Relational Algebra: 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.

SET Differences

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

$\boxed{r - s}$

(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

$\pi_{Course_ID}(\sigma_{Semester = "Fall" \wedge year = 2017}(section))$

$- \pi_{Course_ID}(\sigma_{Semester = "Spring" \wedge year = 2018}(section))$

Chapter one:

12-01-24

Database System

* Data defination Language (DDL)

Create table table_name {

variable_name Datatype (),

" " "

);

Stored in data dictonary.

* Data manipulation Language (DML)
  * Procedural DML
  * Declarative DML

* SQL query language.
* Database Architechure.
* Database application Architechure.

* 3 tier architechure and 2 tier architec

Which is best? Difference 3 tier arch.

best / Most secure.

* Database users,
* কয় প্রকার database এনের কাজ.

* DDL , DML এর Difference.
* Database Application কোথায় কোথায় কাজে লাগে

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 এগুলোকে বলে।

Basic / Fundamental Operation

Row wise — ① Select : σ ... display/show — show only dis

Column → ② Project : π

③ Union : ∪

④ Set difference : −

⑤ Cartesian Product : ×

⑥ rename : ρ

Felen is Projection

Select:

$\sigma_P(r)$ — selection; r → relation; P → logic

P is called the selection Predicate

Projection:

$\Pi_{attributes}(r)$ — Projection; r → relation

31.01.24
week 3.1

$name(\sigma_{dept_name = "physics"}(instructor))$

Cartesian Product $(A \times B)$

SELECT *
From Instructor, teacher.
~~wher~~ WHERE Instructor.ID = Teachers.ID;

$\sigma_{Instructor.ID = teachers.id}(instructor \times teacher)$

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

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

UNION → set

$r \cup s$

① r, s must have same number 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);

Equivalent.

Sample question set (quiz-1)

The following relational schema form 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.

number er first digit 0 হলে , 1 নিতে হবে

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

Ans: $\pi_{P_name}(\sigma_{P_ID = '095'}(Player))$

Relational Algebra তে সেন্সিটিভ নয়।

(b) Show the sports name having the sports same as the cost 5 digits of your own registration number.

Ans $\pi_{S_name}(\sigma_{S_ID = '01095'}(Sports$

(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: $\pi_{v_ID}(\sigma_{v_location = 'Dhaka'}(venue))$

Project → π → show/display

Previous mid

The following relational schema from a
of an event management company data
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_~~ID~~)

Construct (write down) the relational

(a) The guests name having guest ID

Ans: $\pi_{G_name}(\sigma_{G_ID = '0102}$

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

Ans

$\pi_{E_name}(\sigma_{E_type} = \text{"Wedding"}(Event))$ –

$\pi_{E_name}(\sigma_{E_type} = \text{"Birthday"}(Event))$

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_
                 From Food);
```

b) The customer names with the names

```
SELECT C_Name, F_Name
From Customer, Food.
Where, C_ID = F.F_ID;
```

c) The waiter name with the lowest salary

SLECT w_name
From waiter.
where w_Salary = (Select min (w_Salary)
From waiter)

d) The ~~Ra~~food Prices of the food names starting with 'B' and ending with 'r'

SELECT F_Price
From Food
where F_name Like 'B% r.';

2) The customer ID's where the food ID is 200 but not 300

SELECT C_ID WHERE F_ID = 200
From Customer Except SFlect C_ID From customer Where F_ID = 300);

(15-02-25)

CT1

2) 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 saving,

$\pi_{Ac_Balance}(\sigma_{Ac_Type} = \text{"current"}(Account) - \pi_{Ac_Balance}(\sigma_{Ac_Type} = \text{"Saving"}(Acount))$

(B) Show customer ID layed at Panthpath, Dhaka

$\pi_{Cus_ID}(\sigma_{Cus_location} = \text{'Panthpath Dhaka'}, (customer))$

c) List deposit IDs where amount is more than 30,

$\pi_{Dep_ID}(\sigma_{Dep_Account} > 30000 (Deposit))$

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) SE

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 but table remains just data gone

DROP → Used to delete TABLE

DELETE → used to delete row (tuples)

Rename ALTER

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

(ii) Add column:
ALTER TABLE table-name,
ADD column name datatype;

(iii) Modify:

(iv) Rename column:
ALTER Table table-name,
RENAME COLUMN old-name TO new

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

Insert:

INSERT INTO tablename (colum1, c2..)

Values (value1, value2);

or,

INSERT INTO tablename VALUES (value1 ...);

UPDate

UPDATE tablename.

Set column1 = value, Value...

WHERE filter condition;

RENAME Table:

RENAME TABLE old-tablename TO new-tablenam.

ELETE:

DELETE from table_name,

here

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.

Value of Salary attribute will be Salary

To rename it write query,

Select ID, Name, Salary/12 as monthSal

WHERE

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

Logical Connectives = and, or, not

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

From

is used for cartesian products,

Select *
From instructor, teachers (ins x teacher)

d the name of all instructor in Arts dept
o thought some course and course ID

SELECT name, cid
From instructor, teachers,
where instructor. Id = teachers.id and dept = 'Art

* We use as clause as rename

Select distinct T. name,
from instructor as T, instructor as S (ins as T = ins

String operations

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

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

Select Column
From table name
where Column like

Exm ① Find the name of instructor that includes substring "dent"

Select name,
From instructor
Where, name Like '%dent%'

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

Example:

① '%intro%' → any string begining with intro

② '%comp%' → matchs any string containing comp

③ '___' → matchs any string exactly of 3 characters.

④ '___%' matches any of at least 3 char

- represent each charac
- a → Here a is 2nd char
- a → " " " 3rd "

Order by :

Between :

Anything + NULL returns NULL

Count( ) :

SELECT count(*)
From employee.

SELECT count
From employee.
Count ignoring column
Null value.

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

Syntax:

SELECT column_name, groupby_name.
FROM table_name.
WHERE filter condition.
Group By column name
Order By column name

Group BY এর সাথে Having
" এর সাথে WHERE

Subquery:

Database user

2) **Naive Users:**

(I) Naive users are those who have minimal understanding of DB or SQL.

(II) They interact with DB with simple interface such as forms or predefined queries,

(III) Their primary goal is to operations like forms and predefined queries.

(IV) They are rely on user friendly interface without needing to know underlying complexity,

Application Programmers:

(I) They are who develop Applications for DB system,

(II) They have intermediate to advance level of knowledge of DB.

(III) Their primary responsiblity is to design develop and maintain.

(IV) They write code to perform Crud,

3) **Sophisticated users:**

(I) They possess advance level of knowledge and skills in DB system & SQL

(II) This data analysts, business analysts

(IV) Proefficient in writing sql codes,

(V) Often leverage tool as dot VS.

4) **DBA:**

(I) responsible for managing and ma

DB system.

(II) have in-depth knowledge.

(IV) implementing control,

(V) Plans crucc in capacit scaling.

* যে table গুলো সব থেকে বেশি Dependent গুলো আগে Delete করবেন

* Create এর সময় independent আগে Create করবেন

Fall - 23 mid

14.03.24

Our mid Solutions

1) a) $\pi_{s_name}(\sigma_{year = '2nd' \wedge semester = '2nd'}(stud$

b) $\pi_{d_name}(\sigma_{floor = '7th'}(department))$

c) $\pi_{c_code}(\sigma_{c_title = 'database system'}(course))$

2) (a)
```
CREATE Department (
    Dept_id varchar (10),
    Dept_name varchar (20),
    Primary KEY (Dept_id),
);
CREATE TABLE Project(
    Project_id varchar (10),
    Project_name varchar (20),
    PRIMARY KEY (Project_id);
);
```

4 ACID Properties in transaction management within DB system:

Atomicity: it ensures that a transaction is treated as a single unit of work, meaning that either all operations are completely done or none of them are. If a part of transaction fails entire transaction rolled back. This property ensures data consistency and integrity.

Consistency: Consistency guarantees that the DB remains in a consistent state before and after transaction. Any changes made by a transaction must adhere to all defined rules & constraints to preserve consistency of DB preventing entering invalid state.

Isolation: it ensures that the execution of multiple transactions concurrently produces same result as if the transactions were executed sequentially, one after another. It prevents transaction from interfering with each other while they are in progress. By maintaining isolation it prevents issues such as dirty reads, non-repeatable reads.

Durability: it guarantees that once a transaction has been commited the changes it made to the DB persist even in the event of system failure such as power outages, crashes or hardware failure. These changes are recorded permanently and durable manner. Typically through techniques like write ahead logging or journaling. Even if the system crashes after a transaction is commited, the DB can recover its state by replaying the logged changes ensuring data durability and reliability.

Explain Logical, Physical schema, Physical Data independence in DB system.

Logical schema:

(I) represents logical structure of entire DB it defines the organization of data element and relationship among them.

(I) Describe what data stored in DB and how its related, without concerning itself.

(II) includes, column, tables, keys.
ER (entity relationship)

Physical Schema:

(I) Shows how data is physically stored in DB. Deals with aspects such storage structure, indexing mechanism, access method.

(II) logical schema conceptual, representation of Data. Physical schema is concerned with actual implem details storage level.

(III) includes, file organisation, indexing method, stor allocation data access path.

Instance:

(I) refers to actual Data stored in the D at a particular moment in time.

(II) represents snapshot of DB contents including Column of each table and relation between the

(iii) every change in DB insertion, deletion.

(IV) instances are dynamic and can change over time as transaction are executed against the DB.

Physical Data Independence:

(I) is a property of DB that allows changes to the physical schemas without affecting the logical schema or application program that access data.

(II) it enables DB administrations to modify the physical storage structures indexing method, or access paths without requiring changes to logical schema or the application code.

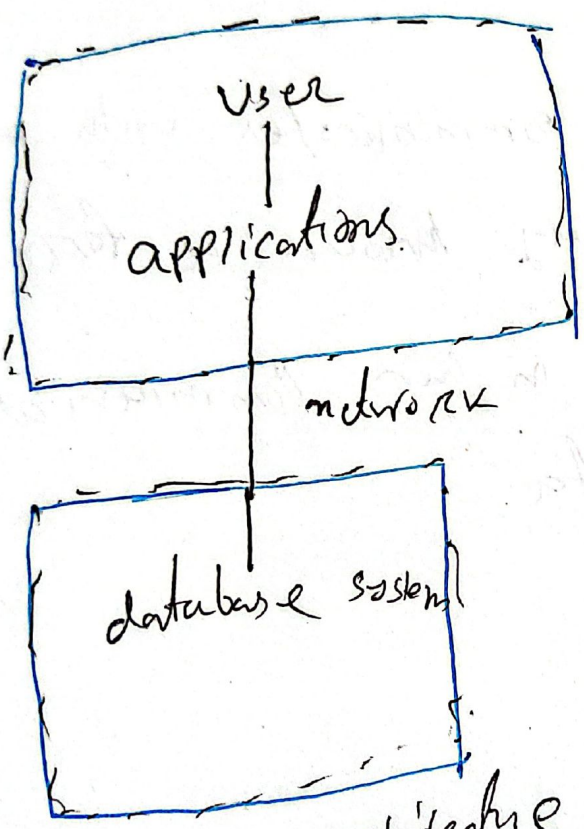
(III) The seperation between the logical and schema provides flexibility scalability and performance optimization in DB system.

(V)

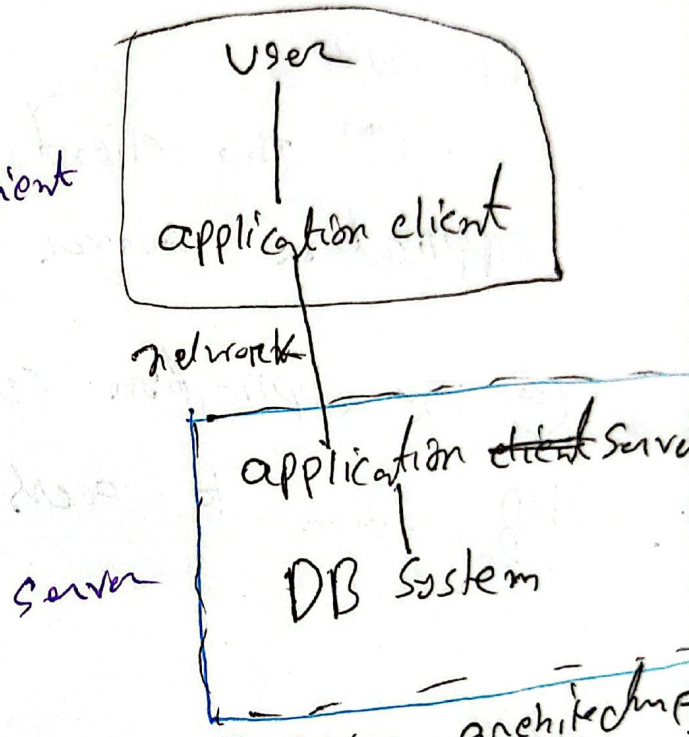
logical schema defines the conceptional structure of DB. The physical schema defines how data is stored and accessed. Physically an instance represents the actual data stored in the DB at a specific point in time and physical data independence allows changes to the physical schema without affecting the logical schema or applications.

2-tier 3-tier

Explain the two tier and three tier architecture of DB applications with proper example



a) 2 tier architecture



(b) - 3 tier architecture

Two-tier architecture; the application resides at the client machine where it invokes DB system functions at the server machine

3 tier architecture:

Discuss DB applications examples in different s
such as enterprise information, banking and fin
Universities airlines telecommunication and navig
systems

1) Enterprise information systems;

• Enter Price Ressource Planning (ERP) systems in
Core business Processes such as finance, HR
management and supply chain operations
centralized DB for streamlining workflows
improving efficency!

Customer Relationship management (CRM):

CRM system stores and manage customer data, interactions and preference. They help business. track sales, manage marketing campaigns and improve customer services.

HRMS:

This system handles employee data payroll. Procu. They assist organization in hire managing keeping workforce effectively.

) Banking:

Core banking systems: Bank use DB to store Customer account information, transaction records and other financial enabling efficient banking operations

Risk management systems: Risk management system analyze financial data to assess and mitigate risks associated with investments loans, market flow

Trading platforms:

3) Universities:

Student information system (SIS): It maintains stude records schedules, grades, They facilitate effi Communication between students faculty adm

Library MS: used to catalog books, journals. Resources, enabling efficient search retrieval b librarians.

4) Airlines:

- Reservation systems: manages flight book Seat assignments, ticket sales. They interact
- Crew scheduling:

Maintanance: aircraft maintanance record,

2) Telecommunications:

- Customer billing system: use DB to store customer information; service plans, usage data, billing record,

- network management

Navigations systems:

Geographic information systems: stores data such as maps, satelite images

Traffic monitoring systems: store realtime traffic data

Fleet management: stores vehicle location, logistics tracking, dispatch information

Fall - 22

1) (a) Acid for ATM Transaction at Bank:

1) Atomicity: it ensures that a transaction is either completed in its entirety or not at all in other words. it guarantees that all operations within a transaction are executed successfully or none of them are executed at all. if any fail it rolls back

Role in scenario: in the context of ATM atomicity ensures that if the ~~transaction are completed successfully~~ ATM transaction follows Principle of atomicity it means either all of its operations (including withdrawing money updating) are completed successfully. or none of them are. if machine fails after you've entered your PIN and clicked ok but before the transaction is completed atomicity ensures that no changes are made to your account balance and the machine return your card without processing withdrawal.

Consistency: it ensures that the DB remains valid state before and after the transaction. For ATM tran consistency ensures that the withdrawal doesn't result in your account balance going below zero or exceeding any withdrawal limit. if machine fails before complete system should ensure your account balance remain same

Isolation: it ensures that intermediate state of a transaction is not visible to other transaction until its completed. For atm isolation ensures that even multiple customer are using atm simultaniously the failure of one transaction doesn't affect the outcome of other transaction. your withdrawal transaction should be isolated from other. its effects

are not visible until it's fully complete.

Durability; it ensures once a transaction is committed it effects persist even in the event of system failure, if atm withdraw money from your account before failing durability ensure withdrawal is permanent and recorded in system, conversely if the machine fails before completing, it ensures no changes are made.

(B)

While withdrawing money and depositing at the same time, ACID properties that plays is Isolation.

Isolation ensures that the transaction are executed in a manner that outcome of each transaction should be independent of other transaction occurring simultaneously.

Isolation: when you are withdrawing money and father is depositing cash simultaneously isolation ensures the effect of your withdrawal transaction don't interfere with the effects of your father deposits and vice versa. if your father deposit is updating the available balance in his account this update shouldn't be visible to you withdrawal until its done vice versa with father.

Spring 22

DB Architecture

Centralized Database:

- Data is stored in a single location. typically on a single server.
- Data controlled centrally, use directly interact in DB server,
- Simple to manage and maintain,
- Administration, management performed centrally

Client-Server:

- DB is split between client, application, serv
- Client application handle user interface, while server manage data storage, processing.
- Client communicate with server over n request or submit data. Server process and return results.

- Client server DB offer scalability as multiple client can access the same server.

Parallel Database:

- Utilize multiple processor/server to perform data processing task.
- Data divided into smaller chunks. Each processor works on subset of data in parallel.
- designed to improve performance and scalability.

well suited for application with heavy computing demands.

Summarize the atomicity and concurrent access issues solved by DB

Atomicity:

- ensure transaction treat as single entity
-
- enforce atomicity through transaction management, live transaction log, rollback
- if transaction fails midway, the DB system rollback the transaction to its original state

Concurrent Access:

- refers multiple users or process accessing and modifying DB simultaneously
- DB handle concurrent access through using multiversion, concurrency control, MVCC
- locking mechanism prevent conflicting operation from interfering each other by temporarily restricting access to resources
- MVCC allows multiple transaction to read and write by maintaining multiple versions of data

By addressing atomicity and concurrent access issues, DB maintain data consistency, integrity, reliability even in multi user environment with simultaneous data modifications. These mechanism ensure transactions executed reliably and DB remain consistent state regardless of concurrent activity

Normalization

(I) First normal Form:

a) Contains only atomic value

b) There are no repeating groups.

(II) 2nd normal Form:

a) it is in normal form

b) All non-key attributes are fully function dependent on the Primary key.

(III) 3rd normal form:

a) it is in 2nd normal form

b) There are no transitive dependenc

Delete Anomaly:

1) This anomaly occurs when deleting data from the database unintentionally deletes other unrelated data.

2) Consider a table that contains information about projects & employee assigned to them. If you delete project from table then you might also remove info about the employees who were only associate with that project.

Deletion anomaly can result in the loss of valuable info that can impact the integrity and quality of DB.

Update anomaly:

1) This anomaly occurs when data redundancy leads to inconsistencies that that arise when updating info in a DB.

2) If you have same info in multiple places and you update in one place but forget to update it

3) This anomalies can lead to confusion and inaccuracies in DB compromising integrity.

Insertion anomaly

1) This anomaly occurs when you can't add data into DB without adding additional unrelated data.

for example,

(i) if you have a table where you store info about employees and their projects. If any employee do currently have any project assigned you can't a