CHAPTER 2
RELATIONAL DATA MODEL

CLO1: Apply fundamental of Database Management System (DBMS), relational data model and normalization concepts in database development process.(C3, PLO2)

Upon completion of this chapter, students should be able to:

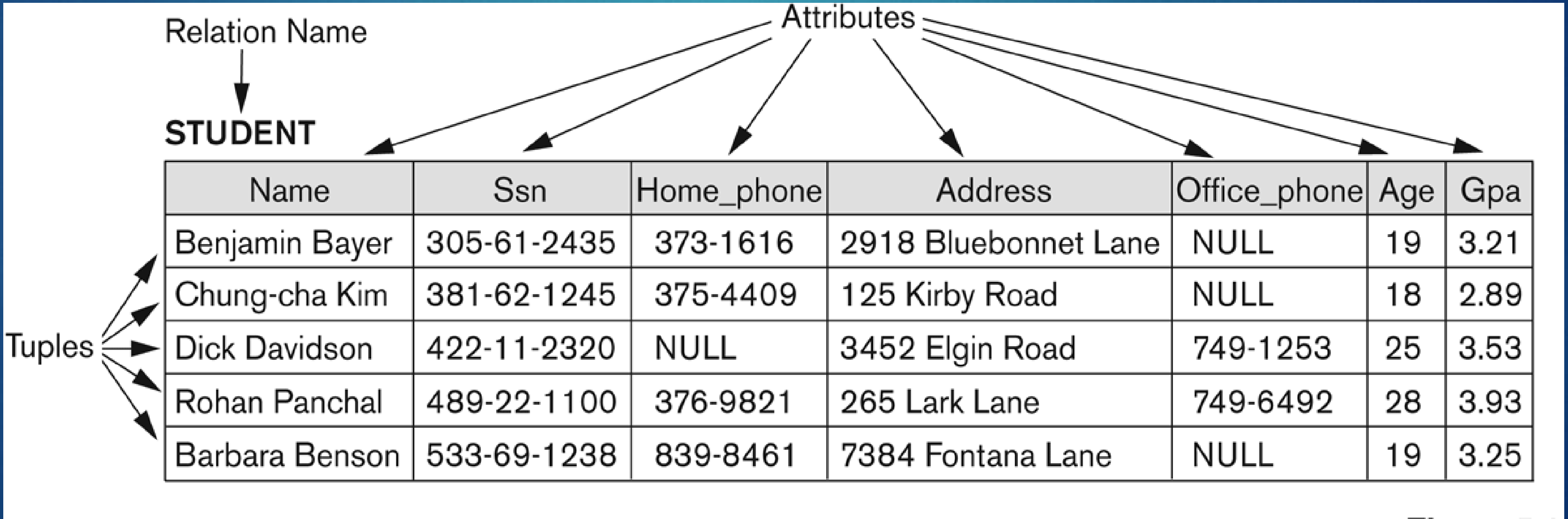
2.1 Explain relational database
2.2 Explain operators or relational algebra

2.1 Explain relational database

RDBMS

- Relational Database Management System (RDBMS) is a database base on the relational model developed by Dr. E.F. Codd of IBM Research in 1970.

- The model represent data in form of table.



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Components & Characteristics of RDBMS

- **RELATION**
  - Is a table with column and rows

- **ATTRIBUTE (FIELD)**
  - An attribute is named column of relation
  - Used to hold information about the objects to be represented in database

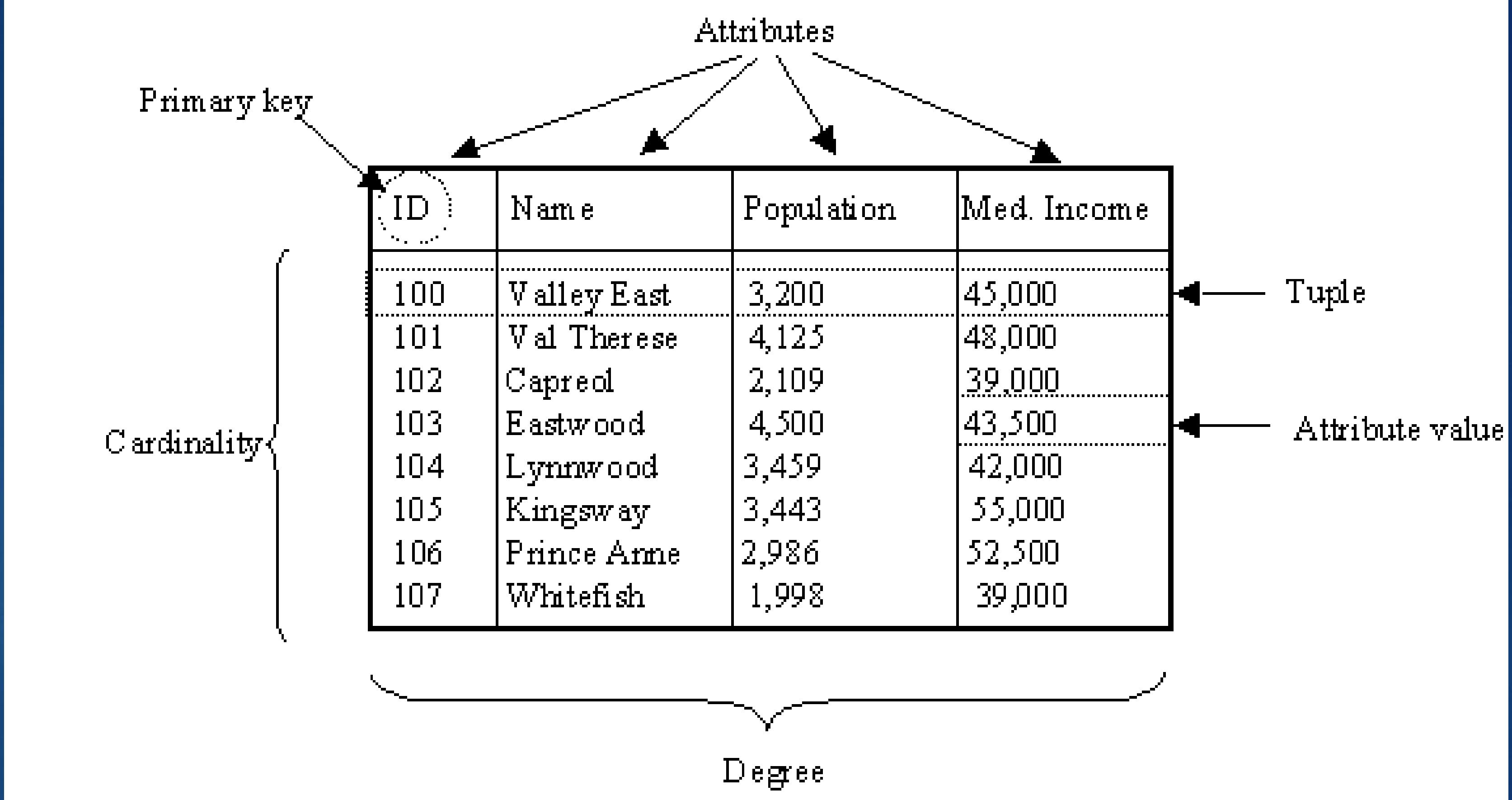
- **RECORD (TUPLE)**
  - A record is a row of a relation

Components & Characteristics of RDBMS

- **DOMAIN**
  - A domain is the set of allowable values for one or more attributes
- **DEGREE**
  - The degree of relation is the number of attributes it contains
- **CARDINALITY**
  - The cardinality of relation is the number of rows it contain.

Components & Characteristics of RDBMS

Figure 13: Characteristics of a relational table



ID	Name	Population	Med. Income
100	Valley East	3,200	45,000
101	Val Therese	4,125	48,000
102	Capreol	2,109	39,000
103	Eastwood	4,500	43,500
104	Lynnwood	3,459	42,000
105	Kingsway	3,443	55,000
106	Prince Anne	2,986	52,500
107	Whitefish	1,998	39,000

A.K. Yeung 1998-10-10 u51-13

Relation Instance

- **Relation: made up of 2 parts:**

1. <u>SCHEMA</u> : specifies name of relation, plus name and type of each column

E.G.

Student(sid: char(10), name: char(20), age: integer, gpa: real, login: char(20)).

2. <u>INSTANCE</u> : a table of values in rows and columns

Relation instance is a set of rows (tuples, records)

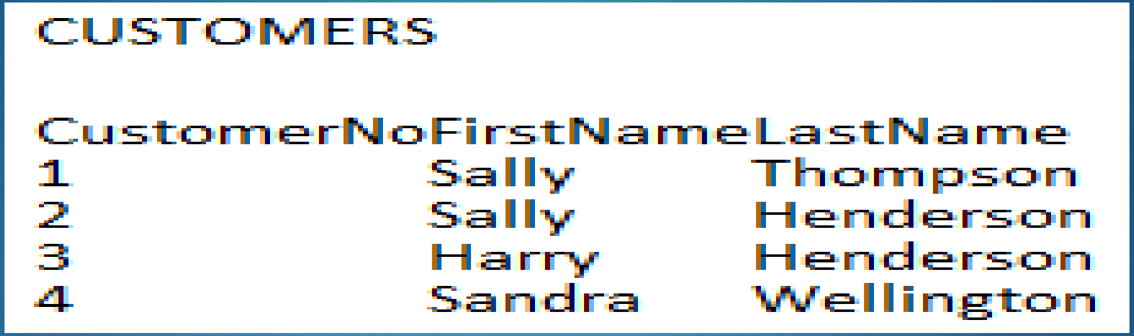
As a set, it is not supposed to have duplicates!

Relation Keys

- There are no duplicate tuples within a relation.
- Therefore, we need to be able to identify one or more attributes (called relational keys) that uniquely identifies each tuple in a relation.
- There are several types of relational keys;
  - Primary Key
  - Candidates Key
  - Foreign Key

Relation Keys

- **PRIMARY KEY**
  - is a column (or columns) in a table that uniquely identifies the rows in that table.



CUSTOMERS

CustomerNo	FirstName	LastName
1	Sally	Thompson
2	Sally	Henderson
3	Harry	Henderson
4	Sandra	Wellington

For example, in the table above, CustomerNo is the primary key.

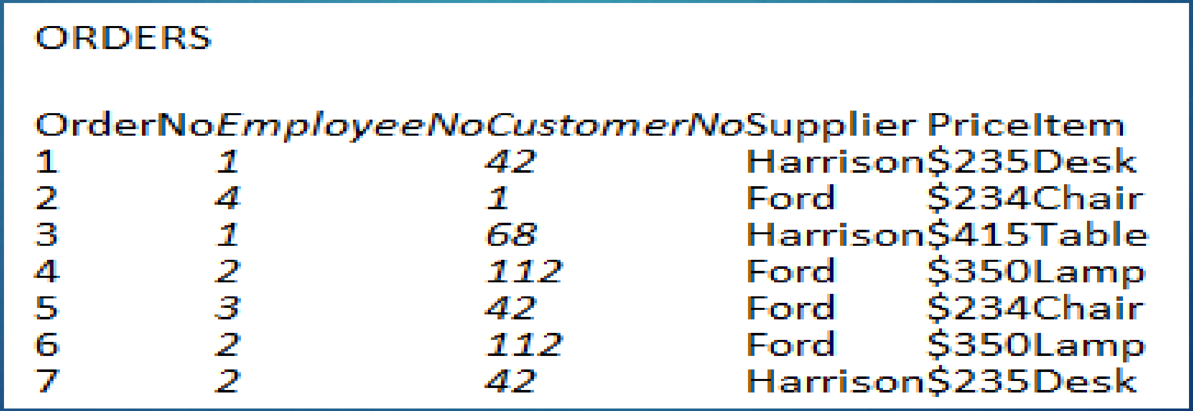
  - The values placed in primary key columns must be unique for each row: no duplicates can be tolerated. In addition, nulls are not allowed in primary key columns.

Relation Keys

- **CANDIDATES KEY**
  - is a column that meets all of the requirements of a primary key. In other words, it has the potential to be a primary key.

Relation Keys

- **FOREIGN KEY**
  - are columns that point to primary key columns.
  - So, for example, OrderNo is the primary key of the table ORDERS below and CustomerNo is a foreign key that points to the primary key in the CUSTOMERS table.



ORDERS

OrderNo	EmployeeNo	CustomerNo	Supplier	Price	Item
1	1	42	Harrison	$235	Desk
2	4	1	Ford	$234	Chair
3	1	68	Harrison	$415	Table
4	2	112	Ford	$350	Lamp
5	3	42	Ford	$234	Chair
6	2	112	Ford	$350	Lamp
7	2	42	Harrison	$235	Desk

Integrity Rules

- Rules of constraints or restrictions that apply to all instances of the database.
- A feature provided by relational database management system (RDBMS), which prevents users or applications from entering inconsistent data.
- There are 2 types of integrity rules;
  - Entity Integrity
  - Referential Integrity

- **ENTITY INTEGRITY**
  - No attribute of a primary key can be null.
  - If we allow null for any part of primary key, we are implying that not all the attributes are needed to distinguish between tuples.

<u>NULL</u>

- Null mean that no value has yet been supplied.
- Null not the same with zero numeric value or text string filled with spaces; zeros and spaces are values
- A null represent the absence of a value.

- **REFERENTIAL INTEGRITY**

  - This rule applies to a foreign keys.

  - If a foreign key exist in a relation, either the foreign key value must match a candidate key value of some tuples in its home relation or the foreign key value must be wholly null.

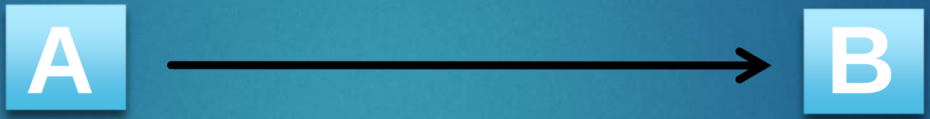
Relational Model Relationships

- Express the specific number of entity occurrences associated with occurrences of the related entity

- There are 3 types of relationships;
  - One to One (1:1)
  - One to Many (1:M)
  - Many to Many (M:N)

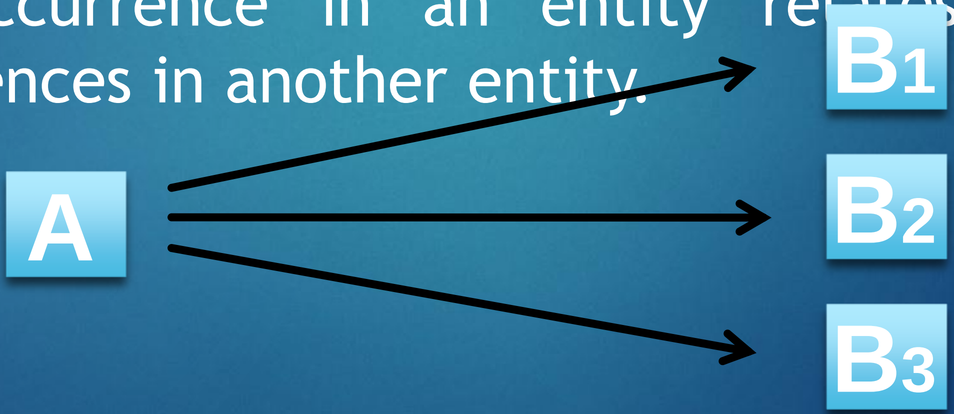
Relational Model Relationships

- ONE TO ONE (1:1)
  - One occurrence of an entity relates to only one occurrence in another entity



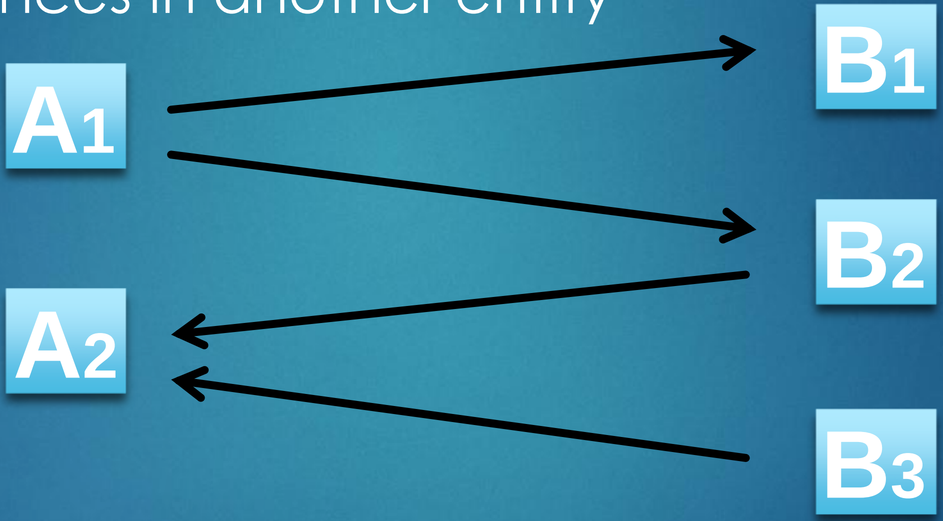
ONE TO MANY (1:M)

- One occurrence in an entity relates to many occurrences in another entity.



Relational Model Relationships

- MANY TO MANY (M:N)
  - many occurrences in an entity relate to many occurrences in another entity



Exercise 1

Give ONE (1) an example for each of cardinality relationship.

1:1

1:M

M:N

Exercise 2

Instruction:

Create complete Relational Data Model. Underline selected primary key.

Case 1:

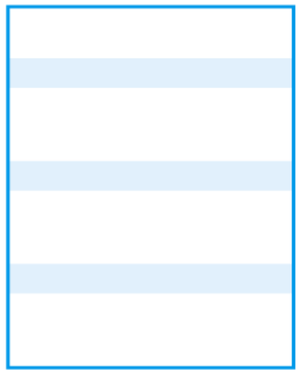
Fantastic Music was a company which offered rental VCD services. Every borrower registered through borrower id num, name, NRIC and telephone num. VCD recorded kept in system including VCD serial num, title and VCD category. If the rental happened, detailed items of necessity will be recorded such as borrower id num, VCD serial num, rental date and return date.

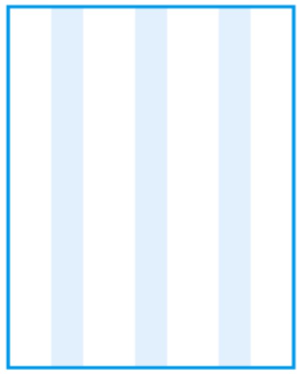
2.2 Explain operators or relational algebra

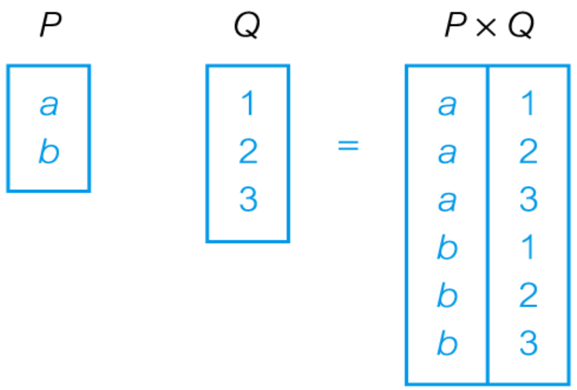
Relational Algebra

- Relational algebra is a (high-level) procedural language associated with the relational model.
- Basic operations in relational algebra:
  - Unary Relational Operations:
    - Selection
    - Projection
  - Cartesian product
  - Union
  - Set Difference
  - Intersection
  - Binary Relational Operations
    - Division operations
    - Join

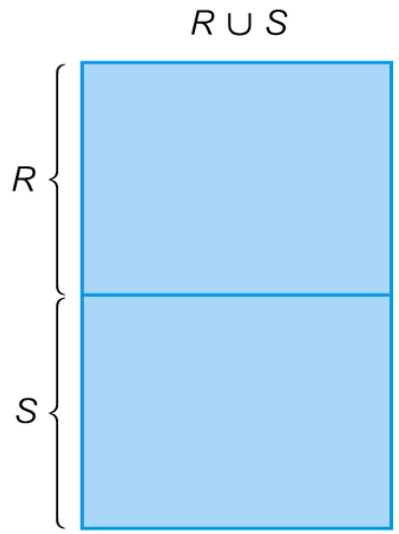
Relational Algebra


(a) Selection

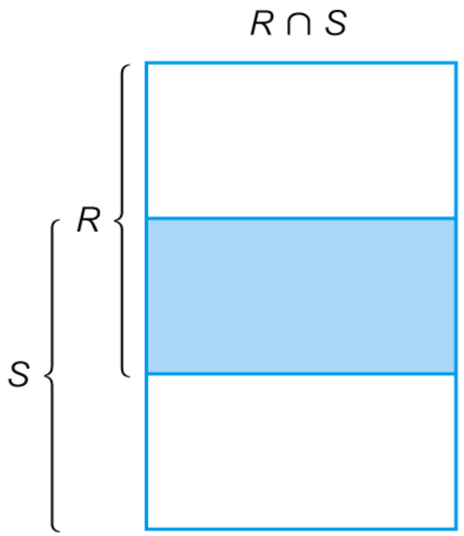

(b) Projection



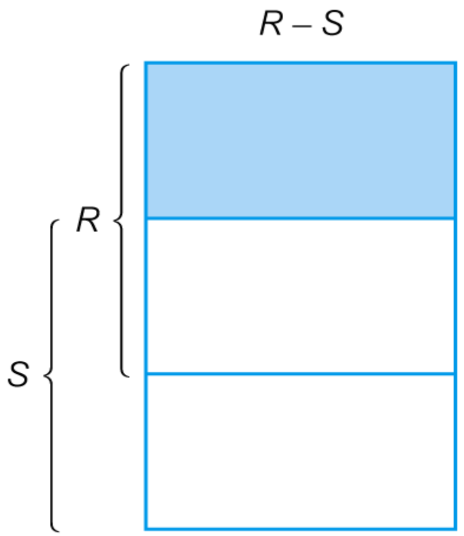
(c) Cartesian product



(d) Union



(e) Intersection



(f) Set difference

5 Basic Operations

- **Selection** (σ) Selects a subset of *rows* from relation (horizontal).
- **Projection** (π) Retains only wanted *columns* from relation (vertical).
- **Cross-product** (x) Allows us to combine two relations.
- **Set-difference** (-) Tuples in r1, but not in r2.
- **Union** ($\cup$) Tuples in r1 and/or in r2.

Example Instances

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

SELECT operation (σ)

- Used to select a subset of the tuples from a relation that satisfies a selection condition
- Symbol -> σ (sigma)
- Syntax : $\sigma_{\text{<selection condition>}}(R)$

SELECT operation (σ)

$\sigma_{rating>8}(S2)$

sid	sname	rating	age
28	yuppy	9	35.0
~~31~~	~~lubber~~	~~8~~	~~55.5~~
~~44~~	~~guppy~~	~~5~~	~~35.0~~
58	rusty	10	35.0

$\pi_{sname,rating}(\sigma_{rating>8}(S2))$

sname	rating
yuppy	9
rusty	10

SELECT operation (σ)

Eg1: Select the STUDENT tuples whose Semester is 3.

$\sigma_{Sem=3}(STUDENT)$

Eg2: Select the MANAGER tuples whose Salary is greater than RM25

$\sigma_{Salary>25000}(MANAGER)$

Eg3: Select the tuples for all SUPERVISOR who either work in IT Department and Salary less than RM20,000 per year, or work in HR Department and ss than RM30,000.

$\sigma_{(Dept=\text{“IT”} \quad AND \quad Salary<20000) \quad OR \quad Dept=\text{“HR”} \quad AND \quad Salary<30000))}(SUPERVISOR)$

PROJECT (π) operation

- Select certain columns from the table and discard the other columns
- Symbol -> Л (pi)
- Syntax – Л $_{\text{<attribute list>}}$ (R)

PROJECT (π) operation

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$\pi_{sname,rating}(S2)$

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$\pi_{age}(S2)$

age
35.0
55.5

PROJECT (π) operation

Eg1: List each student's Regs.No, First Name and Last Name.

$Л_{\text{Regs.No, Fname, Lname}}(\text{STUDENT})$

Eg2: List IDNum, Name and Age of all supervisors who work in IT department.

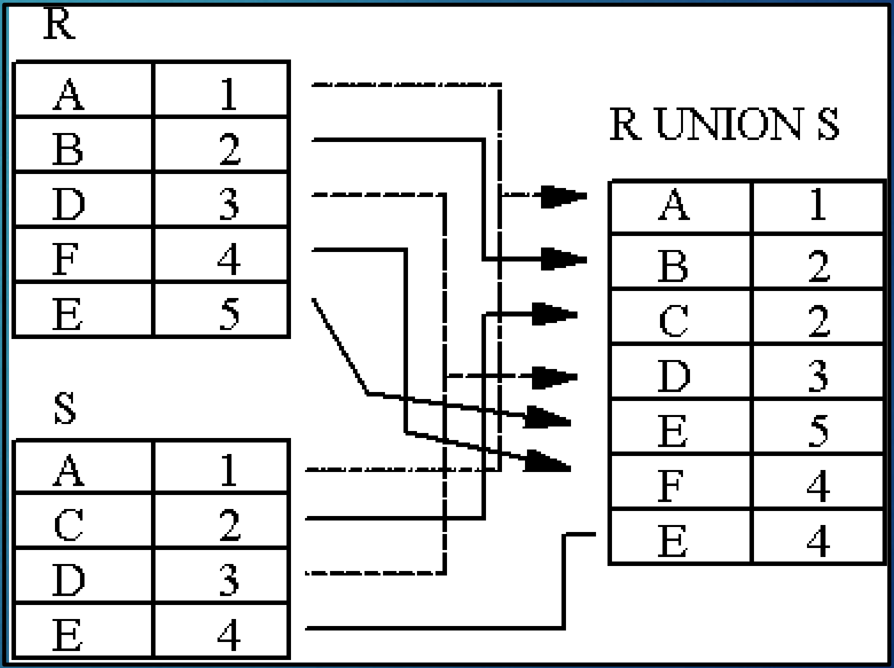
$Л_{\text{IDNum,Name,Age (}\sigma\text{Dept=“IT”}}(\text{SUPERVISOR}))$

UNION (∪) operation

- UNION compares tuples in two relations and create a new relation that contains some of the tuples from each of the input relations.

- Symbol -> ∪ (cup)

- R ∪ S



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

UNION (∪) operation

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup S2$

UNION (∪) operation

Eg1: List all IDNum where there is either a Manager or Supervisor for IT Department.

$\Lambda_{IDNum(\sigma Dept=“IT”}(MANAGER)) \cup$
$\Lambda_{IDNum(\sigma Dept=“IT”}(SUPERVISOR))$

Eg2: List Regs.No for students those register at Department and Hostel.

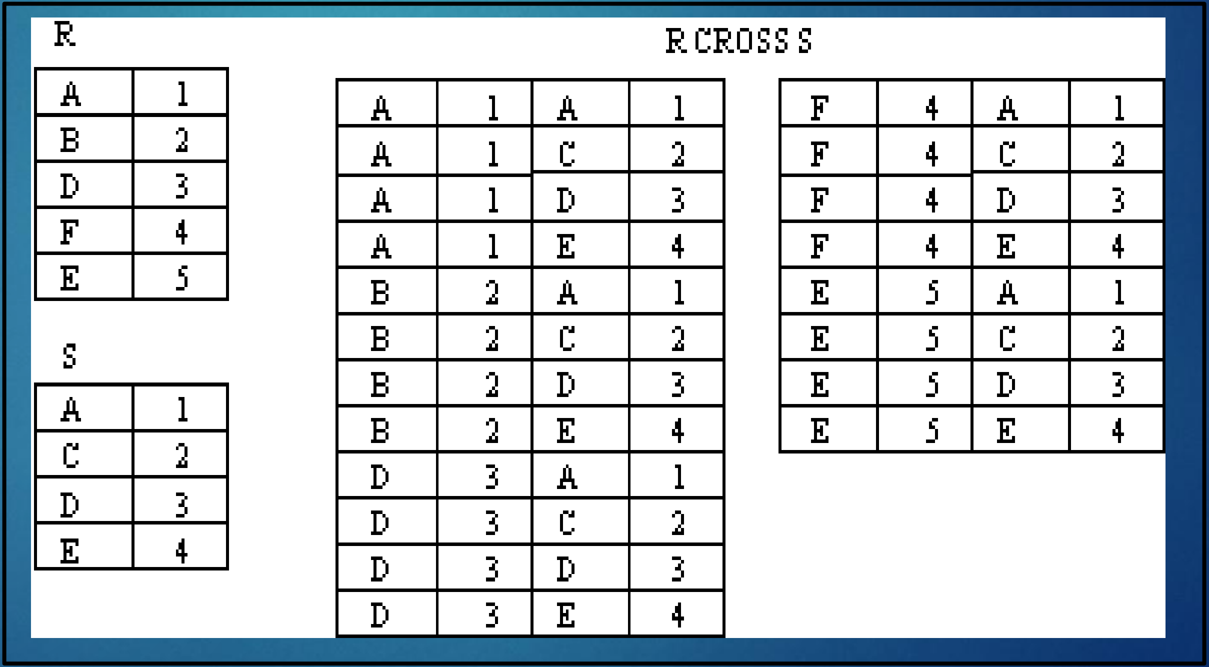
$Л_{Regs.No}(DEPARTMENT) \cup Л_{Regs.No}(HOSTEL)$

CROSS PRODUCT operation

- Defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.

- Denoted by X

- R X S



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

CROSS PRODUCT operation

sid	bid	day
22	101	10/10/96
58	103	11/12/96

R1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

R1 X S1

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/
22	dustin	7	45.0	58	103	11/12/
31	lubber	8	55.5	22	101	10/10/
31	lubber	8	55.5	58	103	11/12/
58	rusty	10	35.0	22	101	10/10/
58	rusty	10	35.0	58	103	11/12/

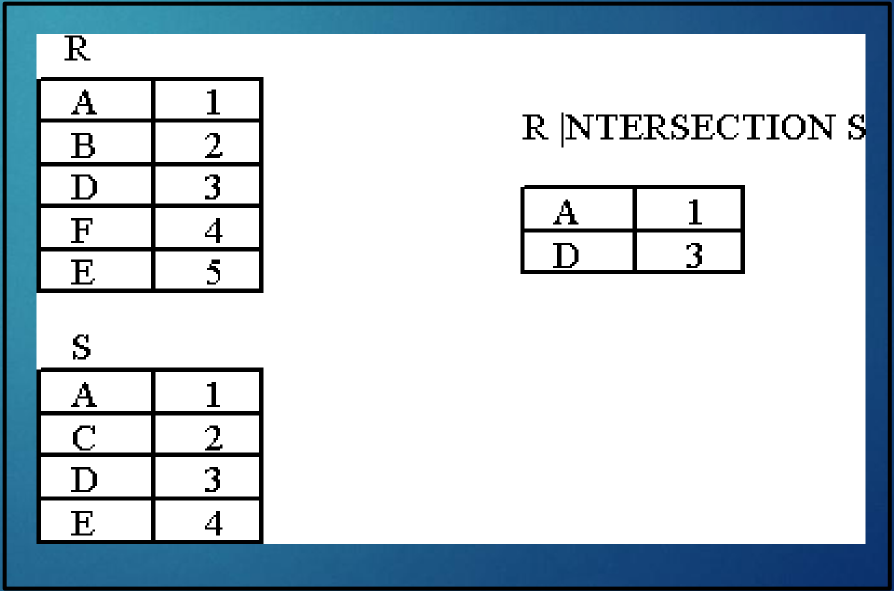
CARTESIAN PRODUCT operation

Eg1: List the Regs.No, Name and NRIC of all students who registred CourseID, CourseName for course.

$\Lambda_{Regs.No,Name,NRIC}(Student) \times (\Lambda_{Regs.No,CourseID,CourseName}(Course))$

INTERSECTION operation

- Defines a relation consisting of the set of all tuples that are in both R and S.
- Symbol -> ∩ (cap)
- R ∩ S



R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R INTERSECTION S

A	1
D	3

INTERSECTION operation

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

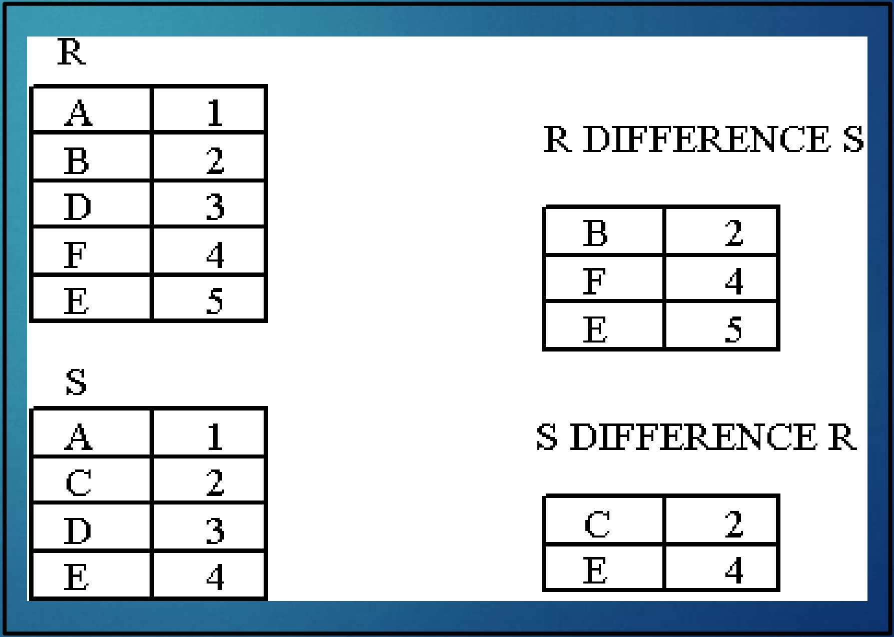
INTERSECTION operation

Eg1: List Regs.No for students those having register at both Department and Hostel.

$Л_{Regs.No}(DEPARTMENT) \cap Л_{Regs.No}(HOSTEL)$

SET DIFFERENCE operation

- Defines a relation consisting of the tuples that are in relation R, but not in S.
- Symbol -> - (minus)
- R - S



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

SET DIFFERENCE operation

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

sid	sname	rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

$S2 - S1$

SET DIFFERENCE operation

Eg1: List Regs.No for students those register at Department but no at Hostel.

$$\Lambda_{\text{Regs.No}(\sigma \text{course}=\text{“DDT3A”}}(\text{DEPARTMENT})) - \Lambda_{\text{Regs.No}(\sigma \text{course}=\text{“DDT3A”}}(\text{HOSTEL}))$$

JOIN operation

- Compute R X S
- Select rows where attributes that appear in both relations have equal values
- Project all unique atttributes and one copy of each of the common ones.

Types of Joins in MySQL

INNER JOIN: Returns records that have matching values in both tables

LEFT JOIN: Returns all records from the left table, and the matched records from the right table

RIGHT JOIN: Returns all records from the right table, and the matched records from the left table

CROSS JOIN: Returns all records from both tables

NATURAL JOIN operation

sid	bid	day
22	101	10/10/96
58	103	11/12/96

R1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

$R1 \bowtie S1$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

OUTER JOIN

- "Pads With Nulls"
- A variant of the join, to keep all pieces of information from the operands.
- Three Variants :
  - Left – only tuples of the left operand are padded.
  - Right – only tuples of the right operand are padded.
  - Full – tuples of both operands are padded

R1

Employee	Department
Smith	Sales
Black	Production
White	Production

R2

Department	Head
Production	Mori
Purchasing	Brown

OUTER LEFT JOIN

- R1 $\bowtie$ LEFT R2

EMPLOYEE	DEPARTMENT	HEAD
Smith	Sales	Null
Black	Production	Mori
White	Production	Mori

OUTER RIGHT JOIN

- $R1 \bowtie$ RIGHT R2

EMPLOYEE	DEPARTMENT	HEAD
Black	Production	Mori
White	Production	Mori
Null	Purchasing	Brown

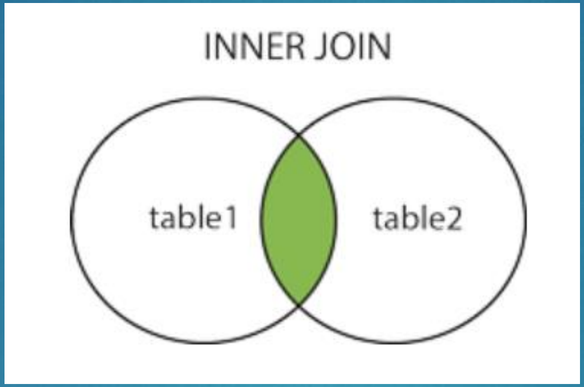
OUTER FULL JOIN

- R1 ⋈ RIGHT R2

EMPLOYEE	DEPARTMENT	HEAD
Smith	Sales	Null
Black	Production	Mori
White	Production	Mori
Null	Purchasing	Brown

INNER JOIN

The `INNER JOIN` keyword selects records that have matching values in both tables.



INNER JOIN Syntax

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_
name;
```

INNER JOIN EXAMPLE

The following SQL statement selects all orders with customer information:

```sql
SELECT Orders.OrderID,
Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID =
Customers.CustomerID;
```

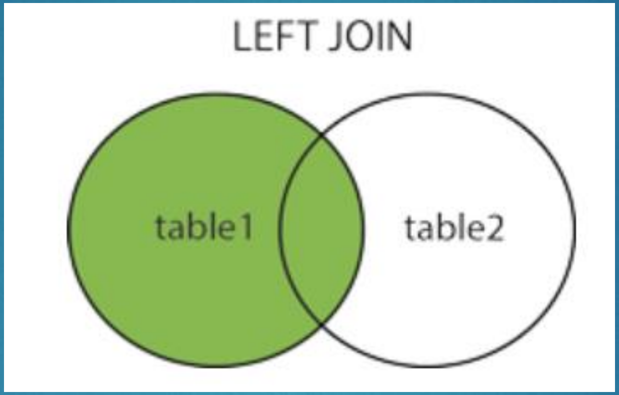
JOIN Three Tables

The following SQL statement selects all orders with customer and shipper information:

```
SELECT Orders.OrderID,
Customers.CustomerName,
Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.Custome
rID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperI
D = Shippers.ShipperID);
```

LEFT JOIN

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records (if any) from the right table (table2).

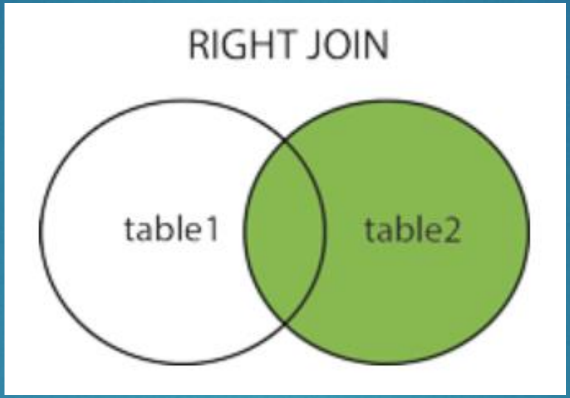


LEFT JOIN Syntax

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_
name;
```

RIGHT JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records (if any) from the left table (table1).



LEFT JOIN Syntax

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_
name;
```

UNION

- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order

UNION Syntax

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

UNION ALL Syntax

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

SQL UNION Example
The following SQL statement returns the cities (only distinct values) from both the "Customers" and the "Suppliers" table:

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```