Database Systems – I

BIT Semester 2

4

Relational Algebra
Procedural DML

1. **Database System Environment (5MCQs)**
2. **Integrity Constraints and DDL (5MCQs)**
3. **Working with database using DML (10MCQs)**
4. **Relation Algebra (6MCQs)**
   Duties of Query Processor and Query Optimizer Modules of a DBMS, Introduction to relational Algebra, Unary Operators-Selection and Projection, Operators which require Union Compatibility-Union, Intersection and Set-difference, Joins-Cross Join (Cartesian Product)/Inner Join/Natural Join/Outer Join (Left-Outer, Right-Outer, Full-Outer), Division, Classification of RA-Operations, RA and SQL, RA Operation Evaluation
5. **Database Designing Process with ER Diagrams (8MCQs)**
6. **Normalization (5MCQs)**
7. **Views and Security with DCL (4MCQs)**
8. **Execute duties of a Database Administrator**

Base Plan

Class Date

1. ☐ **Selection and Projection ← Unary Operator** [*1*] [*1/8/19*]
   - ☐ Query Processor, Query Optimizer and Command Processor in DBMS
   - ☐ Selection and Remove Rows depend on the given Predicate
   - ☐ Projection set of columns given by removing other columns from the table
   - ☐ Closure
   - ☐ Drawing ER for the given set of tables – Reverse Engineering
   - ☐ 2011-19 ☐ 2018-18 ☐ 2017-20 ☐ 2007-13

2. ☐ **Union, Intersection and Set-Difference ← Union Compatibility** [ ] [ ]
   - ☐ Union and Union Compatibility
   - ☐ Set Difference
   - ☐ Intersection
   - ☐ Derive Intersection using Set-Difference
   - ☐ 2017-16 ☐ 2011-35 ☐ 2011-36 ☐ 2015-21 ☐ 2015-22 ☐ 2010-18
   - ☐ 2007-07 ☐ 2006-20 ☐ 2016-17 ☐ 2014-13 ☐ 2012-22

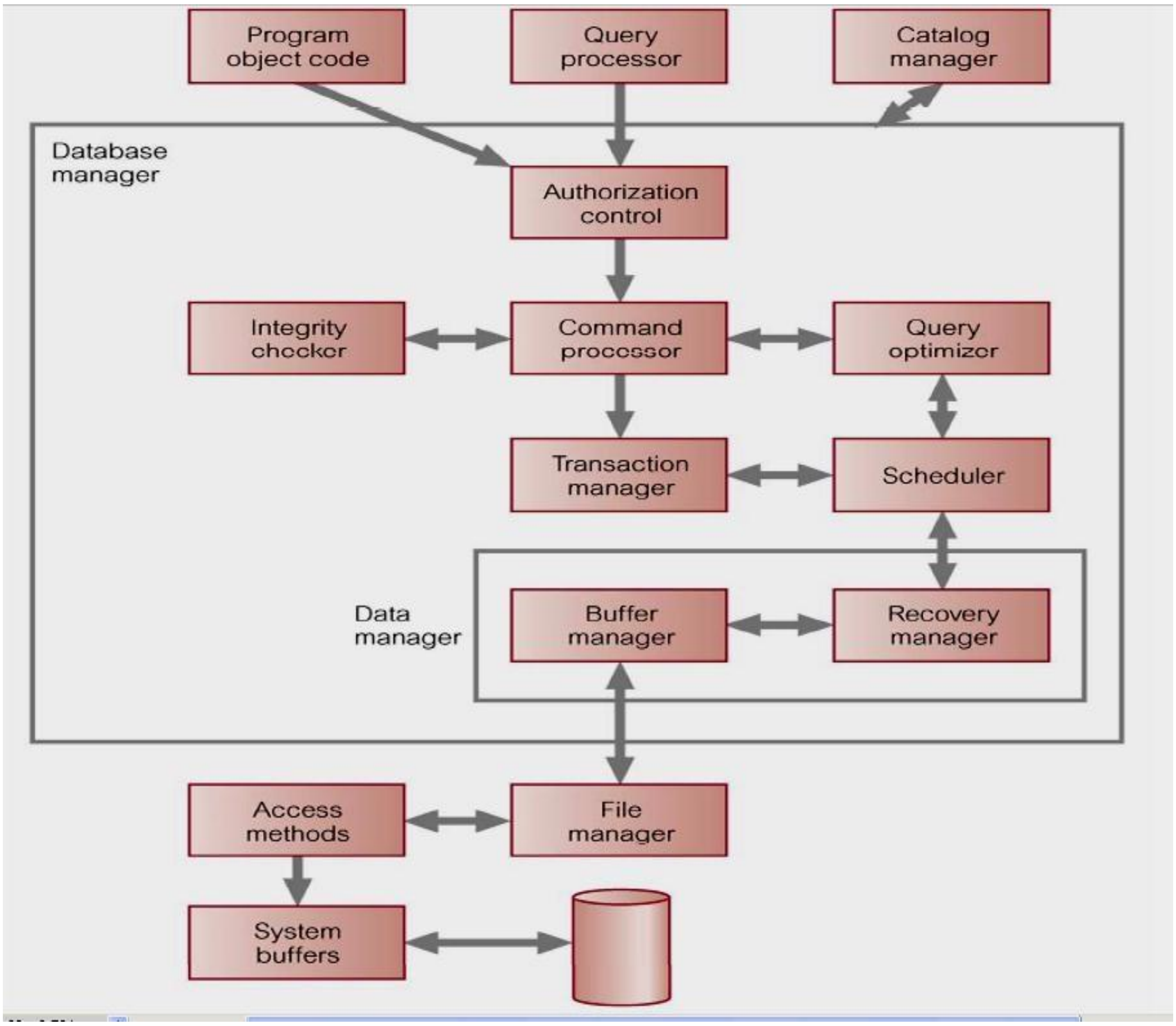
3. ☐ **Join – Cross, Inner, Natural and Outer** [ ] [ ]
   - ☐ Cross Join – Cartesian Product
   - ☐ Inner Join and Natural Join
   - ☐ 2012-23 ☐ 2012-21 ☐ 2012-29 ☐ 2018-28 ☐ 2017-32 ☐ 2018-19
   - ☐ 2018-21 ☐ 2009-16 ☐ 2009-18 ☐ 2017-21 ☐ 2017-23 ☐ 2014-10
   - ☐ 2014-11 ☐ 2014-12 ☐ 2016-11 ☐ 2016-12 ☐ 2015-20 ☐ 2013-15
   - ☐ 2010-16 ☐ 2008-11 ☐ 2008-12 ☐ 2008-13 ☐ 2008-14 ☐ 2007-14
   - ☐ 2007-05
   - ☐ Inner Join vs Outer Join
   - ☐ Left/Right/Full Outer Joins
   - ☐ 2013-14 ☐ 2018-20 ☐ 2009-17 ☐ 2017-22 ☐ 2016-13 ☐ 2013-16
   - ☐ 2015-14 ☐ 2017-44 ☐ 2018-39

4. ☐ **Division** [ ] [ ]
   - ☐ 2010-17 ☐ 2013-17 ☐ 2009-15 ☐ 2006-21 ☐ 2016-14

5. ☐ **Conclusion to RA** [ ] [ ]
   - ☐ Procedural vs Declarative Languages
   - ☐ 2018-01 ☐ 2009-05 ☐ 2017-03 ☐ 2014-15 ☐ 2012-19
   - ☐ 2011-37 ☐ 2010-05
   - ☐ Classification of RA Operations
   - ☐ Basic vs Derived
   - ☐ 2018-15 ☐ 2008-09 ☐ 2006-19
   - ☐ Common for RA and Mathematical-Set-Theory
   - ☐ 2009-14 ☐ 2007-08
   - ☐ Mathematical Evaluation of RA-Operations
   - ☐ 2013-12 ☐ 2015-23 ☐ 2016-09 ☐ 2016-10 ☐ 2014-14
   - ☐ 2012-20 ☐ 2010-15
   - ☐ SQL with RA
   - ☐ 2015-24 ☐ 2013-13 ☐ 2010-24 ☐ 2010-18 ☐ 2012-22

Selection, Projection – Unary Operators of Relation Algebra
Duties of Query Optimizer and Query Processor

Revise Duties of DBMS Components



Catalog Manager	Read and Write Meta-Data in System-Catalog (Data Dictionary)
Authorization Controller	Authenticate users by username and password Authorize users according to the DCL privileges assigned
Scheduler	Manage Concurrent Transaction to minimize conflicts
Recovery Manger	Recover the database to the last consistent status when a failure occur

Query Processor

SQL-Quarry will be converted into set of statements called a Procedure.
These Procedures are written in Relational Algebra(RA).
SQL tells what to retrieve, not how to retrieve. ← Declarative/Non-Procedural
RA tells how to retrieve beyond from what to retrieve. ← Procedural/Non-Declarative

Command Processor
Retrieve the results relevant to the given RA Procedure.

Consider the following table,

no	*name*	*gender*	*age*	*tp*	*city*
1	***Namal***	***male***	***22***	***0013456702***	***Gampaha***
2	***Kumara***	***male***	***27***	***0217896465***	***Boralla***
3	***Vishaka***	***female***	***25***	***0799956662***	***Boralla***
4	***Anuradha***	***male***	***23***	***0567856705***	***Nawala***
5	***Nirmala***	***female***	***18***	***0987034702***	***Gampaha***
6	***Amal***	***male***	***32***	***0348787890***	***Maharagama***

SQL–Query (What we need)

Select name, age, gender
From student
Where gender="female"

Relational Algebra Procedure (How to be retrieved)
Require 2 Steps

R1 ← $\sigma_{\text{gender = 'female'}}$ (Student) **Selection(σ) : Remove Rows**

Operator → Selection (σ) → Remove Rows
Predicate → Selection Condition (gender = 'female') → 'male' Rows will be removed
Operand Table → Student → 6 Columns, 6 Rows
Result Table → R1 → 6 Columns, 2 Rows

no	**name**	**gender**	**age**	**tp**	**city**
3	Vishaka	female	25	0799956662	Boralla
5	Nirmala	female	18	0987034702	Gampaha

R2 ← $\pi_{\text{name, age, gender}}$ (R1) **Projection(π) : Remove Columns**

Operator → Projection (π) → Remove Columns
Columns to be Selected → name, age, gender
Operand Table → R1 → 6 Columns, 2 Rows
Result Table → R2 → 3 Columns, 2 Rows

name	**tp**	**gender**
Vishaka	0799956662	female
Nirmala	0987034702	female

Selection and Projection
Unary Operators: Works on a single table
Selection (σ) → Remove Rows that does not satisfy the given predicate(condition)
Projection (π) → Remove Columns other than the given list

$$R1 \leftarrow \pi_{\text{name, age, address}} (\text{ Student })$$
$$R2 \leftarrow \sigma_{\text{gender = 'female'}} (\text{ R1 })$$

1st Query is incorrect as "Student" does not have a column called "address" to be retrieved.
2nd Query is incorrect as "R1" does not have a column called "gender" to filter out as the predicate specified.

Query Optimizer

Select the most efficient procedure from a set of possible Procedures that could retrieve required result of a given SQL query.

Procedure–1 (Project after Selection)

$$R1 \leftarrow \sigma_{\text{gender = 'female'}} (\text{ Student })$$
$$R2 \leftarrow \pi_{\text{name, age, gender}} (\text{ R1 })$$

Procedure–2 (Select after Projection)

$$R1 \leftarrow \pi_{\text{name, age, gender}} (\text{ Student })$$
$$R2 \leftarrow \sigma_{\text{gender = 'female'}} (\text{ R1 })$$

Suppose, there are a smaller number of "female" in the student table, then it is better to select them first to minimize the data to be placed on the RAM, hence Procedure–1 is efficient than Procedure-2. This procedure selection is essential when the Query Optimizer have to compare Selection/Projection against Join which you will learn later in this lesson.

Closure

If $A = B + C$, $M = N * A$ Then $M = N * (B + C)$

Example-1

$$R1 \leftarrow \pi_{\text{name, age, gender}} (\text{ Student })$$
$$R2 \leftarrow \sigma_{\text{gender = 'female'}} (\text{ R1 })$$

$$R2 \leftarrow \sigma_{\text{gender = 'female'}} (\pi_{\text{name, age, gender}} (\text{ Student }))$$

Example-2

$$R1 \leftarrow \pi_{\text{name, age, gender}} (\text{ Student })$$
$$R2 \leftarrow \sigma_{\text{gender = 'female'}} (\text{ R1 })$$
$$R3 \leftarrow \pi_{\text{name, age}} (\text{ R2 })$$

$$R3 \leftarrow \pi_{\text{name, age}} (\sigma_{\text{gender = 'female'}} (\pi_{\text{name, age, gender}} (\text{ Student })))$$

Usually RA-Questions in Past-Papers will come as a set of Questions based on given set of tables,
Here, it is need to draw the ER-Diagram relevant to the given set of tables and discuss business scenario behind the ER-Diagram.

Reverse Engineering ← Drawing ER using Tables,
Forward Engineering ← Mapping ER to get set of Tables)

Draw the ER-Diagram using a set of tables
Reverse Engineering

Example-1
2018 (18,19,20,21)

> **Consider the following schema to answer questions from (18) to (21). Primary Keys are underlined and Foreign Keys are in Bold italics. Lectures can teach courses offered by other departments as well.**
>
> Lecturer (<u>EmpNo</u>, Name, Gender, Salary, Category, ***DNo***).
> Department (<u>DNo</u>, Dname, ***HeadEmpNo***)
> Course (<u>CNo</u>, Cname, Credits, ***DNo***)
> Deliver (<u>EmpNo</u>, <u>***CNo***</u>, Hours)
> Research_Fund (<u>RFName</u>, ***EmpNo***, Amount)
>
> **The symbols π, σ, ⋈, ⋊ ,⋉ , U, ∩ , and – are used to denote the relational operators Projection, Selection, Natural Join, Right Outer Join, Left Outer Join, Union, Intersection and Set difference respectively.**

Consider following key columns of the given tables
FKs in 1→M and 1→1 Optional Mapping, Composite PKs with FKs relevant with the M→M Mappings.

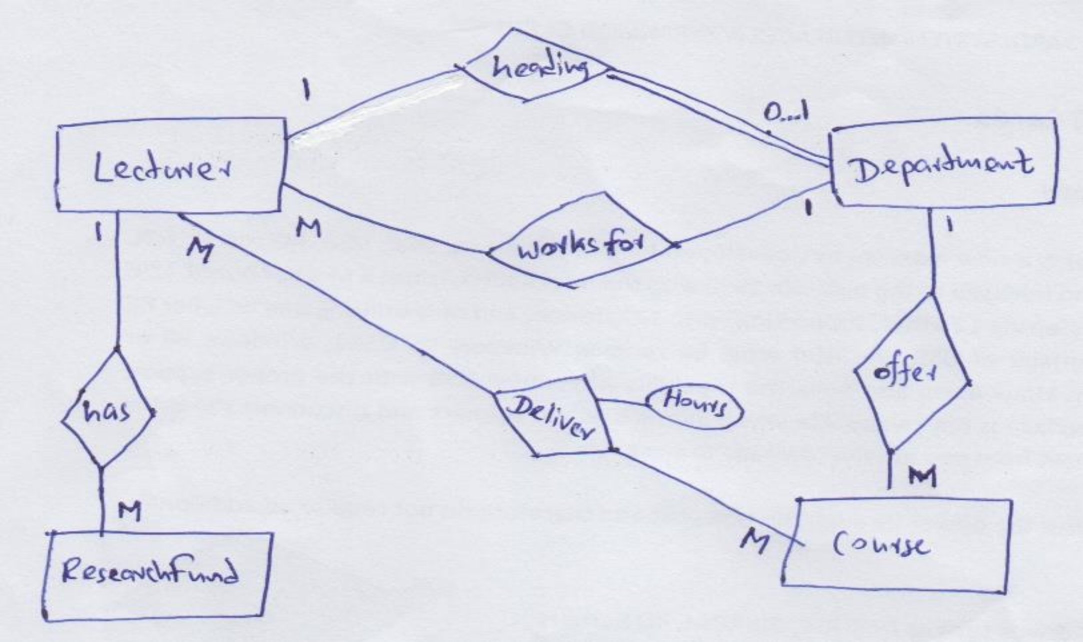
<u>Lecture</u> → DNo (FK) Think 1→M Mapping Rule, Add the PK of 1-side as a FK to the Many-side
Here, 1 lecture is working on 1 department, 1 department may have Many lectures
So, Department ←→ Lecture must have 1→M Relationship

<u>Course</u> → DNo (FK) Think 1→M Mapping Rule, Add the PK of 1-side as a FK to the Many-side
Here, 1 course is offered by 1 department, 1 department may offer Many cources
So, Department ←→ Course must have 1→M Relationship

<u>Research_Fund</u> → EmpNo (FK) Same as the above, Lecturer ←→ Research_Fund must have 1→M Relationship

<u>Department</u> → HeadEmpNo (FK)
Think 1→1-Optional Mapping Rule, Add the PK of Optional-Side as a FK to the Mandatory-Side
Here, 1 lecture may lead only 1 department, 1 department has one head.
All department (Mandatory Side) must have a head,
But all lectures (Optional Side) do not need to be department-heads.
So, Department ←→ Lecture must have 1→1-Optional Relationship

<u>Deliver</u> → Composite PK using 2 FKs → EmpNo(From Lecturer), CNo(from Course)
Think M→M Mapping Rule, Add the PK of associated tables, Make them a Composite PK, additional attribute(hours) of the relationship must also be included in the 3rd table(Deliver).
So, Course ←→ Lecture must have M→M Relationship.
One lecture may deliver many courses, One course may be delivered by many lectures



Example-2

2014(10,11,12,13)

Answer the questions from 10 – 13 considering the following scenario.

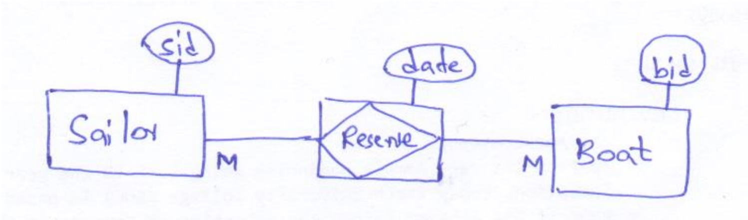
> A local harbour database contains the following tables: Sailor, Boat and Reservation.
> Sailor (**sid**, sname, rating, age)
> *Where sid is Sailor ID and sname is Sailor name*
> Boat (**bid**, bname, colour)
> *Where bid is Boat ID and bname is Boat name*
> Reservation (**sid**, **bid**, date)

Reservation → Composite PK using 2 FKs → bid (from Boat), sid (from Sailor)

Think M→M Mapping Rule, Add the PK of associated tables, Make them a Composite PK, additional attribute(date) of the relationship must also be included in the 3rd table (Reservation).

So, Sailor ←→ Boat must have M→M Relationship.

One Sailor may reserve many boats, one boat may be reserved by many sailors



Exercise : Draw the ER for the following tables given in each year,

2018 (18,19,20,21)	2009(15,16,17,18)	2017(20,21,22,23)	2016(11,12,13,14)
2015(20,21,22,23)	2007(13,14,15,16)	2010(16,17,18)	2014(10,11,12,13)
2013(14,15,16,17)	2008(11,12,13,14)	2006(20,21)	

PP **Selection Projection**

2011-19 ☐ ☐ ☐ 2018-18 ☐ ☐ ☐ 2017-20 ☐ ☐ ☐ 2007-13 ☐ ☐ ☐

Union, Intersection, Set-Difference
Union Compatibility

Union Compatibility

Union, Intersection and Set-Difference are binary-operators that use 2 operand tables as inputs.
Number of columns and data-type of each column must be same for both tables
But column names need not to be same.

bit

name	age
Tharindu	22
Pathum	27
Thilan	25
Dilanka	23
Nalinda	18

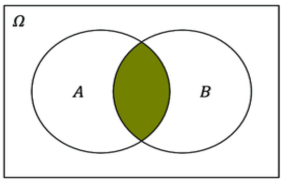
fit

name	age
Nalinda	22
Salani	27
Supun	25
Pathum	23
Danuka	18

<u>Intersection</u>

To find those who did both BIT and FIT.

R1 ← bit ∩ fit

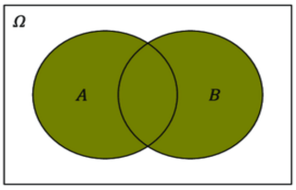


name	age
Pathum	27
Nalinda	18

<u>Union</u>

To find all student who did BIT, FIT or both
(Those who did both will be included once)

R1 ← bit U fit

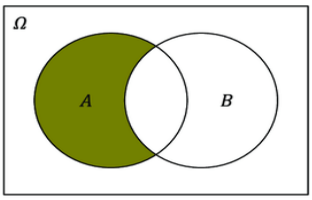


name	age
Tharindu	22
Pathum	27
Thilan	25
Dilanka	23
Nalinda	18
Salani	27
Supun	25
Danuka	18

<u>Set Difference</u>

To find those who did BIT but not did FIT

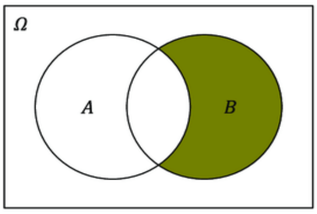
R1 ← bit – fit



name	age
Tharindu	22
Thilan	25
Dilanka	23

To find those who did FIT but not did BIT

R1 ← fit – bit



name	age
Salani	27
Supun	25
Danuka	18

**Both Union and Intersection are Commutative.

A U B = B U A

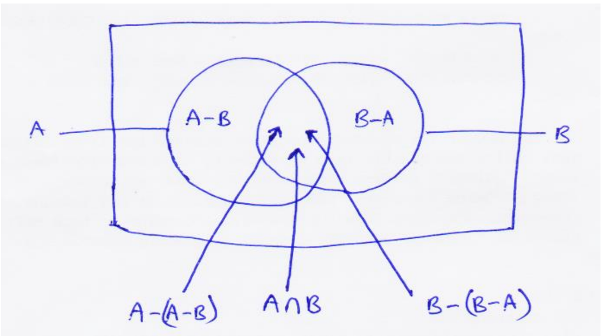
A ∩ B = B ∩ A

**But Set Difference is not Commutative

A – B ≠ B – A

fit – bit ≠ bit – fit

Intersection derives from set-difference



A ∩ B = A – (A – B) = B – (B – A)

Project suitable columns to make them Union Compatible before applying U, ∩ , –

Research_Fund ∩ Department

Above Operation is syntactically incorrect as these tables are not union compatible.
Because they have different number/type of columns.

Project needed columns for compatibility as follows, one column with lecture numbers.

π_{EmpNo} (Research_Fund) ∩ $\pi_{\text{HeadEmpNo}}$ (Department)

PP **Union/Intersection/Set-Difference and Union Compatibility**

2017-16 ☐ ☐ ☐ 2011-35 ☐ ☐ ☐ 2011-36 ☐ ☐ ☐ 2015-21 ☐ ☐ ☐
2015-22 ☐ ☐ ☐ 2010-18 ☐ ☐ ☐ 2007-07 ☐ ☐ ☐ 2006-20 ☐ ☐ ☐
2016-17 ☐ ☐ ☐ 2014-13 ☐ ☐ ☐ 2012-22 ☐ ☐ ☐

Join – Cross | Inner | Natural | Outer
Efficiency of Execution

Consider the following tables for the Join operation described

Employee

empno	empname	workdep
1	Sunil	2
2	Nirmala	1
3	Vimal	3
4	Bimal	1
5	Kusum	3
6	Neela	2
7	Jeewani	4
8	Sanjaya	4
9	Amal	1
10	Narmada	null

Department

depno	depname	extension	head
1	Admin	234	3
2	IT	235	1
3	Engineering	236	5
4	Examination	237	null

Discuss answers for the following questions based on above tables.

What is the Department that the Vimal is working?
Who is the head of the IT Department?
Who is the head of the Bimal?
What is the Department that the Narmada is working?
Who is the head of the Examination Department?

Cartesian Product / Cross Join

Any table could cross join with any table without having any pre-requests.
Concatenate all the tuples of one table with all the tuples of the other table.
Columns will be added(3+4→7), rows will be multiplied(10x4→40).

Employee

empno	empname	workdep
1	Sunil	2
2	Nirmala	1
3	Vimal	3
4	Bimal	1
5	Kusum	3
6	Neela	2
7	Jeewani	4
8	Sanjaya	4
9	Amal	1
10	Narmada	null

Department

depno	depname	extension	head
1	Admin	234	3
2	IT	235	1
3	Engineering	236	5
4	Examination	237	null

R1 ← Employee x Department

empno	empname	workdep	depno	depname	extension	head
1	Sunil	2	1	Admin	234	3
1	Sunil	2	2	IT	235	1
1	Sunil	2	3	Engineering	236	5
1	Sunil	2	4	Examination	237	null
2	Nirmala	1	1	Admin	234	3
2	Nirmala	1	2	IT	235	1
2	Nirmala	1	3	Engineering	236	5
2	Nirmala	1	4	Examination	237	null
3	Vimal	3	1	Admin	234	3
3	Vimal	3	2	IT	235	1
3	Vimal	3	3	Engineering	236	5
3	Vimal	3	4	Examination	237	null
4	Bimal	1	1	Admin	234	3
4	Bimal	1	2	IT	235	1
4	Bimal	1	3	Engineering	236	5
4	Bimal	1	4	Examination	237	null
5	Kusum	3	1	Admin	234	3
5	Kusum	3	2	IT	235	1
5	Kusum	3	3	Engineering	236	5
5	Kusum	3	4	Examination	237	null
6	Neela	2	1	Admin	234	3
6	Neela	2	2	IT	235	1
6	Neela	2	3	Engineering	236	5
6	Neela	2	4	Examination	237	null
7	Jeewani	4	1	Admin	234	3
7	Jeewani	4	2	IT	235	1
7	Jeewani	4	3	Engineering	236	5
7	Jeewani	4	4	Examination	237	null
8	Sanjaya	4	1	Admin	234	3
8	Sanjaya	4	2	IT	235	1
8	Sanjaya	4	3	Engineering	236	5
8	Sanjaya	4	4	Examination	237	null
9	Amal	1	1	Admin	234	3
9	Amal	1	2	IT	235	1
9	Amal	1	3	Engineering	236	5
9	Amal	1	4	Examination	237	null
10	Narmada	null	1	Admin	234	3
10	Narmada	null	2	IT	235	1
10	Narmada	null	3	Engineering	236	5
10	Narmada	null	4	Examination	237	null

Inner Join

Tables must have at least one common column to be joined with each other.
Above 2 tables could be joined in 2 different ways (workdep ←→depno, head ←→empno)
First do the Cross-Join. Then Eliminate Unmatched Rows using values of common column. (Arrowed Rows,)
Columns will be added(3+4→7),
rows will be multiplied(10x4→40), but then will be eliminated (3/4 out of 4) → 9 Rows indicated by short Arrows.
rows will be multiplied(10x4→40), but then will be eliminated (9/10 out of 10) → 3 Rows indicated by long Arrows.

(1) If the join condition (common column) is "workdep ←→ depno"

R1 ← Employee ⋈ $_{\textbf{workdep=depno}}$ Department

empno	empname	workdep	depno	depname	extension	head
1	Sunil	2	2	IT	235	1
2	Nirmala	1	1	Admin	234	3
3	Vimal	3	3	Engineering	236	5
4	Bimal	1	1	Admin	234	3
5	Kusum	3	3	Engineering	236	5
6	Neela	2	2	IT	235	1
7	Jeewani	4	4	Examination	237	null
8	Sanjaya	4	4	Examination	237	null
9	Amal	1	1	Admin	234	3

This join is important, when one wants to find employee list with any details of their working departments
What is the extension of the sunil's working department?
What is the name of the department Amal is working?

(2) If the join condition (common column) is "head ←→ empno"

R1 ← Employee ⋈ $_{\textbf{empno=head}}$ Department

empno	empname	workdep	depno	depname	extension	head
1	Sunil	2	2	IT	235	1
3	Vimal	3	1	Admin	234	3
5	Kusum	3	3	Engineering	236	5

To show this join clearly,
perform "R1 ← Department x Employee" instead of "R1 ← Employee x Department" though both are same

R1 ← Department ⋈ $_{\textbf{head=empno}}$ Employee

1 Department Row concatenate with 10 Employee Rows. Then 9 will be eliminated.

Department

depno	depname	extension	head
1	Admin	234	3
2	IT	235	1
3	Engineering	236	5
4	Examination	237	null

Employee

empno	empname	workdep
1	Sunil	2
2	Nirmala	1
3	**Vimal**	**3**
4	Bimal	1
5	Kusum	3
6	Neela	2
7	Jeewani	4
8	Sanjaya	4
9	Amal	1
10	Narmada	null

Elimination of 1st Department
3→1, 3→2, 3→4, 3→5, 3→6, 3→7, 3→8, 3→9, 3→10
Preservation of 1st Department
3→3 (Matching Tuple)

depno	depname	extension	head	empno	empname	workdep
1	Admin	234	3	3	Vimal	3
2	IT	235	1	1	Sunil	2
3	Engineering	236	5	5	Kusum	3

This join is important, when one wants to find department list with any details of their heads.

What is the name of the head of the IT-department?

Suppose, there is a "address" column in "Employee" table, then

What is the address of the head of the Engineering-department?

Inner Join reflects Intersection

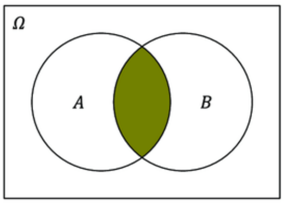
Inner-Join (select rows with same value in both tables) with common column is equal to the Intersection (select rows with same value in both tables). Outcome of both operations are somehow similar. [2018-19-(a) vs (e)]

Hence, R1 $\leftarrow$ $\pi_{\text{HeadEmpNo}}$ (Department)

R2 $\leftarrow$ π_{EmpNo} (Research_Fund)

R3 $\leftarrow$ R1 $\cap$ R2

R4 $\leftarrow$ R1 $\bowtie_{\text{HeadEmpNo=EmpNo}}$ R2



Advantage of Inner Join over Intersection

Inner Join do not need to have union-compatibility, existence of common-columns is enough.

Inner Join could find data in other-columns other than the common columns.

[2018-19-(a) vs (c)] $\leftarrow$ No need to project columns for union compatibility, just join using common columns.

Natural Join $\rightarrow$ Same as Inner-Join, Remove one of the Duplicated Common Column.
("workdep" Or "depno") , ("head" Or "empNo")

PP 2018-19

Syntax Errors

Error : Department does not have "EmpNo", it must be "HeadEmpNo"

(c) HEADS(EmpNo) $\leftarrow$ $\pi_{\text{HeadEmpNo}}$(Department $\bowtie_{\text{EmpNo=EmpNo}}$ Research_Fund)
RESULT $\leftarrow$ π_{Name}(HEADS $\bowtie_{\text{EmpNo=EmpNo}}$ Lecturer)

No Error : HEADS has 1 column, it was taken from the "HeadEmpNo" and Renamed as "EmpNo"

If the above error corrected, this will be correct. Because Inner-Join results matchings, same as intersection, "EmpNo" that is common for both Department Heads and Who get Research Funds.

Logical Errors (b) and (d)

Compare (a), (b) and (d).

We need Intersection (Presents in Both), not Union or Difference.

Inner-Join as Intersection $\leftarrow$ Common for both Department and Research Funds

Compare (a) with (e)

PP 2018-21

Syntax Errors

Compare (a), (b) and (c) $\leftarrow$ Operands are not Union Compatible

They use difference and union that must have union compatibility (number and type of columns).

Logical Errors

Compare (d) and (e)

A–B? Or B–A? , They are not same, Difference is not commutative like Union and Intersection.

Question asks, "find the names of Lecturers who have **no** research fund"

Data Manipulation in SQL-Sub Query Fashion, Joined Fashion and Relation Algebra

PP 2014-10

10. Which of the following relational algebra expression(s) can be used to find the colours of boats reserved by Gihan?

(a) $\pi_{colour}\ (\sigma_{sname="Gihan"}\ Sailor)$
(b) $\pi_{colour}\ ((\sigma_{sname="Gihan"}\ Sailor \bowtie Reservation \bowtie Boat))$
(c) $\pi_{sname="Gihan"}((\sigma_{colour} Boat) \bowtie Reservation)$
(d) $\pi_{sname}\ ((\sigma_{colour}\ Boat) \bowtie \pi_{sid}\ Reservation \bowtie \pi_{sname="Gihan"}\ Sailor)$
(e) $\pi_{colour}\ ((\sigma_{sname="Gihan"}\ Sailor\ ) \bowtie Reservation \bowtie Boat)$

"Find the colors of boats reserved by 'Gihan'"

SQL-Sub-Query Fashion	SQL-Joined-Fashion	Relational Algebra
Select color 3 From Boat Where bid In Select bid 2 From Reservation Where sid = Select sid 1 From Sailor Where sname = "Gihan" Sub Query 1 will return a Single Value of a Single Column, so connector is "=" Or "!=" Sub Query 2 will return a Multiple Values of a Single Column, so "In" Or "Not In" If a Sub Query return multiple columns then use "Exists" and "Not Exists"	Select b.color From Sailor s, Boat b, Reservation r Where s.sid = r.sid And b.bid = r.bid And s.sname = "Gihan" Join Conditions 2 for 3 Tables 3 for 4 Tables Selection Condition	R1 $\leftarrow \sigma_{sname='Gihan'}$(Sailor) R2 $\leftarrow \pi_{sid}$(R1) R3 $\leftarrow$ R2 $\bowtie_{sid=sid}$ Reservation R4 $\leftarrow \pi_{bid}$(R3) R5 $\leftarrow$ R4 $\bowtie_{bid=bid}$ Boat R6 $\leftarrow \pi_{color}$(R5) Closure above steps into a single sentence. R6 $\leftarrow$ R5(R4 (R3 (R2 (R1))))

Syntax Errors

(a) Sailor table do not have a column "color" to be projected

(c) Boat and Reservation tables do not have a column "sname", Predicate for Selection, Columns for Projection

Logical Errors

Request colors not "sname", (c) and (d)

Efficiency of various procedures

Both (b) and (e) correct. But (e) is more efficient.

How? Suppose Sailor has 100 rows, Boat has 20 rows and Reservation has 80 rows.

(b) Join all tables together and thereafter select rows related to “Gihan” → 100x80x20 = 160000
(e) Select the row related to “Gihan” and then Join Tables → 1x80x20 = 1600

Both RAM and Processor is free from the hardworking during the procedure in (e), hence (e) is more efficient than (b), It is the duty of “Query-Optimizer” to select this for the “Command-Processor”.

PP 2014-11

All are correct. But (I) is more efficient than (III).
In the (I), Selection is done before Join and then number of row-concatenation will be low.

PP 2014-12

SQL-Sub-Query Fashion	SQL-Joined-Fashion	Relational Algebra
Select sname **3** From Sailor Where sid In Select sid **2** From Reservation Where bid In Select bid **1** From Boat Where color = “Red” Sub Query 1 will return a Single Value of a Single Column, so connector is “=” Or “!=” Sub Query 2 will return a Multiple Values of a Single Column, so “In” Or “Not In” If a Sub Query return multiple columns then use “Exists” and “Not Exists”	Select s.sname From Sailor s, Boat b, Reservation r Where s.sid = r.sid And b.bid = r.bid And b.color = “Red” Join Conditions 2 for 3 Tables 3 for 4 Tables Selection Condition	R1 ← $\sigma_{color='Red'}$(Boat) R2 ← π_{bid}(R1) R3 ← R2 $\bowtie_{bid=bid}$ Reservation R4 ← π_{sid}(R3) R5 ← R4 $\bowtie_{sid=sid}$ Sailor R6 ← π_{sname}(R5) Closure above steps into a single sentence. R6 ← R5(R4 (R3 (R2 (R1))))

Syntax Errors

(a) “sid” is not available in “Boat” table, Sides of the Union are not Union compatible (sid vs sname)
(c) “sid” is not enough from the Sailor as the output need “sname”, so both “sid”, “sname” must be projected from “Sailor”. “sid” is not enough from “Reservation”, both “sid”, “bid” must be protected from “Reservation” to be joined with both “Sailor-sid-sid” and “Boat-bid-bid”.
(e) Projection must have set of columns, Selection must have predicate/condition, misuse operations

Efficiency of various procedures

Both (b) and (d) are correct, but (b) is more efficient than (d) as it select before join.

PP **Inner Join**

2012-23 ☐ ☐ ☐	2012-21 ☐ ☐ ☐	2012-29 ☐ ☐ ☐	2018-28 ☐ ☐ ☐	2017-32 ☐ ☐ ☐
2018-19 ☐ ☐ ☐	2018-21 ☐ ☐ ☐	2009-16 ☐ ☐ ☐	2009-18 ☐ ☐ ☐	2017-21 ☐ ☐ ☐
2017-23 ☐ ☐ ☐	2014-10 ☐ ☐ ☐	2014-11 ☐ ☐ ☐	2014-12 ☐ ☐ ☐	2016-11 ☐ ☐ ☐
2016-12 ☐ ☐ ☐	2015-20 ☐ ☐ ☐	2013-15 ☐ ☐ ☐	2010-16 ☐ ☐ ☐	2008-11 ☐ ☐ ☐
2008-12 ☐ ☐ ☐	2008-13 ☐ ☐ ☐	2008-14 ☐ ☐ ☐	2007-14 ☐ ☐ ☐	2007-05 ☐ ☐ ☐

Inner Join vs Outer Join

Inner Join must be performed first. All unmatching rows will be eliminated.

R1 ← Employee $\bowtie_{workdep=depno}$ Department (Narmada **will not** be included in R1 ← Unmatching)
R2 ← Employee $\bowtie_{empno=head}$ Department (Examination **will not** be included in R2 ← Unmatching)

Include unmatched rows ones in the output either from one of the tables or both tables depending on what outer join is performing? Left-Outer, Right-Outer, Full-Outer.

Left Outer
Unmatched rows of the Left-Side table will be included once in the output of the Inner-Join.
Right-Side columns of the output table will have null values for the un-matched rows.

R1 ← Employee ⟕ $_{workdep=depno}$ Department (Narmada **will** be included in R1)

Right Outer
Unmatched rows of the Right-Side table will be included once in the output of the Inner-Join.
Left-Side columns of the output table will have null values for the un-matched rows.

R2 ← Employee ⟖ $_{empno=head}$ Department (Examination **will** be included in R2)

Full Outer
Unmatched rows of the both Left-Side and Right-Side tables will be included once in the output of the Inner-Join.
Both Right-Side and Left-Side columns of the output table will have null values for the un-matched rows.

R2 ← Employee X $_{empno=head}$ Department

Employee

empName	proNo
Nimal	1
Kalum	2
Sunil	2
Bimal	4
Dimuthu	5

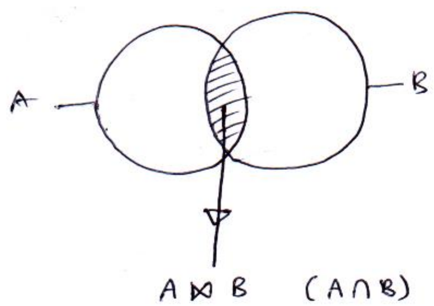
Project

proNo	proLocation
1	Colombo
2	Gampaha
3	Galle

1. Inner Join Include only Matched Records in the result table.

R1 = Employee $\bowtie_{Employee.proNo\ =\ Project.proNo}$ Project

empName	proNo	proLocation
Nimal	1	Colombo
Kalum	2	Gampaha
Sunil	2	Gampaha

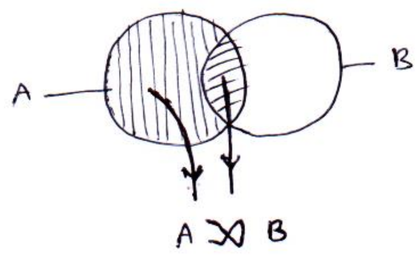


2. Left Outer (Left Join = Left Outer Join)

All Records of the Left Table will be included whether they are matched in common columns or not.

R2 = Employee $\rtimes_{\text{Employee.proNo = Project.proNo}}$ Project

empName	proNo	proLocation
Nimal	1	Colombo
Kalum	2	Gampaha
Sunil	2	Gampaha
Bimal	**Null**	**Null**
Dimuthu	**Null**	**Null**



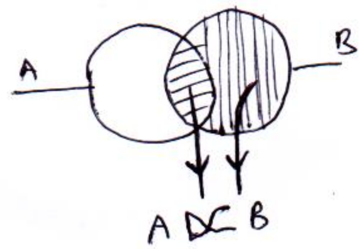

```
Select  empName, proNo, proLocation From  Employee Left Join Project On project = proNo
```

3. Right Outer (Right Join = Right Outer Join)

All Records of the Right Table will be included whether they are matched in common columns or not.

R3 = Employee $\ltimes_{\text{Employee.proNo = Project.proNo}}$ Project

empName	proNo	proLocation
Nimal	1	Colombo
Kalum	2	Gampaha
Sunil	2	Gampaha
Null	**3**	**Galle**



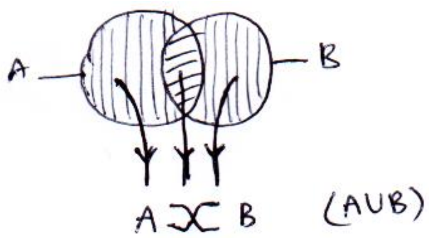

```
Select  empName, proNo, proLocation From Employee Right Join Project On project = proNo
```

4. Full Outer

All Records of the both tables will be included whether they matched in common columns or not.

R4 = Employee X $_{\text{Employee.proNo = Project.proNo}}$ Project

empName	proNo	proLocation
Nimal	1	Colombo
Kalum	2	Gampaha
Sunil	2	Gampaha
Bimal	**Null**	**Null**
Dimuthu	**Null**	**Null**
Null	**3**	**Galle**



**Null Values in the output of the “Outer Join”.

Right Outer → Null Values for un-matchings are in the Left-Side Columns.
Left Outer → Null Values for un-matchings are in the Right-Side Columns.
Full Outer → Null Values for un-matchings are in Both-Side Columns.

PP **Outer Join**

2013-14 ☐ ☐ ☐ 2018-20 ☐ ☐ ☐ 2009-17 ☐ ☐ ☐ 2017-22 ☐ ☐ ☐
2016-13 ☐ ☐ ☐ 2013-16 ☐ ☐ ☐ 2015-14 ☐ ☐ ☐ 2017-44 ☐ ☐ ☐
2018-39 ☐ ☐ ☐

Universal Quantifier – All
Division

Consider the following table

Module

modno	modname	credit
1	PRO–1	4
2	DB–1	4
3	SAD	3
4	MC–1	3

Exam

stuno	module	date
123	1	2019.08.25
123	2	2019.09.01
120	**1**	2019.09.02
120	**2**	2019.09.06
120	**3**	2019.09.06
120	**4**	2019.09.08
126	2	2019.09.02
126	3	2019.09.06
126	4	2019.09.06
124	1	2019.09.08
118	**1**	2019.09.02
118	**2**	2019.09.10
118	**3**	2019.09.12
118	**4**	2019.09.12

Suppose you need to find numbers of Students who apply for **All Modules**

Then You need to division

R1 ← Exam / Module

But for division, Divider must have a smaller number of columns than the Dividend.
Dividend must have the same type of columns that the Divider has. Names are not need to be same, but types.

So, R1 ← Exam / Module (This could not be done)

You can project needed columns and then devide,

$$\underset{\text{(1Col)}}{R1} \leftarrow \underset{\text{(2Col)}}{\pi_{\text{module, stuno}} (\text{ Exam })} \;/\; \underset{\text{(1Col)}}{\pi_{\text{modno}} (\text{ Module })}$$

If there are any row value matched for all rows of the divider, that values (120 and 118) will be resulted.
Column "module" will be removed by "modno" during the division.

stuno
120
118

PP **Division**

2010-17 ☐ ☐ ☐ 2013-17 ☐ ☐ ☐ 2009-15 ☐ ☐ ☐ 2006-21 ☐ ☐ ☐
2016-14 ☐ ☐ ☐

Conclusion to Relational Algebra
Classification

Procedural vs Declarative and Relational Calculus

2018-01 ☐ ☐ ☐ 2009-05 ☐ ☐ ☐ 2017-03 ☐ ☐ ☐ 2014-15 ☐ ☐ ☐
2012-19 ☐ ☐ ☐ 2011-37 ☐ ☐ ☐ 2010-05 ☐ ☐ ☐

Classification

Basic vs Derived

Basic → Set Difference, Selection, Projection, Cartesian Product
Derived → Intersection(Using Difference), Join(Using Product), Division(Using Product)

2018-15 ☐ ☐ ☐ 2008-09 ☐ ☐ ☐ 2006-19 ☐ ☐ ☐

Common for RA and Mathematics Set Theory vs Specific for RA

Common for RA and Set → Union, Intersection, Cartesian Product, Set-Difference
Specific for RA → Selection, Projection, Join, Division

2009-14 ☐ ☐ ☐ 2007-08 ☐ ☐ ☐

Mathematical Evaluation of RA-Operation

2013-12 ☐ ☐ ☐ 2015-23 ☐ ☐ ☐ 2016-09 ☐ ☐ ☐ 2016-10 ☐ ☐ ☐
2014-14 ☐ ☐ ☐ 2012-20 ☐ ☐ ☐ 2010-15 ☐ ☐ ☐

SQL with RA

2015-24 ☐ ☐ ☐ 2013-13 ☐ ☐ ☐ 2010-24 ☐ ☐ ☐ 2010-18 ☐ ☐ ☐
2012-22 ☐ ☐ ☐

****