBMS	CC
VB	KC
VC++	VK

RESULT ← MCASTU X SUB

RESULT

Name	Class	Subject	Faculty
Neeraj	MCA	DBMS	CC
Neeraj	MCA	VB	KC
Neeraj	MCA	VC++	VK
Gaurav	MCA	DBMS	CC
Gaurav	MCA	VB	KC
Gaurav	MCA	VC++	VK

Features of CARTESIAN PRODUCT

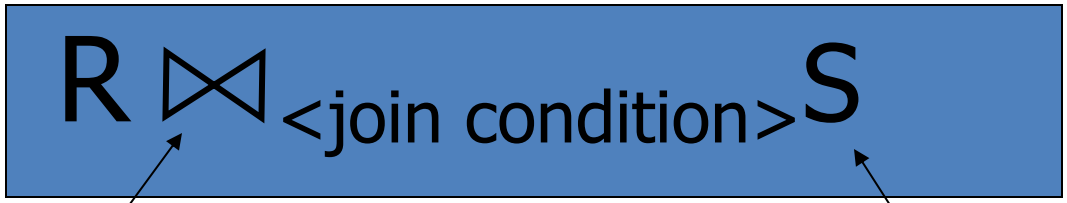
- ✓ It also refers as CROSS PRODUCT or CROSS JOIN.
- ✓ It is a Binary set operation, relations need not be union compatible.
- ✓ If one relation say R is of degree n and another relation S is of degree m then,

  Resultant relation obtained is of degree n+m

JOIN Operation

- JOIN operation is used to combine related tuples from two relations into single tuples.

Syntax:



Join Symbol

Relation Name

Types of JOIN

1.INNER JOIN

- 1.1 THETA JOIN
- 1.2 EQUI JOIN
- 1.3 NATURAL JOIN

2. OUTER JOIN

- 2.1 LEFT OUTER JOIN
- 2.2 RIGHT OUTER JOIN
- 2.3 FULL OUTER JOIN

3. SEMI JOIN

4. ANTI JOIN

Theta-Join

- Produces all combinations of tuples from R1 and R2 that satisfy the join condition.
- $R3 \leftarrow R1 \bowtie_C R2$
- As for join condition, *C* can be any Boolean-valued condition like:
  - A θ B, where θ can be (=,<,<=,>,>=,<>)

    (where A is attribute of R1 & B is attribute of R2).

Example

STU

Name	Marks	Subject
Neeraj	75	DBMS
Gaurav	25	VB
Neeraj	34	DBMS
Ajay	65	OS

RCHECK

Result	PMark
Pass	35
pass	98

Final ← STU  STU.Marka >= RCHECK.PMark RCHECK

Final

Name	Marks	Subject	Result	PMark
Neeraj	75	DBMS	Pass	35
Ajay	65	OS	Pass	35

Equi Join

- The most common join involves the join condition with equality comparison only.
- So in join if comparison operator used is '=' then it is called Equi Join.
- In Equi join resultant relation have one or more pair of attributes that have identical values in each tuple.

Example

STU

Name	Subject	Marks
Shweta	DBMS	70
Shweta	VB	50
Komal	DBMS	65
Komal	OS	75

ADDRESS

Sname	Place
Shweta	Faridabad
Komal	Karnal

Result ← STU ⋈ STU.Name = ADDRESS.Sname ADDRESS

Result

Name	Subject	Marks	Sname	Place
Shweta	DBMS	70	Shweta	Faridabad
Shweta	VB	50	Shweta	Faridabad
Komal	DBMS	65	Komal	Karnal
Komal	OS	75	Komal	Karnal

Natural Join

- A frequent type of join connects two relations by:
  - Equating attributes of the same name, and
  - Projecting out one copy of each pair of equated attributes.
- Called *natural* join.
- Denoted R3 ← R1 * R2.

Example

STU

Name	Subject	Marks
Shweta	DBMS	70
Shweta	VB	50
Komal	DBMS	65
Komal	OS	75

ADDRESS

Name	Subject	Address
Shweta	DBMS	Faridabad
Shweta	VB	Faridabad
Komal	DBMS	Karnal
Komal	OS	Karnal

Result ← STU * ADDRESS

Result

Name	Subject	Marks	Place
Shweta	DBMS	70	Faridabad
Shweta	VB	50	Faridabad
Komal	DBMS	65	Karnal
Komal	OS	75	Karnal

Outer Join

- The result of a join (or inner join) consists of tuples formed by combining matching tuples in the two relations.
- An **outer join** contains those tuples and additionally some tuples formed by extending an unmatched tuple in one of the relations by "fill" values for each of the attributes of the other relation.
- NULL is used as “fill” value.

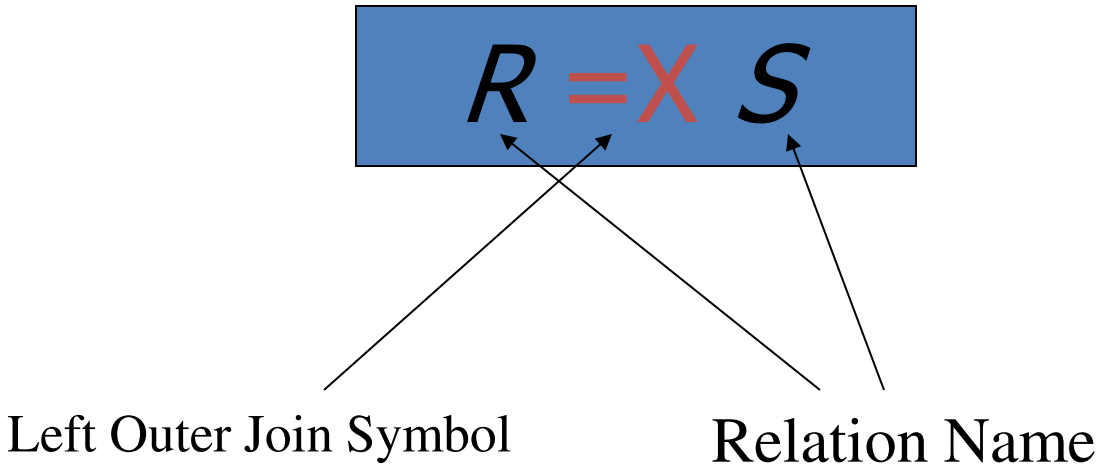
Types of Outer Join

- ✓ Left Outer Join
- ✓ Right Outer Join
- ✓ Full Outer Join

Left Outer Join

- The result of the left outer join is the set of all combinations of tuples in *R* and *S* that are equal on their common attribute names, in addition to tuples in *R* that have no matching tuples in *S*.

Syntax:



Example

STU

Name	RollNo	Subject
Neeraj	1009	PP
Brijesh	1019	DBMS
Komal	1029	PP
Gaurav	1039	DBMS
Sweta	1049	OS

FACULTY

Subject	FName
DBMS	CC
VC++	VK

Result ← STU =X FACULTY

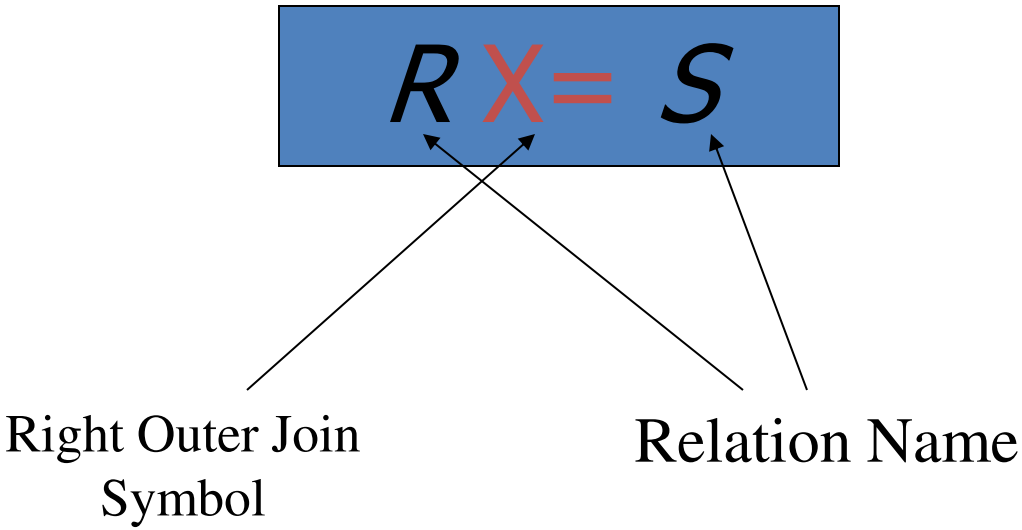
Result

Name	RollNo	Subject	FName
Neeraj	1009	PP	ø
Brijesh	1019	DBMS	CC
Komal	1029	PP	ø
Gaurav	1039	DBMS	CC
Sweta	1049	OS	ø

Right Outer Join

- The result of the right outer join is the set of all combinations of tuples in *R* and *S* that are equal on their common attribute names, in addition to tuples in S that have no matching tuples in R.

Syntax:



Example

STU

Name	RollNo	Subject
Neeraj	1009	PP
Brijesh	1019	DBMS
Komal	1029	PP
Gaurav	1039	DBMS
Sweta	1049	OS

FACULTY

Subject	FName
DBMS	CC
VC++	VK

Result ← STU X= FACULTY

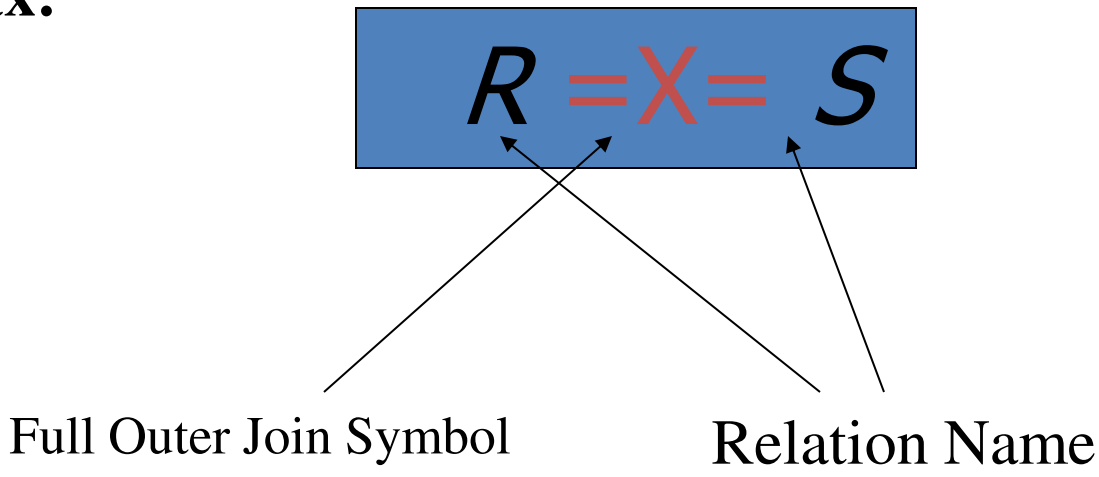
Result

Name	RollNo	Subject	FName
Brijesh	1019	DBMS	CC
Gaurav	1039	DBMS	CC
ø	ø	VC++	VK

Full Outer Join

- The result of the full outer join is the set of all combinations of tuples in *R* and *S* that are equal on their common attribute names, in addition to tuples in *S* that have no matching tuples in *R* and tuples in *R* that have no matching tuples in *S* in their common attribute names.

Syntax:



Example

STU

Name	RollNo	Subject
Neeraj	1009	PP
Brijesh	1019	DBMS
Komal	1029	PP
Gaurav	1039	DBMS
Sweta	1049	OS

FACULTY

Subject	FName
DBMS	CC
VC++	VK

Result ← STU =X= FACULTY

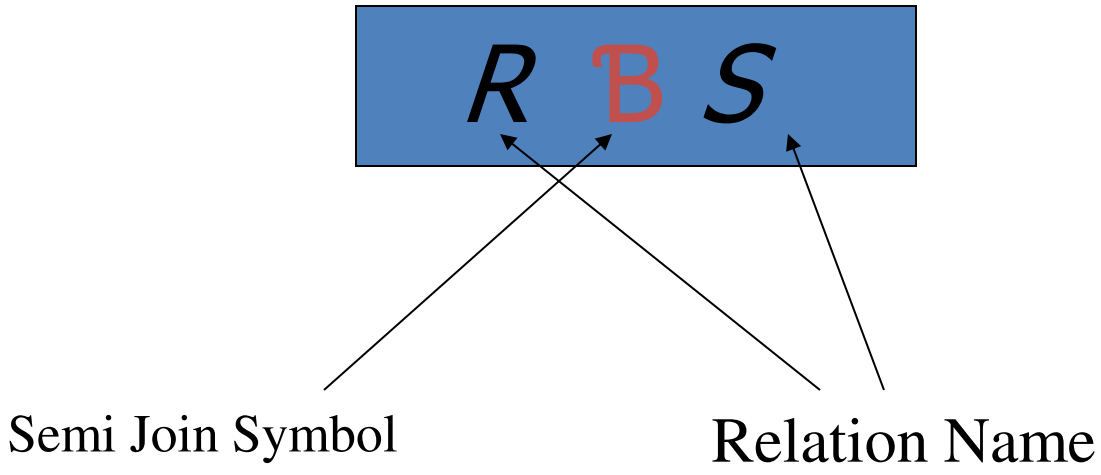
Result

Name	RollNo	Subject	FName
Neeraj	1009	PP	ø
Brijesh	1019	DBMS	CC
Komal	1029	PP	ø
Gaurav	1039	DBMS	CC
Sweta	1049	OS	ø
ø	ø	VC++	VK

Semi Join

- The result of the semijoin is only the set of all tuples in *R* for which there is a tuple in *S* that is equal on their common attribute names.

Syntax:



Example

STU

Name	RollNo	Subject
Neeraj	1009	PP
Komal	1029	DBMS
Gaurav	1039	PP
Shweta	1049	DBMS

FACULTY

Subject	FName
DBMS	CC
VC++	VK

Result ← STU ℬ FACULTY

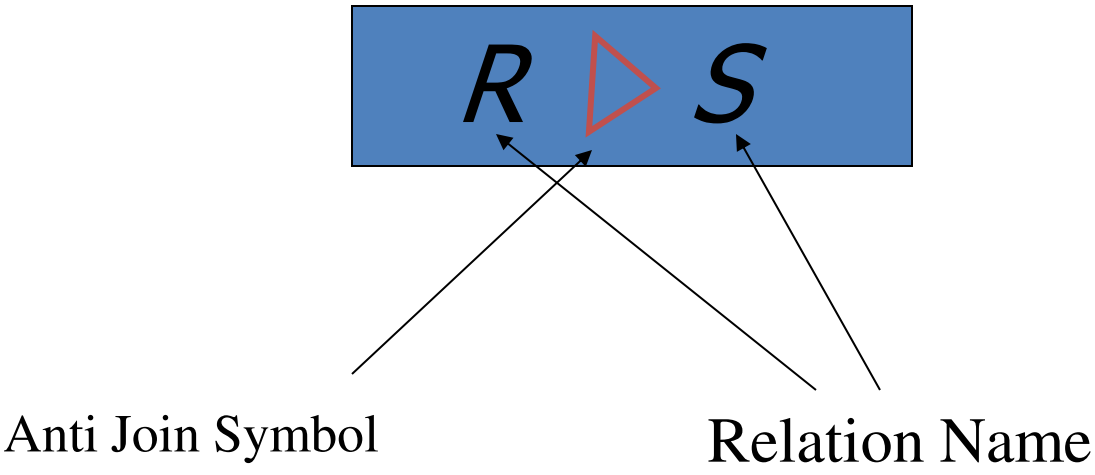
Result

Name	RollNo	Subject
Komal	1029	DBMS
Shweta	1049	DBMS

Anti Join

- The result of an anti-join is only those tuples in *R* for which there is NOT a tuple in *S* that is equal on their common attribute names.

Syntax:



Example

STU

Name	RollNo	Subject
Neeraj	1009	PP
Komal	1029	DBMS
Gaurav	1039	PP
Sweta	1049	DBMS

FACULTY

Subject	FName
DBMS	CC
VC++	VK

Result ← STU ▷ FACULTY

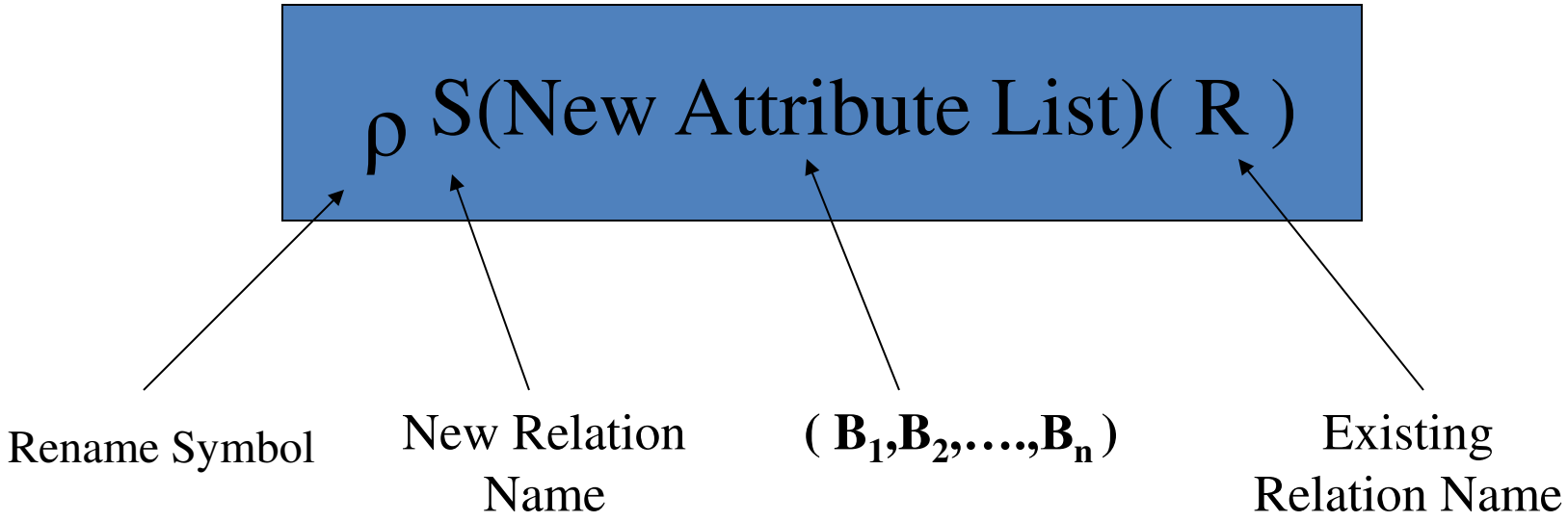
Result

Name	RollNo	Subject
Neeraj	1009	PP
Gaurav	1039	PP

RENAME

➢The RENAME operation gives a new schema to a relation.

Syntax:



- Simplified notation: $S(B_1,B_2,\ldots,B_n) \leftarrow R$

Example

STU

name	rno
Neeraj	1009
Gaurav	1039
Ajay	1059

ρ (StuName,RollNo)(STU)

STU

StuName	RollNo
Neeraj	1009
Gaurav	1039
Ajay	1059

ρ STUDENT(STU)

STUDENT

name	rno
Neeraj	1009
Gaurav	1039
Ajay	1059

ρ STUDENT(StuName,RollNo)(STU)

or

STUDENT(StuName,RollNo) $\leftarrow$ STU

STUDENT

StuName	RollNo
Neeraj	1009
Gaurav	1039
Ajay	1059

QUERIES ??

THANK YOU

UNIVERSITY INSTITUTE of COMPUTING

Asia's Fastest Growing University

Database Management System
CAT -202

Design By:
Prof. Pawandeep Sharma
Assistant Professor
Chandigarh University-Gharuan

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Syllabus

UNIT-I

- **Introduction:** Overview of Database Management System: Various views of data Models, Schemes and Introduction to database Languages & Environments, Advantages of DBMS over file processing systems, Responsibility of Database Administrator. Three level architecture of Database Systems: Introduction to client/Server architecture.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Syllabus

UNIT-II

- **Data Models:** E-R Diagram (Entity Relationship), mapping Constraints, keys, Reduction of E-R diagram into tables. Network & Hierarchical Models,
- **File Organization:** Sequential File, index sequential files, direct files, Hashing, B-trees Index files, Inverted Lists., Relational Model.
- **Relational Algebra:** Meaning & various operations (set operations, select, project, join, division), Order
- **Relational calculus:** Domain, Tuple, Well Formed Formula, specification, quantifiers, Introduction to Query Language, QBE.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Syllabus

UNIT-III

- Integrity constrains, functional dependencies & Normalization, 1st, 2nd, 3rd and BCNF.
- Introduction to Distributed Data processing, Concurrency control: Transactions, Time stamping, Lock-based Protocols.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Reference Books

- Fundamentals of Database Systems by R.Elmasri and S.B.Navathe, 3rd Edition, Pearson Education, New Delhi.
- An Introduction to Database Systems by C.J. Date, 7th Edition, Pearson Education, New Delhi.
- A Guide to the SQL Standard, Data, C. and Darwen, H.3rd Edition, Reading, Addison-Wesley Publications, New Delhi.
- Introduction to Database Management system by Bipin Desai, Galgotia Pub, New Delhi.
- Database System Concepts by A. Silberschatz, H.F.Korth and S.Sudarshan, 3rd Edition, McGraw-Hill, International Edition.
- SQL / PL/SQL, by Ivan Bayross, BPB Publications.

Relational Calculus

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Relational Calculus

- A **relational calculus** expression creates a new relation, which is specified in terms of variables that range over rows of the stored database relations (in **tuple calculus**) or over columns of the stored relations (in **domain calculus**).

- In a calculus expression, there is *no order of operations* to specify how to retrieve the query result—a calculus expression specifies only what information the result should contain. This is the main distinguishing feature between relational algebra and relational calculus.

- Relational calculus is considered to be a **nonprocedural** language. This differs from relational algebra, where we must write a *sequence of operations* to specify a retrieval request; hence relational algebra can be considered as a **procedural** way of stating a query.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Types

- Comes in two forms: *Tuple relational calculus* (TRC) and *Domain relational calculus* (DRC).
- Calculus has *variables, constants, comparison ops*, *logical connectives* and *quantifiers*.
  - *TRC*: Variables range over (i.e., get bound to) *tuples*.
    - Like SQL.
  - *DRC*: Variables range over *domain elements* (= field values).
    - Like Query-By-Example (QBE)
  - Both TRC and DRC are simple subsets of first-order logic.
- Expressions in the calculus are called *formulas.*
- Answer tuple is an assignment of constants to variables that make the formula evaluate to *true*.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Tuple relational calculus

The tuple relational calculus is based on specifying a number of **tuple variables.** Each tuple variable usually *ranges over* a particular database relation, meaning that the variable may take 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).

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Tuple relational calculus

Example: To find the first and last names of all employees whose salary is above $50,000, we can write the following tuple calculus expression:

{t.FNAME, t.LNAME | EMPLOYEE(t) AND t.SALARY>50000}

The condition EMPLOYEE(t) specifies that the **range relation** of tuple variable t is EMPLOYEE.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Quantifiers

Two special symbols called **quantifiers** can appear in formulas; these are :

- **universal quantifier** (∀)
- **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.**

Languages Based on TRC

- The language **SQL** is based on tuple calculus. It uses the basic

  SELECT <list of attributes>

  FROM <list of relations>

  WHERE <conditions>

  block structure to express the queries in tuple calculus where the SELECT clause mentions the attributes being projected, the FROM clause mentions the relations needed in the query, and the WHERE clause mentions the selection as well as the join conditions.

- Another language which is based on tuple calculus is **QUEL** which actually uses the range variables as in tuple calculus.

  Its syntax includes:

  RANGE OF <variable name> IS <relation name>

  Then it uses

  RETRIEVE <list of attributes from range variables>

  WHERE <conditions>

  This language was proposed in the relational DBMS INGRES.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Well Formed Formula

- A general expression of the tuple relational calculus is of following form :
- {t1.A1,t2.A2…..tn.An|COND(t1,t2,t3…tn)}
- Where t1,t2….tn are tuple variables each Ai is an attribute of the relation on which ti ranges and COND is a condition or formula of the tuple relational calculus.
- Every condition is a Well Formed Formula (WFF).

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Domain Relational Calculus

- Another variation of relational calculus called the domain relational calculus, or simply, **domain calculus** .
- The language called QBE (Query-By-Example) is related to domain calculus .
- 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

  $\{x1, x2, \ldots, xn \mid COND(x1, x2, \ldots, xn, xn+1, xn+2, \ldots, xn+m)\}$

  where x1, x2, . . ., xn, xn+1, xn+2, . . ., xn+m are domain variables that range over domains (of attributes) and COND is a **condition** or **formula** of the domain relational calculus.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Query By Example

- A query language based upon domain relational calculus.
- This language is based on the idea of giving an example of a query using **example elements**.
- An example element stands for a domain variable and is specified as an example value preceded by the underscore character.
- P. (called **P dot**) operator (for "print") is placed in those columns which are requested for the result of the query.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Query By Example

- A user may initially start giving actual values as examples, but later can get used to providing a minimum number of variables as example elements.
- The language is very user-friendly, because it uses minimal syntax.
- QBE was fully developed further with facilities for grouping, aggregation, updating etc. and is shown to be equivalent to SQL.
- The language is available under QMF (Query Management Facility) of DB2 of IBM and has been used in various ways by other products like ACCESS of Microsoft, PARADOX.

Thank You