Relational AL Debrag: It's a procedural query language, which takes relation as input and generates relation as output

Types of operation

(1) Basic/Fundamental operation | (2) Additional / Derived operation

SET Differences

* It allows us to find tuples (row) that are in one relation but are not in other.

$r - s$

(1) r and s must have same arity
(2) attribute domains of r and s must be compatible.

* Find all the courses thought in Fall 2017 but not in Spring 2018

Π Course_ID (σ Semester = "Fall" $\wedge$ Year = 2017 (Section))
$-$ Π Course_ID (σ Semester = "Spring" $\wedge$ Year = 2018 (Section))

Chapter one:

18-01-24

Database System

* Data definition Language (DDL)

```
Create table table_name {
    Variable_name Datatype (),
    "          "      "  "
);
```

Stored in data dictionary.

* Data manipulation Language (DML)
  * Procedural DML
  * Declarative DML
* Sql query language
* Database Architecture.
* Database application Architecture
* 3 tier architecture and 2 tier architecture. Which is best? Difference 3 tier archi: best / most secure.

* Database Users,
* কয় প্রকার database থাকে কয়.

* DDL , DML এর Difference.

* Database Application কোথায় কোথায় ব্যবহার হয়েছে

Additional / Derived operation

① Join (⋈) Binary

② Division (÷) → all select

③ Set intersection (∩)

25.01.24

Chapter - 2

Touple → row
Attribute — column.

and = ∧
or = ∨
not = ¬

* Relation schema and instance
Relation — table.
instance → r
Schema → R

Tuples = Row

* Candidate key এগারটির 2টি.

Basic / Funclamental Operation

Row wise
① Select : σ

Column
② Project : π ... display/show — show only distinct
③ Union : ∪
④ Set difference : −
⑤ Cartesian Product : ×
⑥ Rename : ρ

Felen is Projection

Select :

selection ← $\sigma_P(r)$ → relation

P is called the selection Predicate → logic

Projection :

Projection ← $\pi_{attributes}(r)$ → relation

31·01·24
week 2-1

name ($\sigma_{dept\_name = \text{"Physics"}}$ (instructor))

Cartesian Product (A+B)

long form means

```
SELECT *
From instructor, teacher
WHERE instructor.ID = teachers.ID;
```

→ $\sigma_{instructor.ID = teachers.id}$ (instructor × teacher)

used to join 2 table based on some condition. → Both table ID value has to be same only then condition will be fulfilled

Join Operation

$r \bowtie s = \sigma_\theta(r \times s)$

Instructor ⋈ instructor.Id = teacher.id. teacher

* arity = same number of attributes
* Join → attribute হলে Join হয়
* Combine → tupple হলে Join হয়

UNION → set

$r \cup s$
① r, s must have same number of attribute
② same type of datatype

Union ALL = List

Intersect

EXCEPT → set differences

Rename (ρ) :

$\rho(R_1, R)$
R_1 → new
R → existing Relation

Week 3.2 01.02.2025

Select *
FROM instructor,
WHERE dept_name = 'physics' AND salary > 8;

→ Equivalent.

Sample question set (quiz-1)

The following relational schema forms a part of a sports management company database held in a relational DBMS.

Sports (S_ID, S_Name, S_Duration)
Player (P_ID, P_name, S_ID, VID)
Venue (V_ID, V_location, V_Type)

Please write down the relational Algebra for the following queries.

(a) Find the name of the Player having Player ID same as the last 3 digits of your own registration number.

number of first digit 0 থেকে 9

Ans: $\pi_{P\_name}(\sigma_{P\_ID = '095'}(Player))$

Relational Algebra তে ডেটাবেজ 2টা।

(b) Show the Sports name having the Sports_ID same as the last 5 digits of your own registration number.

Ans: $\pi_{S\_name}(\sigma_{S\_ID = '01095'}(sports))$

Project → π → show/display

Previous Mid

The following relational schema from a part of an event management company database held in relational DBMS.

Event (E_ID, E_Name, E_type)

Guest (G_ID, G_Name, E_ID, V_ID)

Venue (V_ID, V_name, E_ID, V_location)

Construct (write down) the relational Algebra.

(a) The guests name having Guest ID as 01023

Ans $\pi_{G\_name}(\sigma_{G\_ID = '01023'}(Guest))$

(c) List the sports IDs where the duration is 90 minutes.

Ans $\pi_{S\_ID}(\sigma_{Duration = '90\ minutes'\ (Sports)})$

(d) Find the venue locations for indoor but not outdoor types.

Ans $\pi_{v\_location}(\sigma_{v\_type = 'Indoor'}(venue))$

$= \pi_{v\_location}(\sigma_{v\_type = 'outdoor'}(venue))$

) Show ven ID for ven located at Dhaka.

Ans $\pi_{v\_ID}(\sigma_{v\_location = 'Dhaka'}(venue))$

(b) The event name having the event type as wedding but not Birthday.

Ans

$\pi_{E\_name}(\sigma_{E\_type = \text{"wedding (Event)}}) - \pi_{E\_name}(\sigma_{E\_type = \text{"Birthday (Event)}})$

(c) The venue IDs where the address is Green road, Dhaka.

Ans $\pi_{V\_ID}(\sigma_{V\_Address = \text{'Greenroad'}}(venue))$

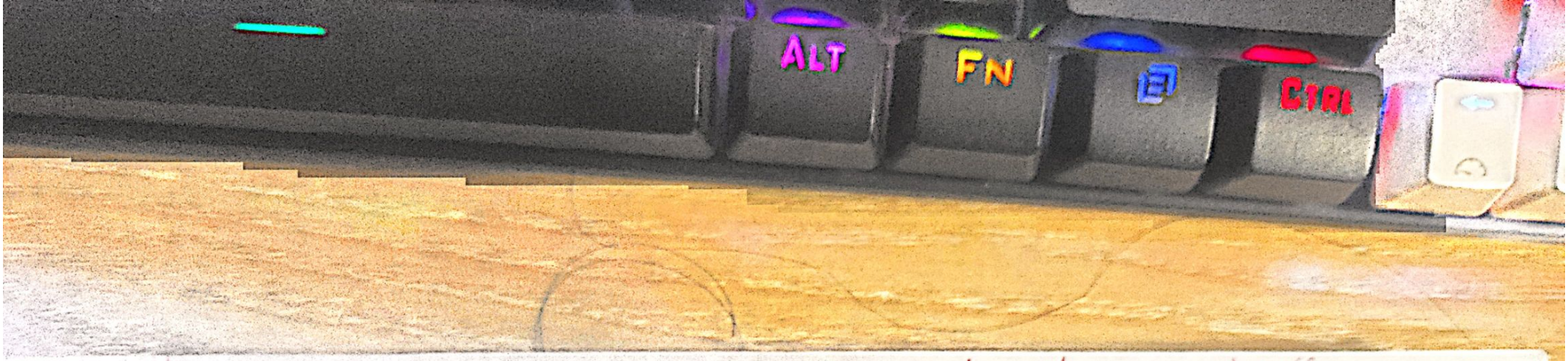
Spring 2023 (Final)

3| Construct (write down) the SqL commands For following queries.

Food (F_ID, F_Name, F_Type, F_Price)
Customer (C_ID, C_Name, F_ID, W_ID)
Waiter (W_ID, W_Name, W_Salary)

a) The food type of most expensive food?

```
SELECT F_Type
From Food
WHERE F_Price = (SELECT max (F_Price)
                 From Food);
```



Join, Cartesian, Union, Intersection, Set Difference, Select

Projection, rename

(15-02-25)

Q1 Answer

a) Account (Ac_ID, Ac_ID, Ac_Balance, Cus_ID)
Customer (Cus_ID, Cus_Name, Cus_Location)
Deposit (Dep_ID, Dep_Amount, Ac_ID)

a) Find ac balance but not Savings

$\pi_{Ac\_Balance}(\sigma_{Ac\_Type = "current"}(Account) - \pi_{Ac\_Balance}(\sigma_{Ac\_Type = "Savings"}(Account)))$

(B) Show customer ID payed at Panthpath, Dhaka

$\pi_{Cus\_ID}(\sigma_{Cus\_location = 'Panthpath\ Dhaka'}(Customer))$

(C) List deposit IDs where amount is more than 50,000

$\pi_{Dep\_ID}(\sigma_{Dep\_Account > 50000}(Deposit))$

left side 0 থাকলে তবে quotation এ হবে

d) Find account IDs having customer ID as '040608'

$\pi_{Ac\_ID}(\sigma_{Cus\_ID = "040608"}(Account))$

e) Show location for customer named Andrew

$\pi_{Cus\_location}(\sigma_{Cus\_Name = "Andrew"}(customer))$

Chapter - 3

SQL Statements

- DDL
  1) CREATE
  2) ALTER
  3) DROP
  4) TRUNCate
  5) RENAME
- DML
  1) INSERT
  2) UPDATE
  3) DELETE
- DCL
  1) GRANT
  2) REVOKE
- TCL
  1) TCL
  2) ROLLBACK
  3) SAVE POINT
- DQL
  1) SEL

Data Types:

1) Char( )
2) Varchar( )
3) Int ( )
4) float

Creat :

(i) CREATE Database;

(ii) CREATE Table table_name (
columnname datatype;
);

(iii) SHOW Databases;

Types of integrity (constraints):

* Primary key
* foreign key
* not null

Truncate is used to delete all data from table
ut table reamains just data done

DROP → Used to delete TABLE
DELETE → used to delete ea row (tuples)

ALTER

(rename)

(i) ALTER TABLE Old-tablename,
TO new_table;

(ii) Add column:
ALTER TABLE table_name,
ADD Columnname datatype;

(iii) Modify :

(iv) Rename column :
ALTER Table table_name,
RENAME COLUMN old_name TO new

(v) Drop Column:
ALTER TABLE tablename,
DROP COLUMN column_name;

Insert:

INSERT INTO tablename (colum1, c2...)

Values (value1, value2);

or,

INSERT INTO tablename VALUES (value1...);

UPDate

UPDATE table_name

Set column1 = value, Val...

WHERE filter condition;

SELECT

(i) SELECT Column_name, Column 2 ....

From table_name

or, SELECT * → it means select all

From table_name;

(ii) SELECT DISTINCT Colum

From table_name

Example:

Select ID, name, Salary/12

From Instructor,

output will be same as instructor relation, except Value of Salary attribute will be Salary/12

To rename it write query,

Select ID, Name, Salary/12 (as) monthSal

WHERE

SELECT column1, column2 ---
FROM table_name
WHERE filter condition

Logical Connectives = and, or, not

Select columns ---
From table_name
where condition 1 and, or, not, other condn...

From

is used for cartesian products.

Select *
From instructor, teachers (ins x teachers)

Find the name of all instructor in Arts dept
Who thought some course and courseID

```
SELECT name, c'd
From instructor, teachers
Where instructor.Id = teachers.idand dept
                                = 'Arts';
```

* we use as clause as rename

```
Select distinct T.name
from instructor as T, instructor as S
```

ins as T = ins T

String operations

NOT LIKE **Like :** The operator like uses patterns that are described using 2 characters

(i) % → it matches any substring
(ii) (_) it matches any character.

```
Select Column
From table name
where Column Like Pat
```

(Ex) (1) Find the name of instructor that includes substring "dar"

```
Select name
From instructor
Where name Live '%dar%'
```

* Match the string "100\%" → backslase used as escape

Example

(I) 'Intro %' → any string beginning with Intro

(II) '%Comp%' → matches any string containing Comp

(III) '___' → matches any string exactly of 3 characters

(IV) '___%' matches any of at leas 3 char

_ represent each chrac

_a → Here a is 2nd char

__a → " " 3rd "

Order by :

Between :

Anything + NULL returns NULL

Count() :

```
SELECT count(*)
From employee.
```

```
SELECT count(salary)
From employee ;
```

Count দিয়ে column নাম দিলে Null value ignore করে.

Group by: Groups rows that have the same values into summary rows, like "Find the number of employes.

Syntax:

```
SELECT  column_name
From    table_name,
WHERE   filter condition.
Group By  column name
Order By  column name
```

Group BY এর সাথে Having
,, এর মতো WHERE

Subquery: