

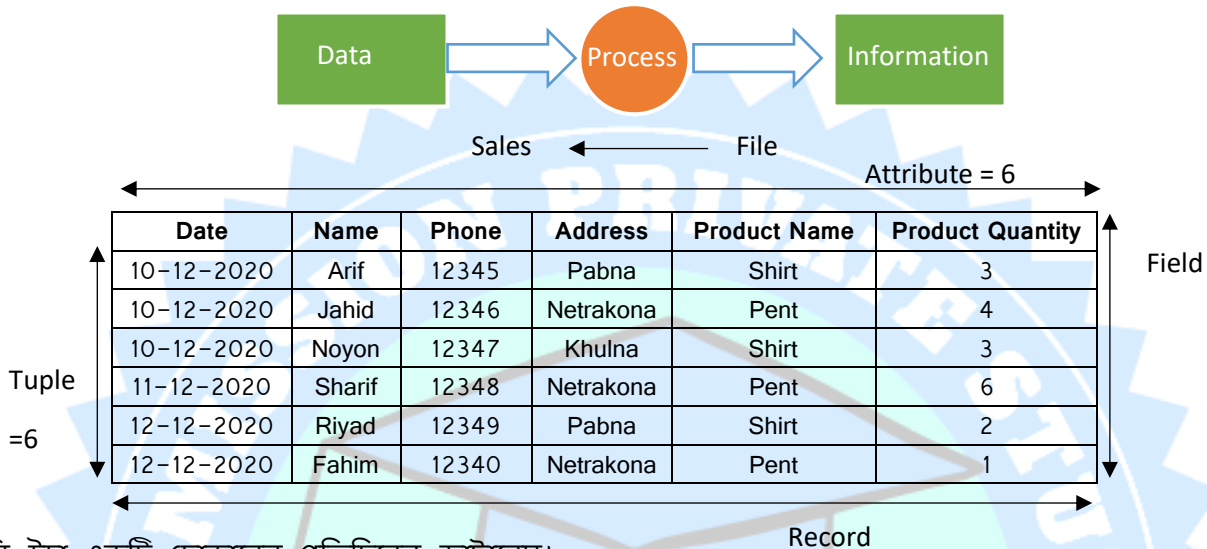
# PRIVATE STUDY ZONE

## Database Management System

Asmaul Haque (CSE, DUET)

Md Jahirul Islam (CSE, DUET)

**Data & Information:** Data শব্দটি হল Datum এর বহুবচন। Datum অর্থ হচ্ছে তথ্যের উপাদান। ডাটা হচ্ছে ঐ সমস্ত উপাদান বা উপকরণ যাদের প্রক্রিয়া করলে ইনফরমেশন পাওয়া যায় অর্থাৎ ইনফরমেশনের ক্ষুদ্রতম উপাদান হচ্ছে ডাটা। ডাটাকে প্রসেস করলে যা (আউটপুট) পাওয়া যায় থাকে ইনফরমেশন বলে।



ধরি ইহা একটি দোকানের প্রতিদিনের ডাটাবেস।

✓ 10-12-2020 মোট কতগুলি পণ্য বিক্রি হয়েছে? উঃ  $3+4+3=10$  টি।

এখানে মূলত ডাটাগুলোকে প্রসেস করে ইনফরমেশন বানানো হয়েছে। অর্থাৎ টেবিল ডাটা আছে যদি আমরা সেই ডাটাগুলোকে প্রসেস করি তাহলে ইনফরমেশন পাবো।

**Attribute:** Column header.

**Tuple:** Each Row.

**Degree:** Total number of attribute.

**Cardinality:** Total number of tuple or row.

**Meta Data:** ডাটা সম্পর্কিত ডাটাকে মেটা ডাটা বলে।

**Field:** একই টাইপের column কে ফিল্ড বলে। ফিল্ড হলো রেকর্ড এর ক্ষুদ্রতম অংশ। Related Record থাকবে।

Name
Arif
Jahid
Noyon
Sharif
Riyad
Fahim

Name কলামে শুধু String type এর ডাটা আছে।

Note: জমিতে একই রকমের ফসল চাষ করা হয়।

**Record:** কতগুলো ফিল্ড এর সমষ্টিকে রেকর্ড বলে। (Row)

10-12-2020	Noyon	12347	Khulna	Shirt	3
------------	-------	-------	--------	-------	---

Note: একজন ব্যক্তি সম্পর্কে আইডিয়া।

**File:** একক unit এ ব্যবহৃত পরস্পর সম্পর্কিত রেকর্ডের সংগ্রহকে ফাইল বলা হয়। [A collection of related record used as a single unit is called file.]

**ডাটার প্রকারভেদ:** ডাটা প্রধানত তিন প্রকার।

- Numerical Data: Integer, Float, Decimals, Complex etc.
- Textual Data: Information.
- Graphical Data: Bar graph, the pie chart etc.

এছাড়াও কিছু ডাটা টাইপ আছে যা ডাটাবেস এ ব্যবহৃত হয়।

- Logical Data: 0/1, true/false, yes/no etc
- Currency.
- Data & Time.
- Memo type.

**Database:** ডাটা শব্দের অর্থ উপাত্ত এবং Base শব্দের অর্থ হচ্ছে ঘাঁটি বা সমাবেশ। Database হলো বিভিন্ন সম্পর্কিত বিষয়ের উপর ব্যাপক উপাত্তের সমাবেশ।

**DBMS(Data Management System):** DBMS মূলত একটি সফটওয়্যার সিস্টেম যা ডাটাবেস তৈরি ডাটা সংরক্ষণ পরিবর্তন নিয়ন্ত্রণ ও পরিচালনা ইত্যাদি কাজে ব্যবহার করা হয়। DBMS 4 প্রকার।

- Hierarchical DBMS
- Relational DBMS
- Inverted DBMS
- Network DBMS

**Database Management System এর উদ্দেশ্য:**

- Data Redundancy(অপ্রয়োজনীয়তা):** একই ফাইল বিভিন্ন স্থানে হুবুহু জমা থাকলে এ সমস্যা দেখা দেয়। DBMS এ বিভিন্ন স্কিমা ব্যবহার করে এ সমস্যা দূর করা হয়।
- Inconsistency(অসংলগ্নতা):** একই নামের একই ফাইল বিভিন্ন স্থানে হুবুহু আছে। যদি একটির পরিবর্তন করা হয় অপরটি অপরিবর্তিত অবস্থায় থাকে। DBMS এ সমস্যা হয় না।

214105	Rimon	Netrakona
--------	-------	-----------

214105	CSE	Netrakona
--------	-----	-----------

- Security:** DBMS এ যদি এক্সেস পার্মিশন থাকে শুধু তাহলেই Data Read/Write করা যায়।
- Atomicity:** System এর কার্যকারিতা হারিয়ে ফেললে জমাকৃত ডাটার ব্যাকআপ ও রিকভারি ব্যবস্থা অটোম্যাটিক এবং ডাটা ট্রানজ্যাকশন অটোম্যাটিক হওয়ার ব্যবস্থা রয়েছে।
- Data Isolation:** File Format এর ভিন্নতা এবং ডাটা ছড়ানো ছিটানো অবস্থায় থাকে না।
- Integrity Problem:** ফাইল ফরম্যাট ভিন্ন ভিন্ন না হওয়ায় consistency constraints পালন করার জন্য কোডে লেখা খুব সহজ।

**DBMS Package:** Dbase, Sybas, Informix, MySQL, Oracle, Access, Foxbase, Microsoft SQL Server etc.

**DBMS এর অপারেশন/কাজসমূহ:** i. Create Database ii. Append iii. Debug iv. Delete v. Edit vi. Search vii. Sort viii. Print ix. Save x. Security xi. Update.

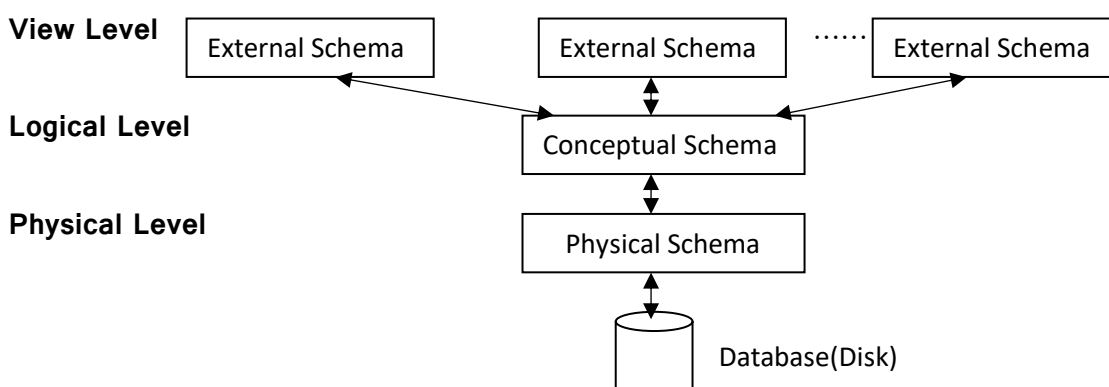
**DBMS এর অসুবিধাসমূহ:**

- Cost of hardware and software
- Size
- Complexity
- Higher impact of failure.

### Data Abstraction

**Data Abstraction:** Database এর হতে দক্ষতার সাথে Data উদ্ধার করার জন্য বিভিন্ন ধরনের জটিলতাকে গোপন রেখে DBMS কে ব্যবহারকারীদের নিকট সহজবোধ্য ও ব্যবহারযোগ্য করার জন্য প্রয়োজনীয় বৈশিষ্ট্য সমূহকে বাচায় করার প্রক্রিয়াকে Data Abstraction বলে।

**Data Abstraction Level / three schema architecture:**



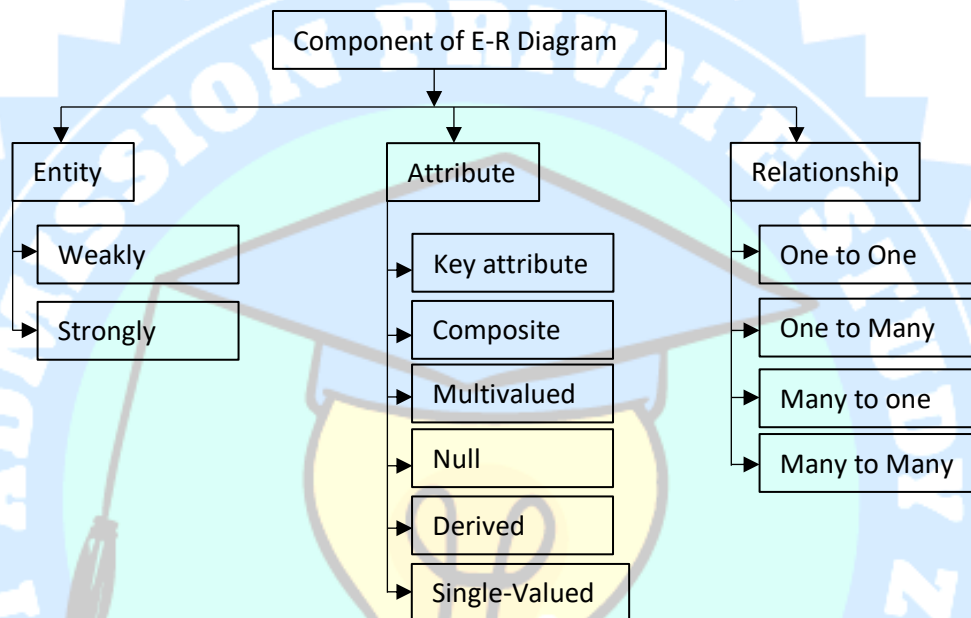
**Physical Level:** Database এ Data সমূহ কোথায় কীভাবে সংরক্ষিত থাকে তা এই level এ আলোচনা করা হয়।

**Logical Level:** ডাটাবেজে কি ডাটা সংরক্ষণ করা হবে তাদের ডিজাইন তৈরি করা হয় এবং এদের ভিতরে Relation কেমন হবে তার Blue Print তৈরি করা হয়।

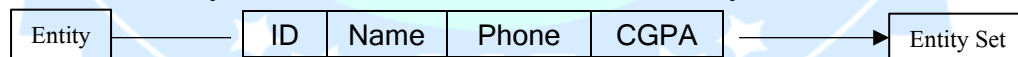
**View Level:** সমগ্র Data base এর অংশ বিশেষ বর্ণনা করে থাকে। সাধারণত User যে সারফেস দেখে থাকে তা এ অংশে আলোচনা করা হয়।

**Database Schema:** ডাটাবেজের সামগ্রিক ডিজাইনকে ডাটাবেস স্কিমা বলে। একইভাবে কোনো রিলেশনাল ডাটাবেজ এ Table Field Table-Field এর মধ্যকার Relation ইত্যাদি Relational Design কে ঐ Relation এর স্কিমা বলে। ডাটাবেজে সিস্টেমে বিভিন্ন ধরনের স্কিমা ব্যবহার করা হয়ঃ Physical Schema, Logical Schema এবং View Schema ।

### E-R Diagram



**Entity:** Entity হলো এমন কতগুলো object যাদের অস্তিত্ব আছে এবং যাদের একটি হতে অপরটি আলাদা করা যায় অর্থাৎ object এর একক বৈশিষ্ট্যকে Entity বলে। Entity set হলো একই রকম কতগুলো অবজেক্ট এর একটি গ্রুপ যারা একই ধরনের প্রোপার্টি বা অ্যাট্রিবিউট শেয়ার করে থাকে। এখানে ID, Name, Phone ও CGPA সবগুলোকে একত্রে Entity set বলে। এদেরকে আলাদা ভাবে Entity বলে।



Entity এর উদাহরণঃ Physical Existence- House, Person এবং Logical Existence- Course, Job ।

**Strong Entity Set:** যে entity set এর primary key থাকে তাকে Strong Entity Set বলে।

- ❖ আয়তক্ষেত্র দ্বারা প্রকাশ করা হয়।
- ❖ অন্য কোন Entity Set এর উপর নির্ভরশীল নয়।
- ❖ ইহার relationship set হিসেবে Single Diamond symbol ব্যবহৃত হয়।
- ❖ Discriminator বা partial key ব্যবহৃত হয় না।

**Weak Entity Set:** যে entity set এর primary key থাকে না তাকে Weak Entity Set বলে।

- ❖ ডাবল-আয়তক্ষেত্র দ্বারা প্রকাশ করা হয়।
- ❖ Strong Entity Set এর উপর নির্ভরশীল।
- ❖ ইহার রিলেশনশীপ সেট হিসেবে double Diamond symbol ব্যবহৃত হয়।
- ❖ Discriminator বা partial key ব্যবহৃত হয়।

**Object:** যে সব real world thing একটি হতে অপরটিকে আলাদা করা যায় না তাদের অবজেক্ট বলে।

**Attribute:** কোনো অবজেক্ট এর একক বৈশিষ্ট্যকে ঐ অবজেক্টের এর Attribute বলে। E-R Diagram এ ব্যবহৃত ঐ অবজেক্ট এর Attribute বলে। নিম্নে বিভিন্ন প্রকারের Attribute সম্পর্কে বর্ণনা করা হল:

**Single-valued Attribute:** যার মাত্র একটি মান বা Value আছে। যেমন: Age

**Multivalued Attribute:** যার একাধিক মান বা Value আছে। যেমন: Degree-BSc, MSc,SSC,HSC ইত্যাদি।

**Composite Attribute:** যদি কোনো attribute কে দুইভাবে বিভক্ত করা যায়। যেমন: Name- First name, Middle name, Last name.

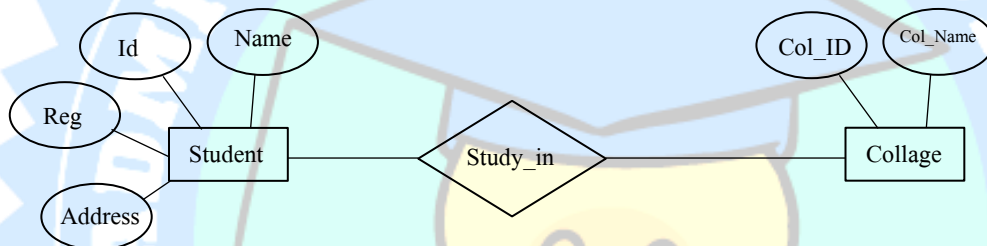
**Derived Attribute:** অন্য কোনো attribute এর ওপর নির্ভরশীল বা অন্য কোনো attribute থেকে আসে। যেমন: Date of birth থেকে Age আসে অর্থাৎ Age হলো derived attribute.

**Null Attribute:** বলতে বোঝায় অজানা কোনো মান অর্থাৎ ইনপুট করা হয়নি ফাঁকা এমন attribute.

**Key Attribute:** যার মাধ্যমে দুই বা ততোদিক entity set থেকে ইউনিকভাবে কোনো entity কে চিহ্নিত করা হয়।

**Relationship:** দুই বা ততোদিক এনটিটি এর মধ্যে সম্পর্ক স্থাপন করাকে relationship বলে। Example: “Teacher teaches students” Teacher ও Students হল এনটিটি এবং teach হল এনটিটি দুটির মধ্যে relationship।

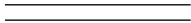
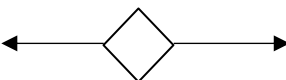
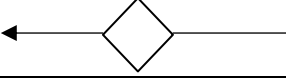
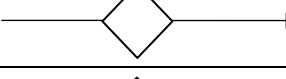

**E-R Diagram:** এনটিটি রিলেশনশীপ মডেল এ Attribute এর সাথে Entity এবং Entity এর সাথে Entity এর সম্পর্ক যে ডায়াগ্রাম এর মাধ্যমে দেখানো হয়, তাকে E-R Diagram বলে।



#### E-R Diagram এর Symbol এবং কাজ:

<b>আয়তক্ষেত্র(Rectangles):</b> Entity set কে উপস্থাপন করার জন্য ব্যবহার করা হয়।	
<b>উপবৃত্ত (Ellipses):</b> Entity set এর Attribute কে উপস্থাপন করার জন্য ব্যবহার করা হয়।	
<b>লাইন(Line):</b> Attribute সমূহকে Entity set এর সাথে এবং Entity Set সমূহকে Relationship Set এর সাথে সংযুক্ত করার জন্য লাইন ব্যবহার করা হয়।	
<b>ডায়মন্ড(Diamond):</b> Relationship কে উপস্থাপন করার জন্য ব্যবহার করা হয়।	
<b>ডাবল ডায়মন্ড(Double Diamond):</b> Weak entity set এর relationship কে উপস্থাপন করার জন্য।	
<b>ডাবল আয়তক্ষেত্র:</b> Weak entity set কে উপস্থাপন করার জন্য।	
<b>ডাবল উপবৃত্ত:</b> Multi-valued Attribute কে উপস্থাপন করার জন্য।	
<b>ড্যাস উপবৃত্ত:</b> Derived Attribute কে প্রকাশ করার জন্য ইহা ব্যবহার করা হয়।	



<b>Double Line:</b> রিলেশনশীপ সেট এর মধ্যে total participation কে নির্দেশ করার জন্য ইহা ব্যবহার করা হয়।	
<b>One to One Relationship (1:1):</b>	
<b>One to Many Relationship (1:M):</b>	
<b>Many to One Relationship (M:1):</b>	
<b>Many to Many Relationship (M:M):</b>	

### Database Keys

**Keys:** যার মাধ্যমে দুই বা ততোদিক entity set থেকে ইউনিকভাবে কোনো entity কে চিহ্নিত করা হয়। নিচে কিছু কী এন্ট্রিবিউট আলোচনা করা হল।

Employee

ID	Name	SSN	Salary	Phone	Email
----	------	-----	--------	-------	-------

**প্রাইমারি কী:** যার মাধ্যমে কোনো এনটিটি সেট থেকে কোনো এনটিটিকে ইউনিকভাবে আইডেন্টিফাই করা যায়। Attribute কে অবশ্যই ইউনিক হতে হবে এবং Null ভ্যালু হওয়া যাবে না। যেমন ID, SSN.

**সুপার কী:** এক বা একাধিক এন্ট্রিবিউট এর একটি সেট যার মাধ্যমে কোনো এনটিটিকে ইউনিকভাবে আইডেন্টিফাই করা যায় তাকে সুপার কী বলে। Null ভ্যালু থাকতে পারে।

{ID}, {ID, Name}, {SSN}, {ID, SSN}, {SSN, Phone}, {Name, Phone}, {ID, SSN, Phone}

**ক্যান্ডিডেট কী:** এক বা একাধিক এন্ট্রিবিউট এর একটি ন্যূনতম সেট যার মাধ্যমে কোনো এনটিটিকে অদ্বিতীয়ভাবে আইডেন্টিফাই করা যায় তাকে ক্যান্ডিডেট কী বলে। {ID}, {SSN}, {Name, Phone}

**ফরেন কী:** যদি কোনো প্রাইমারি কী অন্য কোনো এনটিটি সেট এ ব্যবহার করা হয়।

**কম্পোজিট কী:** দুটি এন্ট্রিবিউট এর সমন্বয়ে গঠিত Primary key কে কম্পোজিট কী বলে।

### Data Model

**Data Model:** ডাটার গঠন, ডাটার রিলেশনশীপ, ডাটার শর্তারোপ বা বাধ্যবাধকতা ইত্যাদি বর্ণনা করার জন্য, কতগুলো ধারণাগত টুলের এর সমষ্টিকে Data Model বলে। Data Model এর বিভিন্ন Group-

**Object Based Logical Model:** Logical ও View Level এ ডাটা বর্ণনা করার জন্য যে মডেল ব্যবহার করা হয় থাকে Object based Logical Model বলে।

i) **The entity relationship model:** পূর্বে আলোচনা করা হয়েছে।

ii) **The object oriented relation model:**

class\_account

Account_No Balance
Pay_interest ( ) Pay_update ( )

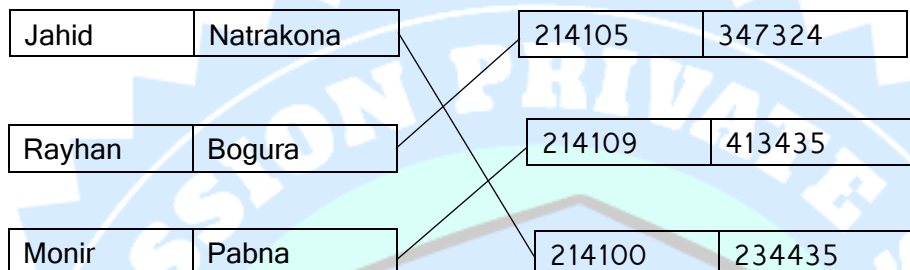
iii) **The semantic data model:** একটি অবজেক্ট যা জমাকৃত symbol, Data কে Real World এর সাথে সম্পর্ক তৈরিতে সাহায্য করে।

**Record Based Logical Model:** Record based logical model সমূহকে logical ও view level এ বর্ণনা করা হয়। Object based এর সাথে মূলপার্থক্য হলো এখানে সমগ্র কাঠামো সুস্পষ্ট।

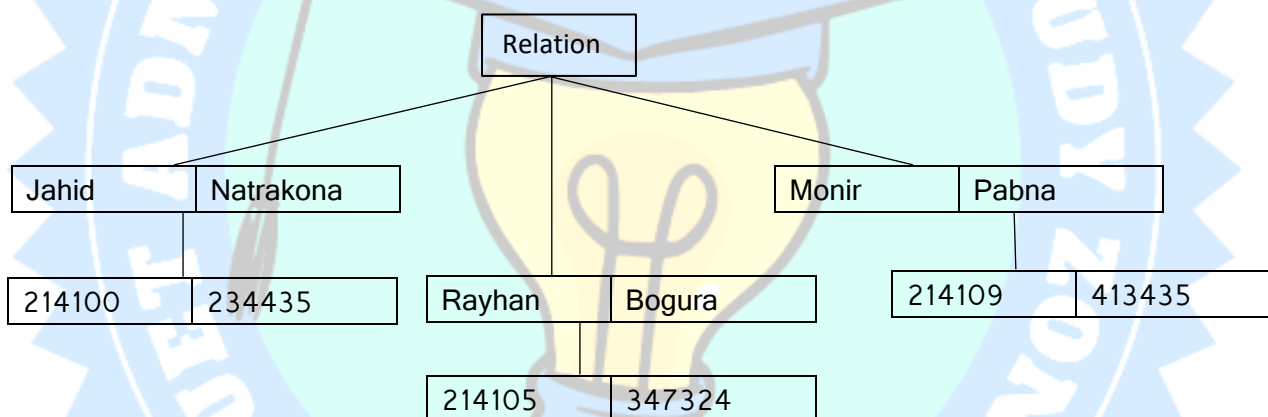
i) **Relational Based Data Model:** যে ডাটা মডেল এ ডাটা এবং তাদের রিলেশনকে টেবিলের মধ্যে উপস্থাপন করা হয় তাকে Relational Based Data Model বলে।

Date	Name	Phone	Address	Product Name	Product Quantity
10-12-2020	Arif	12345	Pabna	Shirt	3
10-12-2020	Jahid	12346	Netrakona	Pent	4
10-12-2020	Nayem	12347	Khulna	Shirt	3
11-12-2020	Sharif	12348	Netrakona	Pent	6
12-12-2020	Riyad	12349	Pabna	Shirt	2
12-12-2020	Fahim	12340	Netrakona	Pent	1

ii) **Network Model:** Collection of record এর মধ্যে relationship কে link দ্বারা প্রকাশ করা হয়।



iii) **Hierarchical Model:** Collection of tree দ্বারা প্রকাশ করা হয়।



## Transection

**Transection** হলো Program execution এর একটি ইউনিট যা ডাটা আইটেম access করে এবং প্রয়োজনে পরিবর্তন (update) করে।

**Transection State:**

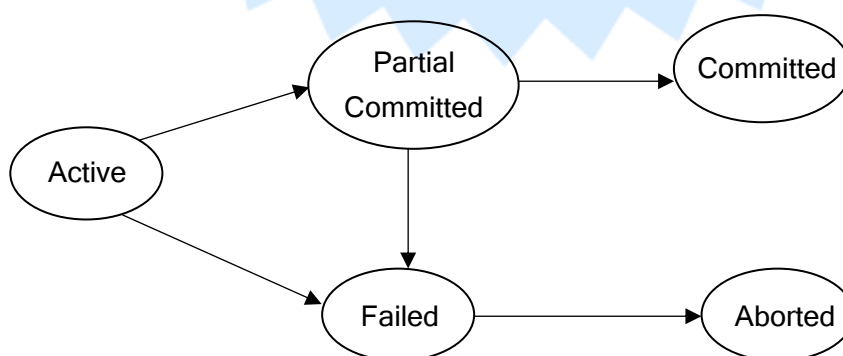


Figure: State Diagram

**ACID properties of transection:****Atomicity:** All or none**Consistency:** সামঞ্জস্যপূর্ণ। The previous sum of the transaction=The next sum of the transaction.**Isolation:** একসাথে একাধিক transection এর সময় Inconsistency বজায় রাখা।**Durability:** স্থায়িত্ব। ট্রানসেকশন সম্পূর্ণ হওয়ার পর ডাটাবেজে তাৎক্ষণিক ভাবে update থাকতে হবে।**Normalization**

Database এর রিলেশনের ডেটার পুনরাবৃত্তি (Redundancy) কম করানোর জন্য টেবিল/রিলেশন কে একাধিক ছোট ছোট টেবিলে বা রিলেশনে ভাগ করার পদ্ধতিকে Normalization বলে। এর বিভিন্ন ধাপগুলি হল:

Students				
ID	Name	Dept	Dept ID	Course
214105	Jahir	CSE	40	CSE22, CSE23, EEE21
214106	Bristy	CSE	40	CSE22, CSE23, EEE21
214107	Sharmin	CE	41	CE21, CE22
214108	Sakib	EEE	42	CSE21, EEE21
214109	Masura	CE	41	CE21

1. 1<sup>st</sup> Normal Form:

i. Atomic ii. Unique

Students				
ID	Name	Dept	Dept ID	Course
214105	Jahir	CSE	40	CSE22
214105	Jahir	CSE	40	CSE23
214105	Jahir	CSE	40	EEE21
214106	Bristy	CSE	40	CSE22
214106	Bristy	CSE	40	CSE23
214106	Bristy	CSE	40	EEE21
214107	Sharmin	CE	41	CE21
214107	Sharmin	CE	41	CE22
214108	Sakib	EEE	42	CSE21
214108	Sakib	EEE	42	EEE21
214109	Masura	CE	41	CE21

2. 2<sup>nd</sup> Normal Form:i. 1<sup>st</sup> NF ii. Primary Key iii. Partial Dependency থাকা যাবে না।

Students			
ID	Name	Dept	Dept ID
214105	Jahir	CSE	40
214106	Bristy	CSE	40
214107	Sharmin	CE	41
214108	Sakib	EEE	42
214109	Masura	CE	41

ID	Course
214105	CSE22
214105	CSE23
214105	EEE21
214106	CSE22
214106	CSE23
214106	EEE21
214107	CE21
214107	CE22
214108	CSE21
214108	EEE21
214109	CE21

3. 3<sup>rd</sup> Normal Form:i. 2<sup>nd</sup> NF

## ii. Transitive functional Dependencies থাকা যাবে না।

ID	Name	Dept ID
214105	Jahir	40
214106	Bristy	40
214107	Sharmin	41
214108	Sakib	42
214109	Masura	41

Dept	Dept ID
CSE	40
CE	41
EEE	42

ID	Course
214105	CSE22
214105	CSE23
214105	EEE21
214106	CSE22
214106	CSE23
214106	EEE21
214107	CE21
214107	CE22
214108	CSE21
214108	EEE21
214109	CE21

## 4. BCNF (Boyce and Codd Normal Form):

i. 3<sup>rd</sup> NFii. For each functional dependency ( $X \rightarrow Y$ ), X is super key5. 4<sup>th</sup> Normal Form:

## i. BCNF

## ii. Multi-valued dependency থাকা যাবে না।

6. 5<sup>th</sup> Normal Form:i. 4<sup>th</sup> NF

## ii. Data loss হওয়া যাবে না।

7. 6<sup>th</sup> Normal Form:i. 5<sup>th</sup> NF

**Partial Dependency:** একটি non-prime attribute কোনো Candidate key এর উপর functionally dependent হয় তখন তাকে Partial Dependency বলে।

**Non-Prime:** যে সকল attribute candidate key ও primary key নয়।

**Transitively functional dependency:** টেবিলের/রিলেশনের Primary key উপর dependent না হয়ে অন্য কোনো কলামের উপর উপর নির্ভর করলে।



## Relational Algebra

একসেট অপারেশন নিয়ে গঠিত যা এক বা একাধিক রিলেশন ইনপুট নিয়ে কাজ করে এবং আউটপুটে নতুন রিলেশন কে দেখায়।

রিলেশনাল অপারেটর এবং তার অপারেশন:

### EMPLOYEE

EMP_ID	EMP_NAME	EMP_MGR	TITLE	EMP_DEPT
1045	Jerry	4536	Director	10
1245	Tom	1045	Accountant	10
1540	Minton	2050	Director	15

**Selection ( $\sigma$ ):** রো এর সাবসেট দেখতে সিলেকশন ব্যবহার করা হয়।  $\sigma_{condition}(Table\_Name)$

$\sigma_{EMP\_DEPT=10}(Employee)$

### EMPLOYEE

EMP_ID	EMP_NAME	EMP_MGR	TITLE	EMP_DEPT
1045	Jerry	4536	Director	10
1245	Tom	1045	Accountant	10

**Projection ( $\Pi$ ):** কলামের সাবসেট দেখতে Projection ব্যবহার করা হয়।  $\Pi_{Column\_name}(Table\_Name)$

$\Pi_{EMP\_ID, EMP\_NAME}(EMPLOYEE)$

### EMPLOYEE

EMP_ID	EMP_NAME
1045	Jerry
1245	Tom
1540	Minton

**Selection Projection:**  $\Pi_{EMP\_ID, EMP\_NAME}(\sigma_{EMP\_DEPT=10}(Employee))$

### EMPLOYEE

EMP_ID	EMP_NAME
1045	Jerry
1245	Tom

**A**

Name	PRIZE
M	10
N	12
M	15

**B**

Name	Prize
M	10
N	18

**Union ( $\cup$ ):**

**(A $\cup$ B)**

Name	PRIZE
M	10
N	12
M	15
N	18

**Intersection ( $\cap$ ): (A $\cap$ B)**

Name	PRIZE
M	10

**Set Difference (-): (A-B)**

Name	PRIZE
N	12
M	15

**Joining two relations****Cartesian Product (X):** r

A	B
$\alpha$	1
$\beta$	2

s

C	D	E
$\alpha$	10	a
$\beta$	10	a
$\beta$	20	b
$\gamma$	10	b

r x s

A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\alpha$	1	$\beta$	10	a
$\alpha$	1	$\beta$	20	b
$\alpha$	1	$\gamma$	10	b
$\beta$	2	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b
$\beta$	2	$\gamma$	10	b

**Natural Join:** r

Id	Name
1	Rahim
2	Karim
3	Faruk

s

Id	Age	Dept
1	21	EEE
2	22	ME
3	20	CSE

r x s

r.ID	Name	s.ID	Age	Dept
1	Rahim	1	21	EEE
1	Rahim	2	22	ME
1	Rahim	3	20	CSE
2	Karim	1	21	EEE
2	Karim	2	22	ME
2	Karim	3	20	CSE
3	Faruk	1	21	EEE
3	Faruk	2	22	ME
3	Faruk	3	20	CSE

**After natural join:**  $r \bowtie s = \Pi_{r.ID, Name, Age, Dept}(\sigma_{r.ID=s.ID}(r \times s))$ 

r.ID	Name	Age	Dept
1	Rahim	21	EEE
2	Karim	22	ME
3	Faruk	20	CSE

## Query Language – SQL

**Query Language:** Query শব্দের অর্থ তল্লাসি বা search করা। যে Language এর সাহায্যে database থেকে নির্দিষ্ট Data Insert, delete, modify ইত্যাদি করা যায় থাকে Query Language বলে।

**Database Language:** কম্পিউটার ভাষার সাহায্যে ডেটাবেস টেবিলের গঠন (Table Creation), টেবিলে তথ্য সংযোজন (Data Insertion), তথ্য বিয়োজন (Data Deletion), নবীকরণ (Update), টেবিল থেকে ডেটা খুঁজে বের করা (Query), টেবিলের ডেটার নিয়ন্ত্রণ (Data Control) ইত্যাদি কাজ সম্পন্ন করা যায় তাকে ডেটাবেস ল্যাঙ্গুয়েজ বলে। ব্যবহারকারী এই ল্যাঙ্গুয়েজের সাহায্যে DBMS সফটওয়্যার পরিচালনা করে।

**Database Language চার প্রকার:**

- i. DDL (Data Definition Language)
- ii. DML (Data Manipulation Language)
- iii. DCL (Data Control Language)
- iv. TCL (Transaction Control Language)

### DDL (Data Definition Language)

যে ডেটাবেস ল্যাঙ্গুয়েজের সাহায্যে ডেটাবেসের বিভিন্ন স্ট্রাকচার/স্কিমা নির্ধারণ, ডেটাবেসের অবজেক্ট তৈরি, পাল্টান, বা মুছে ফেলা ইত্যাদি কাজ করা যায় তাকে ডেটা ডেফিনেশন ল্যাঙ্গুয়েজ বলে।

ব্যবহৃত কমান্ড সমূহ:

1. **CREATE:** Database এ নতুন Database Table তৈরি করার জন্য এ কমান্ড ব্যবহার করা হয়।  
Syntax: CREATE DATABASE database\_name;
2. **USE:** ডাটাবেস সিলেক্ট করা হয়। Syntax: USE databse\_name;
3. **DROP:** Database এবং Table delete করার জন্য এ কমান্ড ব্যবহৃত হয়।  
Syntax: DROP DATABASE database\_name;      Syntax: DROP TABLE table\_name;
4. **TRUNCATE:** Table এর ডাটা দ্রুত delete করার জন্য এ কমান্ড ব্যবহৃত হয়। অর্থাৎ Header ছাড়া সকল ডাটা মুছা যাবে এই কমান্ড ব্যবহার করে।  
Syntax: TRUNCATE TABLE table\_name;
5. **SHOW:** সকল Database এবং সকল Table দেখার জন্য এ কমান্ড ব্যবহার করা হয়।  
Syntax: SHOW DATABASES;  
Syntax: SHOW TABLES;      Syntax: SHOW TABLES FROM database\_name;
6. **RENAME TABLE:** টেবিলের নাম পরিবর্তন করার জন্য এ কমান্ড ব্যবহৃত হয়।  
Syntax: ALTER TABLE o\_name RENAME n\_name;      RENAME TABLE o\_name TO n\_name

### Table Constraints:

- ❖ **NOT NULL:** কোনো NULL ভ্যালু ইনসার্ট হবে না।
  - ❖ **UNIQUE:** একটি অপরটি থেকে ভিন্ন হতে হবে।
  - ❖ **PRIMARY KEY:** UNIQUE এবং NOT NULL হবে।
  - ❖ **FOREIGN KEY:** অন্য কোনো টেবিলের কোনো কলামের উপর নির্ভর করে ডাটা ইনপুট হবে।
7. **CREATE TABLE:** এই কমান্ডের সাহায্যে ডাটাবেস এ টেবিল তৈরি করা যায়।  
Syntax: CREATE TABLE table\_name(  
                    column\_name\_1 datatype(size),  
                    column\_name\_2 datatype(size)  
                    );

**Example:**

```
CREATE TABLE Students(
    ID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Birth DATE,
    Home VARCHAR(100),
    Department VARCHAR(100)
);
```

Output: অভ্যন্তরীণ ভাবে নিম্নোক্ত টেবিল তৈরি হবে।

ID	Name	Birth	Home	Department
empty				

8. **ALTER:** এই কমান্ডের সাহায্যে ডেটাবেস টেবিলের গঠন পরিবর্তন করা যায়। অর্থাৎ টেবিলে নতুন ফিল্ড যুক্ত করা, কোনো ফিল্ডকে মুছে ফেলা, কোনো ফিল্ডের নাম এবং ডেটা টাইপ পরিবর্তন ইত্যাদি বিভিন্ন কাজ করা যায়।

❖ **ADD column:**

Syntax: ALTER TABLE table\_name ADD COLUMN column\_name datatype(size);

**Example:** Student টেবিলে Mobile নামে একটি নতুন ফিল্ড যুক্ত করার কমান্ড হল:

```
ALTER TABLE STUDENT
ADD COLUMN Mobile int;
```

Output: অভ্যন্তরীণ ভাবে নিম্নোক্ত টেবিল তৈরি হবে।

ID	Name	Birth	Home	Department	Mobile
empty					

❖ **RENAME column:**

Syntax: ALTER TABLE table\_name RENAME previous\_name to new\_name;

**Example:** Students টেবিলে “Department” এর নাম পরিবর্তন করে “Dept” করার কমান্ড হল:

```
ALTER TABLE Students
RENAME Department to Dept;
```

Output: অভ্যন্তরীণ ভাবে নিম্নোক্ত টেবিল তৈরি হবে।

ID	Name	Birth	Home	Dept	Mobile
empty					

❖ **DROP column:**

Syntax: ALTER TABLE table\_name DROP column\_name;

**Example:** Student টেবিলের Mobile কলাম মুছে ফেলার জন্য কমান্ড হল:

```
ALTER TABLE Students DROP mobile;
```

❖ **ADD CONSTRAINTS:**

Syntax: ALTER TABLE table\_name ADD CONSTRAINT constraint\_name(column\_name);

```
ALTER TABLE students ADD CONSTRAINT PRIMARY KEY(ID);
```

❖ **Delete CONSTRAINTS:**

Syntax: ALTER TABLE table\_name DROP constraint\_name;

```
ALTER TABLE students DROP PRIMARY KEY;
```

**Data Type:**

- |                                     |                                    |
|-------------------------------------|------------------------------------|
| a. INT[integer]                     | d. VARCHAR(size) [variable length] |
| b. DOUBLE(size, size)[float number] | e. DATE [YYYY-MM-DD]               |
| c. CHAR(size) [Fixed length]        |                                    |

## DML (Data Manipulation Language)

এই ডেটাবেস ল্যাঙ্গুয়েজ এর মাধ্যমে ডেটাবেস টেবিলে ডেটা প্রবেশ করা (Insert), ডেটা আপডেট করা, ডেটা মুছে ফেলা (Delete), পালটানো (Modify) ইত্যাদি অর্থাৎ কুয়েরি অপারেশন সম্পন্ন করা যায়। বিভিন্ন ধরনের DML কমান্ডগুলি হল – SELECT, INSERT, UPDATE, DELETE, CALL ইত্যাদি।

1. **INSERT:** কোনো টেবিল ডেটা যুক্ত করার জন্য এই কমান্ডের ব্যবহার করা হয়।

**Syntax 1:** INSERT INTO TABLE table\_name (column\_name\_1, column\_name\_2)  
VALUES (value1, value2);

**Syntax 2:** INSERT INTO table\_name VALUES (value\_1, value\_2, value\_3);  
[এক্ষেত্রে কলাম এবং ভ্যালু সংখ্যা সমান হতে হবে]

- String type এর কিছু ইনসার্টের জন্য single quotes ( ' ' ) ব্যবহার করতে হবে।
- Date insert এর default format হল YYYY-MM-DD

**Example:** Students টেবিলে নতুন রেকর্ড যুক্ত করার কমান্ড হল:

INSERT INTO Students VALUES  
(1, 'J', '01-10-12', 'Netrakona',  
'CSE', '0123');

Output: অভ্যন্তরীণ ভাবে নিম্নোক্ত টেবিল তৈরি হবে।

Students

ID	Name	Birth	Home	Dept	Mobile
1	J	01-10-12	Netrakona	CSE	0123

INSERT INTO Students VALUES  
(2, 'R', '00-11-12', 'Comilla', 'CSE',  
'0124'), (3, 'S', '99-10-16',  
'Chittagong', 'CSE', '0125');

Output: অভ্যন্তরীণ ভাবে নিম্নোক্ত টেবিল তৈরি হবে।

Students

ID	Name	Birth	Home	Dept	Mobile
1	J	01-10-12	Netrakona	CSE	0123
2	R	00-11-12	Comilla	CSE	0124
3	S	99-10-16	Chittagong	CSE	0125

2. **UPDATE:** ডাটাবেসের কোনো একটি নির্দিষ্ট রো এর ডাটা পরিবর্তন করার জন্য UPDATE কমান্ড ব্যবহৃত হয়।

**Example:** Students টেবিলে যার ID 1 তার Mobile পরিবর্তন করার কুয়েরি হল:

UPDATE Students SET mobile =11223  
WHERE ID=1;

Output: অভ্যন্তরীণ ভাবে নিম্নোক্ত টেবিল তৈরি হবে।

Students

ID	Name	Birth	Home	Dept	Mobile
1	J	01-10-12	Netrakona	CSE	11223
2	R	00-11-12	Comilla	CSE	0124
3	S	99-10-16	Chittagong	CSE	0125

3. **DELETE:** ডাটাবেসের একটি টেবিল থেকে কোনো নির্দিষ্ট রো কে মুছে ফেলার জন্য DELETE কমান্ড ব্যবহৃত হয়।

**Example:** Students টেবিলে যার আইডি 3 তার Data মুছে ফেলার কুয়েরি হল:

DELETE FROM Students WHERE  
ID=3;

Output: অভ্যন্তরীণ ভাবে নিম্নোক্ত টেবিল তৈরি হবে।

Students

ID	Name	Birth	Home	Dept	Mobile
1	J	01-10-12	Netrakona	CSE	0123
2	R	00-11-12	Comilla	CSE	0124

4. **Clause:**

- a. **FROM:** Table নির্বাচন করার জন্য ব্যবহার করা হয়।
- b. **WHERE:** Condition Apply করার জন্য ব্যবহার করা হয়।
- c. **SELECT:** কোনো টেবিল থেকে ডেটা খোজার জন্য এ কমান্ড ব্যবহার করা হয়।

**Syntax 1:** SELECT \* FROM table\_name;

[টেবিলের সকল কলাম সিলেক্ট হবে এবং এর মধ্যে কুয়েরি সম্পন্ন হবে]



**Example:** সম্পূর্ণ Students টেবিলকে দেখানোর কুয়েরি হল:

SELECT \* FROM Students;

Students

ID	Name	Birth	Home	Dept	Mobile
1	J	01-10-12	Netrakona	CSE	0123
2	R	00-11-12	Cumilla	CSE	0124

**Syntax 2:** SELECT column\_1, column\_2 FROM table\_name;

[টেবিলের column\_1, column\_2 কলাম সিলেক্ট হবে এবং এর মধ্যে কুয়েরি সম্পন্ন হবে]

**Example:** Students টেবিল থেকে শুধু ID এবং Name দেখানোর কুয়েরি হল:

SELECT ID, Name FROM Students;

Students

ID	Name
1	J
2	R

## 5. Operators:

### Arithmetic Operator:

Operator	Meaning
+	যোগ
-	বিয়োগ
*	গুন
/	ভাগ

Precedency (অগ্রগন্যতা/প্রাধান্য):

- ১) \* ও / এর precedency + ও - এর থেকে বেশি
- ২) একই precedency সম্পন্ন দুইটি অপারেটর একসাথে থাকলে বাম থেকে ডানে precedency পরিমাপ করা হয়।
- ৩) পেরেন্টিসিস ( ) দ্বারা higher precedency বুঝানো হয়।

### Table for basic example (1-14):

Employee

ID	Name	Department	Address	Salary
2201	J	CSE	Netrakona	2000
2202	N	EEE	Netrakona	3000
2203	F	CSE	Barisal	5000
2204	A	CSE	Bogura	10000
2205	R	EEE	Bogura	2000
2206	T	ME	Noakhali	3000

**Example 1:** Employee টেবিল থেকে সকলের salary 10% বৃদ্ধি করে একবছরে সেলারি এবং সকল তথ্য দেখার কমান্ড লিখ।

**সমাধান:** SELECT ID, Name, Department, Address, Salary, Salary\*1.1\*12 FROM Employee.

ID	Name	Department	Address	Salary	Salary*12*1.1
2201	J	CSE	Netrakona	2000	26400
2202	N	EEE	Netrakona	3000	39600
2203	F	CSE	Barisal	5000	66000
2204	A	CSE	Bogura	10000	132000
2205	R	EEE	Bogura	2000	26400
2206	T	ME	Noakhali	3000	66000

**Comparator Operator:** বিভিন্ন শর্ত প্রয়োগের জন্য Comparator Operator সমূহ WHERE clause এর সাথে ব্যবহার করা হয়।

Operator	Meaning
=	Equal
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
<>	Not equal

## Other Comparator Operator

Operator	
Like	NOT Like
Between .... AND..	NOT Between .... AND..
IN	NOT IN

## Logical Operator:

Operator	Meaning
AND	Returns TRUE if both component conditions are TRUE
OR	Returns TRUE if either component condition is TRUE
NOT	Returns TRUE if the following condition is FALSE

**Aggregate function:** Remember aggregate function always return only one value.

- SUM - SELECT SUM(salary) FROM Employee WHERE address = "Netrakona";
- AVG - SELECT AVG(salary) FROM Employee WHERE address = "Netrakona";
- MAX - SELECT MAX(salary) FROM Employee WHERE address = "Netrakona";
- MIN - SELECT MIN(salary) FROM Employee WHERE address = "Netrakona";
- COUNT - SELECT COUNT(salary) FROM Employee WHERE salary>1000 AND salary<5000 AND address = "Netrakona";

Example 2: Employee টেবিল থেকে যাদের Address Netrakona তাদের Name এবং Address দেখার কমান্ড লিখ।

```
SELECT Name, Address
FROM Employee
WHERE Address="Netrakona";
```

Output: Employee

Name	Address
J	Netrakona
N	Netrakona

Example 3: Employee টেবিল থেকে যাদের সেলারি 4000 এর কম তাদের নাম এবং সেলারি দেখার কমান্ড

```
SELECT Name, Salary FROM
Employee WHERE Salary<4000;
```

Output: Employee

Name	Salary
J	2000
N	3000
R	3000

Example 4: Employee টেবিল থেকে স্বতন্ত্র বা ইউনিক Address দেখার কমান্ড

```
SELECT DISTINCT Address FROM
Employee;
```

Output: Employee

Address
Netrakona
Barisal
Bogura
Noakhali

Example 5: Employee টেবিল থেকে কতটি স্বতন্ত্র বা ইউনিক Address আছে তা দেখার কমান্ড

```
SELECT COUNT(DISTINCT
Address) FROM Employee;
```

Output: Employee

COUNT(DISTINCT Address)
4

Example 6: Employee টেবিল যাদের সেলারি 2000 থেকে বেশি 5000 থেকে কম তাদের দেখার কমান্ড  
 SELECT \* FROM Employee WHERE salary>2000 AND salary<5000;

Example 7: Employee টেবিলে যাদের সেলারি 2000 থেকে 5000এর মধ্যে তাদের Name দেখার কমান্ড  
**Type 1:** SELECT \* FROM Employee WHERE salary Between 2000 AND 5000;  
**Type 2:** SELECT \* FROM Employee WHERE salary>=2000 AND salary<=5000;

Example 8: Employee টেবিলে যাদের সেলারি 5000 এবং address Barisal তাকে Name দেখার কমান্ড  
 SELECT \* FROM Employee WHERE salary=5000 AND Address="Barisal";

Example 9: যাদের নামের প্রথম অক্ষর J দিয়ে শুরু হয়েছে তাদের দেখার কমান্ড  
 SELECT \* FROM Employee WHERE name LIKE 'J%';

Example 10: যাদের নামের শেষের অক্ষর y আছে তাদের দেখার কমান্ড  
 SELECT \* FROM Employee WHERE name LIKE '%y';

Example 11: যাদের নামের মাঝে অক্ষর m আছে তাদের দেখার কমান্ড  
 SELECT \* FROM Employee WHERE name LIKE '%m%';

Example 12: যাদের 4th অক্ষর m আছে তাদের দেখার কমান্ড  
 SELECT \* FROM Employee WHERE name LIKE '\_\_\_m%';

Example 13: যাদের 3th অক্ষর m আছে তাদের দেখার কমান্ড  
 SELECT \* FROM Employee WHERE name LIKE '\_\_m\_%';

Example 13: যাদের 3th অক্ষর m আছে তাদের দেখার কমান্ড  
 SELECT \* FROM Employee WHERE name LIKE '\_\_m\_';

Example 14: যাদের নামের শেষের অক্ষর u নেই এবং যার Address Netrakona তাদের দেখার কমান্ড  
 SELECT \* FROM Employee WHERE name NOT LIKE '%u' AND Address='Netrakona';

## Natural Join

দুইটি টেবিলের মধ্যে সাধারণ কোনো কলামের উপর ভিত্তি করে একটি টেবিল হিসেবে দেখানোর প্রক্রিয়া।

Example 01: নিচের টেবিল দুইটি থেকে যাদের Salary 2000 এর উপরে এবং post manager তাদের name, Address, Salary, Branch দেখার জন্য কমান্ড লেখ।

Emp(ID, Name, salary, Address)

Job(ID, Post, Branch)

**Solution 1:** SELECT Emp.Name, Emp.Address, Emp.Salary, Job.Branch FROM Emp,Job  
 WHERE Emp.Salary >2000 AND job.post = "MANAGER";

**Solution 2:** SELECT Emp.Name, Emp.Address, Emp.Salary, Job.Branch FROM Emp, Job  
 WHERE Emp.ID = Job.ID AND Emp.Salary >2000 AND Job.post = "MANAGER";

## ORDER BY, GROUP BY & HAVING

**ORDER BY:** Data কে সাজানোর(sort) জন্য ORDER BY statement ব্যবহার করা হয়। Order by ব্যবহার করার পর ডাটাসমূহ by default ascending order এ থাকে।

Example 1: Emp(ID, Name, salary, Address) যাদের salary সবথেকে বেশি এ রকম দুইজনের সকল তথ্য দেখার কুয়েরি লিখ।

```
SELECT * FROM emp ORDER BY salary DESC LIMIT 2;
```

**GROUP BY:** Data কে সাজানোর(sort) জন্য Aggregate function এর সাথে GROUP BY statement ব্যবহার করা হয়।

**HAVING:** Aggregate function সমূহের সাথে condition ব্যবহারের জন্য HAVING clause ব্যবহার করা হয়।

**Note:** Having সবসময় GROUP BY এর পর বসে।

Syntax:

```
SELECT column_name FROM table_name
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY column_name;
```

Example 2: Students(ID, Name, salary, dept\_id, Address) কোন DEPT এ কতজন ছাত্র আছে দেখাও।

```
SELECT Dept_id, COUNT(*) as COUNT FROM students
GROUP BY Dept_id;
```

Example 3: Students(ID, Name, salary, dept\_id, Address) কোন DEPT এ অন্তত দুইজন ছাত্র আছে তাদের তালিকার ছোট থেকে বড় আকারে দেখাও।

```
SELECT dept_id, COUNT(*) FROM students
GROUP BY dept_id
HAVING COUNT(*) >= 2
ORDER BY COUNT(*) ASC;
```

Example 4: ধরি একটি Customer table এর নিম্নোক্ত গুলো attributes আছে: CustomerID, CustomerName এবং Country. একটি SQL command লিখ, যেটি প্রতিটি দেশে গ্রাহকের সংখ্যা তালিকায় ছোট থেকে বড় হিসেবে সাজাবে। যেখানে তালিকায় কেবলমাত্র ৫টি গ্রাহকভুক্ত দেশ থাকবে। [DUET 20-21]

```
SELECT Country, COUNT(Country) FROM Customer
GROUP BY Country
HAVING COUNT(Country) >= 5
ORDER BY COUNT(Country) DESC;
```

## Sub-Query

যদি কুয়েরির ভিতর কুয়েরি লেখা হয়, এটা হচ্ছে সাব-কুয়েরি।

Example 1: Emp(ID, Name, salary, Address) টেবিল থেকে দ্বিতীয় সর্বোচ্চ সেলারি দেখাও।

```
SELECT MAX(Salary) FROM Emp
WHERE Salary < (
SELECT MAX(Salary) FROM Emp);
```

Example 2: Emp(ID, Name, salary, Address) টেবিল থেকে দ্বিতীয় সর্বনিম্ন সেলারি দেখাও।

```
SELECT MIN(Salary) FROM Emp
WHERE Salary > (
SELECT MIN(Salary) FROM Emp);
```

Example 3: Emp(ID, Name, salary, Address) টেবিল থেকে যাদের সেলারি Jamal এর থেকে কম তাদের সকল তথ্য দেখাও।

```
SELECT * FROM Emp
WHERE Salary < (
SELECT Salary FROM Emp
WHERE Name = "Jamal");
```

Example 4: Emp(ID, Name, Hiredate, salary, Address) টেবিল থেকে MYMENSINGH এর গড় সেলারি থেকে বেশি তাদের নাম এবং সেলারি বের কর।

```
SELECT Name, Salary FROM Emp
WHERE Salary > (
SELECT AVG(salary) FROM Emp
WHERE Address='MYMENSINGH');
```

Example 5: STA Hall শুধু ২য় বর্ষ এর প্রতি ডিপার্টমেন্ট থেকে একজন করে সর্বোচ্চ CGPA প্রাপ্তদের আবাসন সুবিধা দেওয়া হবে। DUET এর Students(SID, NAME, DEPT, GPA, CGPA, YEAR, ADDRESS) টেবিল থেকে তাদের খুঁজে বের করার কুয়েরি লিখ।

```
SELECT * FROM Students
WHERE CGPA IN (
SELECT MAX(CGPA) FROM Students
WHERE YEAR = 2 GROUP BY DEPT);
```

Example 6: Emp(ID, Name, Hiredate, salary, Address) টেবিল থেকে যারা FAHAD এর থেকে সিনিয়র তাদের গড় সেলারি বের কর।

```
SELECT AVG(Salary) FROM Emp
WHERE Hiredate < (
SELECT Hiredate FROM Emp
WHERE Name='FAHAD');
```

Example 7: Emp(ID, Name, Hiredate, salary, Address) টেবিল থেকে যাদের সেলারি Raza থেকে কম এবং তাদের ডাটা গুলো ডিলেট কর এবং Table Show কর।

```
DELETE FROM Emp
WHERE SALARY < (
SELECT salary FROM Emp
```



```
WHERE name = "Raza");
SELECT * FROM Emp;
```

Example 8: যত মার্ক পেয়ে পাস করা যায় ঠিক তত মার্ক পেয়ে পাস করেছে Rana নামের শিক্ষার্থী। বিভাগীয় প্রধান সিদ্ধান্ত নিয়েছে যারা ফেল করেছে তাদের অভিভাবককে তাদের সম্পর্কে অবগত করবেন, এজন্য শিক্ষার্থীদের অভিভাবকের যোগাযোগ নম্বর প্রয়োজন। Students(Id, Name, Phone, Mark, Guardian\_Number) টেবিল থেকে ফেল করা শিক্ষার্থীদের নাম এবং অভিভাবকের নম্বর দেখার SQL কমান্ড লিখ।

```
SELECT Name, Guardian_Phone FROM Students
WHERE Mark < (
SELECT Mark FROM Students
WHERE Name="Rana");
```

### Mixed Example

Example 1: Students (Name, ID, Age, Address)

- Students table এ তোমার নিজের তথ্য দিয়ে একটি কলাম ক্রিয়েট কর।
- তোমার ঠিকানা পরিবর্তন করে Netrakona দাও।
- তোমার তৈরি করা কলামটি ডিলেট কর। [same as DUET 22-23]

- INSERT INTO students  
VALUES ('Sujon', 205, 20, 'Feni');
- UPDATE students SET address='Netrakona'  
WHERE ID=205;
- DELETE FROM students WHERE ID=205;

Example 2: Employee

ID	Name	Department	Address	Salary
2201	J	CSE	Netrakona	2000
2202	N	EEE	Netrakona	3000
2203	F	CSE	Barisal	5000

- Table টি তৈরি করে করে ডাটা সমূহ এর মধ্যে প্রবেশ করাও।
- ID, Name এর মধ্যে যথাক্রমে Primary key, Not Null যুক্ত কর।
- EEE Department এর সকলকে দেখার কুয়েরি লিখ।

- CREATE TABLE Employee (  
ID INT,  
Name VARCHAR(50),  
Department VARCHAR(50),  
Address VARCHAR(50),  
Salary INT);  
  
INSERT INTO Students VALUES(2201, 'J', 'CSE', 'Netrakona', 2000), (2202  
'N', 'EEE', 'Netrakona', 3000), (2203, 'F', 'CSE', 'Barisal', 5000);
- ALTER TABLE students  
ADD CONSTRAINT PRIMARY KEY(ID);  
ALTER TABLE students  
ADD CONSTRAINT NOT NULL(Name);
- SELECT \* FROM Employee WHERE Department = 'EEE';

## Example 3: PSZ\_S

S_Id	S_Name	S_Age	S_Address
10098	J	21	Netrakona
11176	S	21	Chittagong

- Table টি তৈরি করে এর মধ্যে উপরের তথ্য সমূহ প্রবেশ করার SQL Command লিখ।
- তৈরিকৃত PSZ\_S টেবিলে T\_ID ও T\_Name কলামের মধ্যে যথাক্রমে Primary Key ও NOT NULL Constraints যুক্ত করার SQL Command লিখ।
- মনে কর, টেবিলে আরো অনেক তথ্য আছে, যাদের বাড়ি Bogura এবং যাদের নামের শেষে L নেই তাদের দেখার SQL Command লিখ।

```
i. CREATE TABLE PSZ_S(
    S_ID INT,
    S_Name VARCHAR (100),
    S_Age INT,
    S_Address VARCHAR (100)
);
INSERT INTO Students VALUES
    (10098, 'J', 21, 'Netrakona'),
    (11176, 'S', 21, 'Chittagong');
```

```
ii. ALTER TABLE PSZ_S
ADD CONSTRAINT PRIMARY KEY(S_ID);
ALTER TABLE PSZ_S
ADD CONSTRAINT NOT NULL(S_Name);
iii. SELECT * FROM PSZ_S
WHERE S_Address = 'Bogura'
AND S_Name NOT LIKE '%L';
```

Example 4: Students(SID, Name, Mark, dept, Address) টেবিল থেকে যাদের মার্ক 40,30,10,20 এবং যাদের dept EEE তাদের দেখার কুয়েরি লিখ।

```
Solution 1: SELECT * FROM students
WHERE (mark = 40 OR mark = 30 OR mark = 10 OR mark = 20)
AND dept = 'EEE';
Solution 2: SELECT * FROM students
WHERE mark IN(40, 30, 10, 20)
AND dept = 'EEE';
```

Example 5: Students(SID, Name, dept, Address) এবং Stipend(sti\_ID, SID, bank\_ID) টেবিল থেকে যারা বৃত্তি পায় না এবং যাদের বাড়ি Netrakona তাদের দেখার কুয়েরি লিখ।

```
SELECT * FROM students
WHERE Address = "Netrakona"
AND SID NOT IN (
    SELECT SID FROM stipend);
```

Example 6: Emp(ID, Name, Hiredate, dept\_id, salary, Address) টেবিল থেকে প্রতি ডিপার্টমেন্টে সব থেকে বেশি সেলারি পায় তাদের দেখাও তাদের গড় সেলারি বের কর।

```
SELECT * FROM emp
WHERE salary IN (
SELECT MAX(salary) FROM emp
GROUP BY DEPT_ID);
```

Example 7: Emp(ID, Name, salary, Address) টেবিল থেকে যাদের সেলারি গড় সেলারি থেকে বেশি তাদের ডিলেট করার SQL কমান্ড লিখ।

```
DELETE FROM emp
WHERE salary > (
SELECT AVG(salary) FROM emp);
```

Example 8: ঐ সকল Actor দেব দেখার সকল information দেখাও যারা টাইটানিক মুভিতে অভিনয় করে।

```
Actor(act_id, act_fname, act_gender),
Movie_cast(act_id, movie_id, role),
Movie(mov_id, mov_title, mov_year, mov_time)

SELECT * FROM Actor WHERE act_id IN (
SELECT act_id FROM Movie_cast WHERE movie_id IN (
SELECT mov_id FROM Movie WHERE mov_title = 'titanic')));
```

Example 9: PSZ\_Students(ID, Name, Gender, Mark, Phone, Admission\_date)

i. 2022- 2023 সেশনে ঈদুল আযহা উপলক্ষে PSZ এর অধিকাংশ শিক্ষার্থী পরিবারের সাথে ঈদ উৎযাপন করতে গিয়েছিল। একাংশ শিক্ষার্থী নিজের স্বপ্নকে স্বার্থক করতে ঈদে বাড়িতে না গিয়ে গাজিপুর থেকে গিয়েছিল। উক্ত শিক্ষার্থীরা যেন তাদের পড়ার গতি না হারিয়ে ফেলে সেজন্য PSZ থেকে তাদের জন্য ছোট একটি পরীক্ষার আয়োজন করা হয়েছিল। ছেলে ও মেয়েদেরকে আলাদা ভাবে তাদের অর্জিত মার্কের উপর, প্রথম জনের জন্য আকর্ষণীয় পুরস্কার এর ব্যবস্থা করা হয়। উপরের টেবিল থেকে তাদেরকে দেখার SQL Command লিখ।

ii. উপরের টেবিল থেকে যারা 5 January 2023 এর পূর্বে ভর্তি হয়েছে তাদের name এবং ID দেখার SQL command লিখ।

```
i. SELECT * FROM Students
WHERE (MARK, Gender) IN (
SELECT MAX(Mark), Gender FROM Students
GROUP BY Gender
);
ii. SELECT * FROM Students
WHERE Admission_date < "2023-01-05";
```

Example 10: একটি data base table এ Student(Student\_ID, Name, Date\_of\_Birth, CGPA, Department) আছে। এখন, নিম্নোক্ত উত্তরগুলো জানার জন্য SQL Command লিখ: (i) ঐসব ছাত্রদের খুঁজে বের কর যাদের department 'CSE' (ii) ঐসব ছাত্রদের খুঁজে বের কর যাদের CGPA > 3.0 [DUET]

```
(i) SELECT * FROM Student WHERE Department = "CSE";
(ii) SELECT * FROM Student WHERE CGPA > 3.0;
```

Example 11: E\_Info ( E\_ID, E\_Name), E\_Salary(E\_ID, E\_Salary) টেবিলগুলো থেকে যাদের সর্বোচ্চ সেলারি তার নাম দেখার SQL Command লিখ। [DUET 21-22]

```
SELECT E_Name FROM E_Info, E_Salary WHERE E_Info.E_ID = E_Salary.E_ID
AND E_Salary IN (SELECT MAX(E_Salary) FROM E_Info);
```

Example 12: Employee(E\_ID, E\_Name, Dept\_ID), Department(Dept\_ID, Dept\_Name)

i. Employee Name এর সাথে Department Name সংযুক্ত করে দেখাও।

ii. কোন Department এ কতজন কাজ করে দেখাও।

iii. যে সকল Department এ একের অধিক কাজ করে সে সকল Department দেখাও। [DUET 23-24]

- i. SELECT E\_Name, Dept\_Name FROM Employee, Department  
WHERE Employee.Dept\_ID = Department.Dept\_ID;
- ii. SELECT Dept\_Name, COUNT(Dept\_ID) FROM Department  
GROUP BY Dept\_ID;
- iii. SELECT Dept\_Name, COUNT(Dept\_ID) FROM Department  
GROUP BY Dept\_ID  
HAVING COUNT(Dept\_ID) > 1;

Example 13: Consider the following databases: Emp(empno, ename, job, hiredate, sal, comm, mgr, deptno), Dept(deptno, dname, loc) (i) তাদের খুঁজে বের কর যারা একই তারিখে জন্মেন হয়েছে।

```
SELECT * FROM emp e
WHERE hiredate IN (
    SELECT hiredate FROM emp
    WHERE e.salary <> salary
);
```

(ii) To find those employees whose salary is more than the total remuneration (salary + commission) of the designation SALESMAN.

```
SELECT * FROM emp
WHERE sal > (
    SELECT SUM(sal+comm) FROM emp
    WHERE job = 'Salesman'
);
```

(iii) To find the highest paid employee.

```
SELECT * FROM emp
WHERE sal > (
    SELECT MAX(sal) FROM emp
);
```

Example 14: Consider the following databases: Orders (ord\_no, purch\_amt, ord\_date, customer\_id), Customer(customer\_id, cust\_name, city, grade)

(i) Write a SQL statement to make a list with order no, purchase amount, customer name and their cities for those orders which order amount between 500 and 2000.

```
SELECT ord_no, purch_amt, cust_name, city FROM Orders O, Customers C
WHERE O. customer_id = C. customer_id
AND purch_amt BETWEEN 500 AND 2000;
```

(ii) Write a queries to find all orders with order amounts which are on or above-average amounts for their customers.

```
SELECT * FROM Orders WHERE purch_amt >= (
    SELECT AVG(purch_amt) FROM Orders);
```