Sanjivani College of Engineering, Kopargaon

Department of Electronics & Computer Engineering

(An Autonomous Institute)

Affiliated to Savitribai Phule Pune University

Accredited 'A' Grade by NAAC

SANJIVANI
GROUP OF INSTITUTES

Subject: Database Management Systems & SQL

by

Mr. Nitin Bhopale

Introduction to DBMS

Introduction to Database Management Systems, Purpose of Database Systems, Database-System Applications, View of Data, Database Languages, Database System Structure, Data Models, Database users, Database Design and ER Model: Entity, Attributes, Relationships, Constraints, Keys, Design Process.

- Data is the basic building block of any DBMS and it is fact that can be recorded.
  eg: figures, statistics, speech, video, text etc.(e.g.1, ABC,19 etc).
- Information: Meaningful data.
  Record: Collection of related data items, e.g. in the above example the three data items had no meaning. But if we organize them in the following way, then they collectively represent meaningful information.

T1

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	28

T2

Roll	Address
1	KOL
2	DEL
3	MUM

T3

Roll	Year
1	I
2	II
3	I

T4

Year	Hostel
I	H1
II	H2

Files used by the Sales Department.

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo
PA14	16 Holhead Rd	Aberdeen	AB7 5SU	House	6	650	CO46
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93

PrivateOwner

ownerNo	fName	lName	address	telNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Client

clientNo	fName	lName	address	telNo	prefType	maxRent
CR76	John	Kay	56 High St, London SW1 4EH	0207-774-5632	Flat	425
CR56	Aline	Stewart	64 Fern Dr, Glasgow G42 0BL	0141-848-1825	Flat	350
CR74	Mike	Ritchie	18 Tain St, PA1G 1YQ	01475-392178	House	750
CR62	Mary	Tregear	5 Tarbot Rd, Aberdeen AB9 3ST	01224-196720	Flat	600

Files used by the Contracts Department.

Lease

leaseNo	propertyNo	clientNo	rent	payment Method	deposit	paid	rentStart	rentFinish	duration
10024	PA14	CR62	650	Visa	1300	Y	1-Jun-13	31-May-14	12
10075	PL94	CR76	400	Cash	800	N	1-Aug-13	31-Jan-14	6
10012	PG21	CR74	600	Cheque	1200	Y	1-Jul-13	30-Jun-14	12

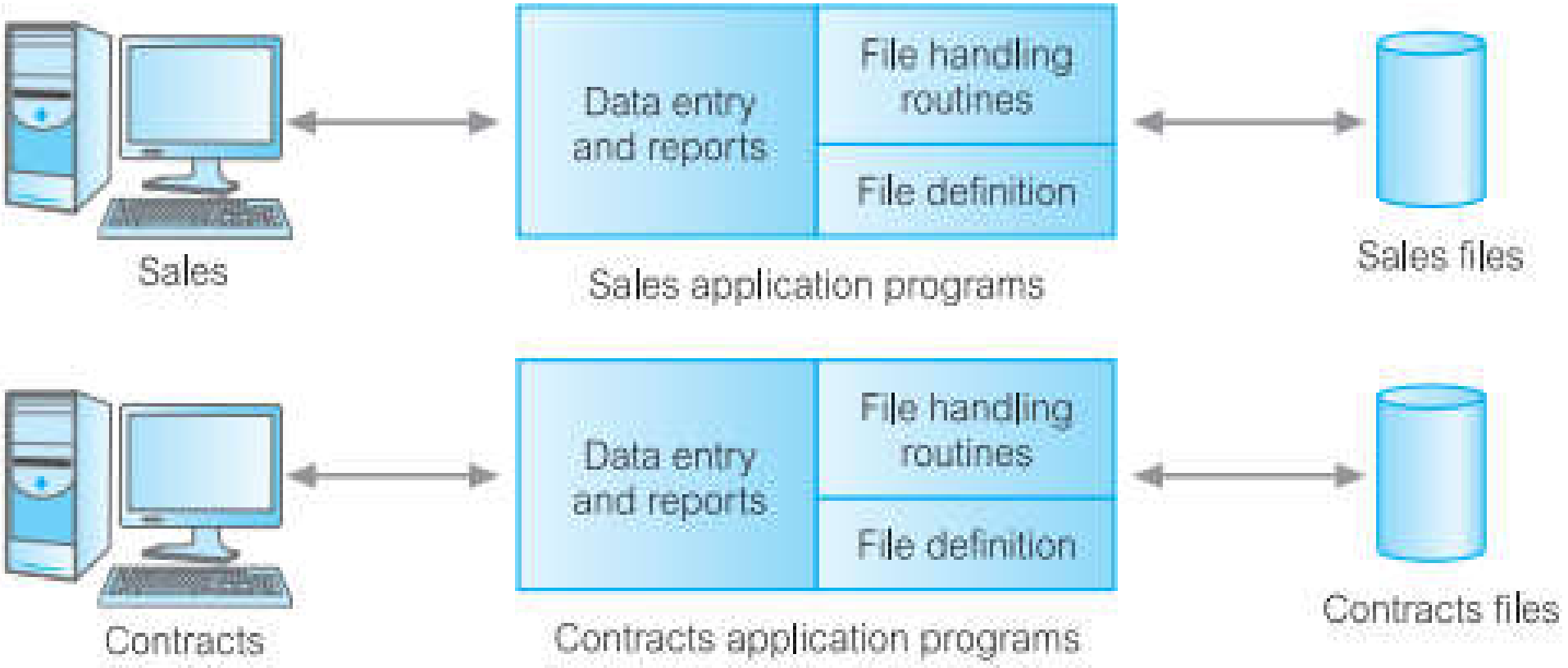
PropertyForRent

propertyNo	street	city	postcode	rent
PA14	16 Holhead	Aberdeen	AB7 5SU	650
PL94	6 Argyll St	London	NW2	400
PG21	18 Dale Rd	Glasgow	G12	600

Client

clientNo	fName	lName	address	telNo
CR76	John	Kay	56 High St, London SW1 4EH	0171-774-5632
CR74	Mike	Ritchie	18 Tain St, PA1G 1YQ	01475-392178
CR62	Mary	Tregear	5 Tarbot Rd, Aberdeen AB9 3ST	01224-196720

File-based processing.



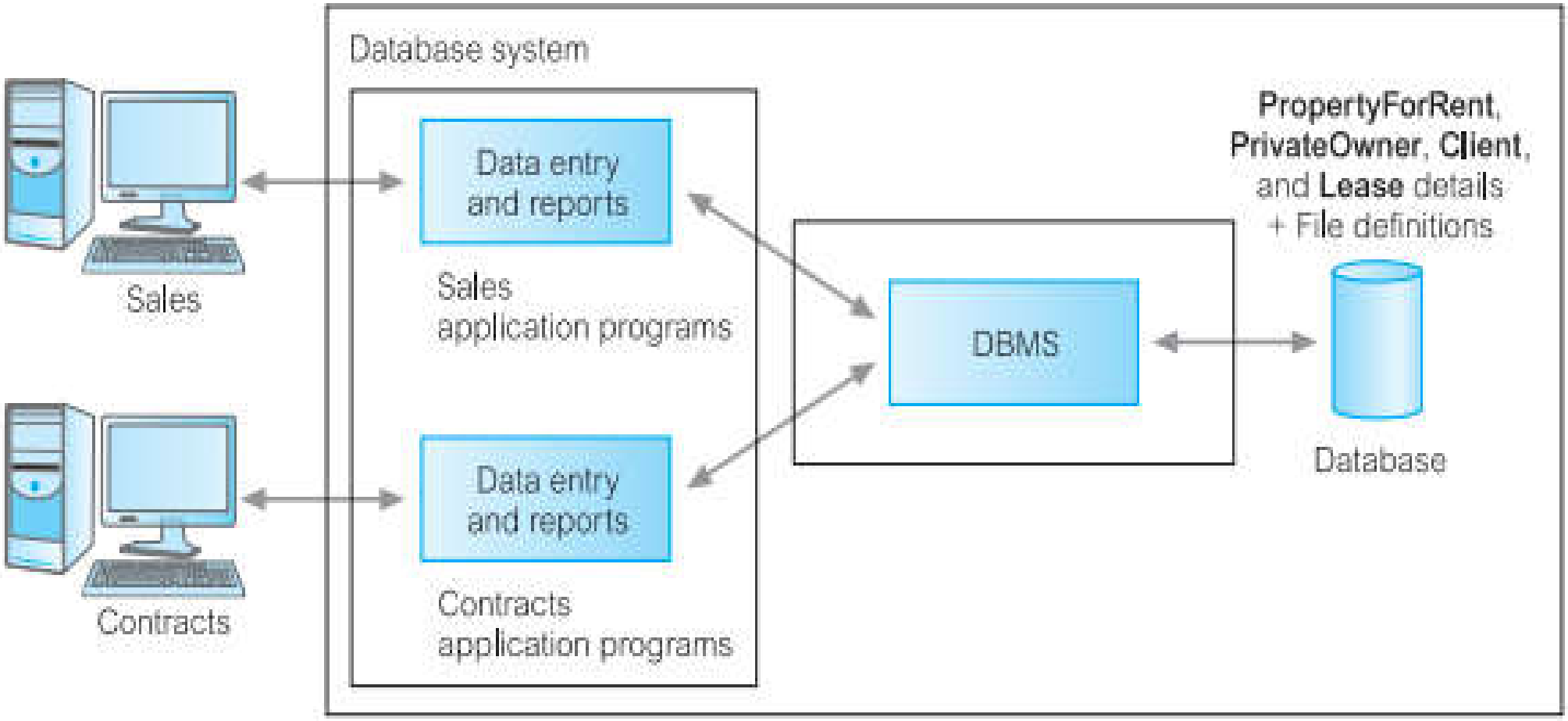
Limitations of the File-Based Approach

- **Separation and isolation of data**
- **Difficult in searching (search of train in DB VS FILE)**
- **Data redundancy and inconsistency**
- **Data dependency**
- **Expecting exact path of data**
- **Data integrity (eg: updating existing account)**
- **Atomicity (both operation or no operation)**
- **Concurrency access**
- **Security problems (role based)**
- **Incompatible file formats**
- **Fixed queries/proliferation of application programs**

❑ There was no provision for security or integrity.
- Recovery, in the event of a hardware or software failure, was limited or nonexistent.
- Access to the files was restricted to one user at a time

❑ There was no provision for shared access by staff in the same department.

Database Approach



- A **Database-Management System (DBMS)** is a collection of interrelated data and a set of programs to access those data.
- The collection of data, usually referred to as the **database**, contains information relevant to an enterprise.
- The primary goal of a **DBMS** is to provide a way to store and retrieve database information in *convenient* and *efficient manner*.
- Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.
- In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.

Database-System Applications

Enterprise Information

- *Sales*: For customer, product, and purchase information.
- *Accounting*: For payments, receipts, account balances and accounting information.
- *Human resources*: For information about employees, salaries, payroll taxes, and benefits.
- *Manufacturing*: For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.
- *Online retailers*: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.

Banking and Finance

- *Banking*: For customer information, accounts, loans, and banking transactions.

- *Credit card transactions*: For purchases on credit cards and generation of monthly statements.

- *Finance*: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers.

- ***Universities*:** For student information, course registrations, and grades.

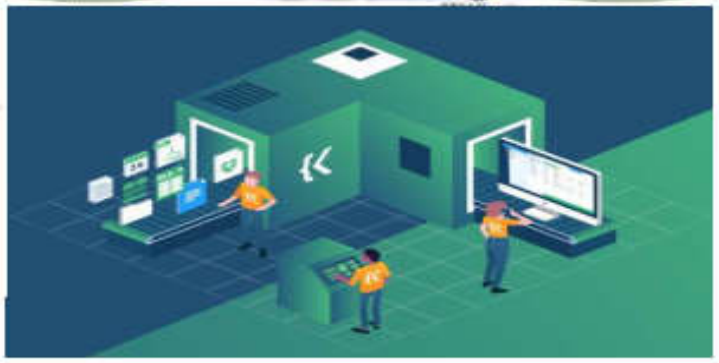
- *Airlines*: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.

- *Telecommunication*: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

Web-based services

- Social-media: For keeping records of users, connections between users.

- Online retailers: For keeping records of sales data and orders as for any retailer, but also for tracking a user's product views, search terms, etc. for the purpose of identifying the best items to recommend to that user.



•**Online advertisements:** For keeping records of click history to enable targeted advertisements, product suggestions, news articles, etc. People access such databases every time they do a web search, make an online purchase, or access a social-networking site.

- **Document databases:** For maintaining collections of new articles, patents, published research papers, etc.

• **Navigation systems:** For maintaining the locations of various places of interest along with the exact routes of roads, train systems, buses, etc.

Purpose of Database Systems (DBMS+DB)

To avoid

- **Data redundancy and inconsistency:** same information may be duplicated in several files. **data** redundancy results to higher storage and access cost lead to data inconsistency eg: students info, gmail account
- **Difficulty in accessing:** conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. eg: clerk needs specific info.
- **Data isolation**: data are scattered in various files, and files may be in different formats
- **Integrity problems**: When new constraints are added, it is difficult to change the programs to enforce them .The problem is compounded, when constraints involve several data items from different files. eg: various dept account info
- **Atomicity problems**: if a failure occurs, the data be restored to the consistent state that existed prior to the failure. **eg:** fund transfer
- **Concurrent-access anomalies**. **eg:** fund transfer concurrently
- **Security problems**. eg: LMS, university portal

View of Data

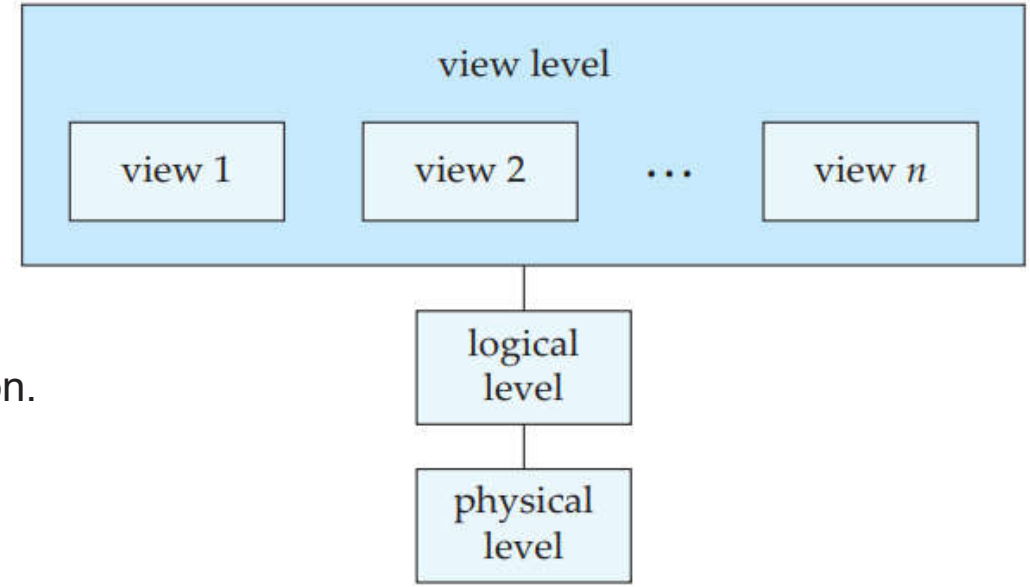
A major purpose of a database system is to provide users with an *abstract* view of the data. That is, the system hides certain details of how the data are stored and maintained.

- **Physical level**. The lowest level of abstraction describes *how* the data are actually stored. The physical level describes complex low-level data structures in detail.

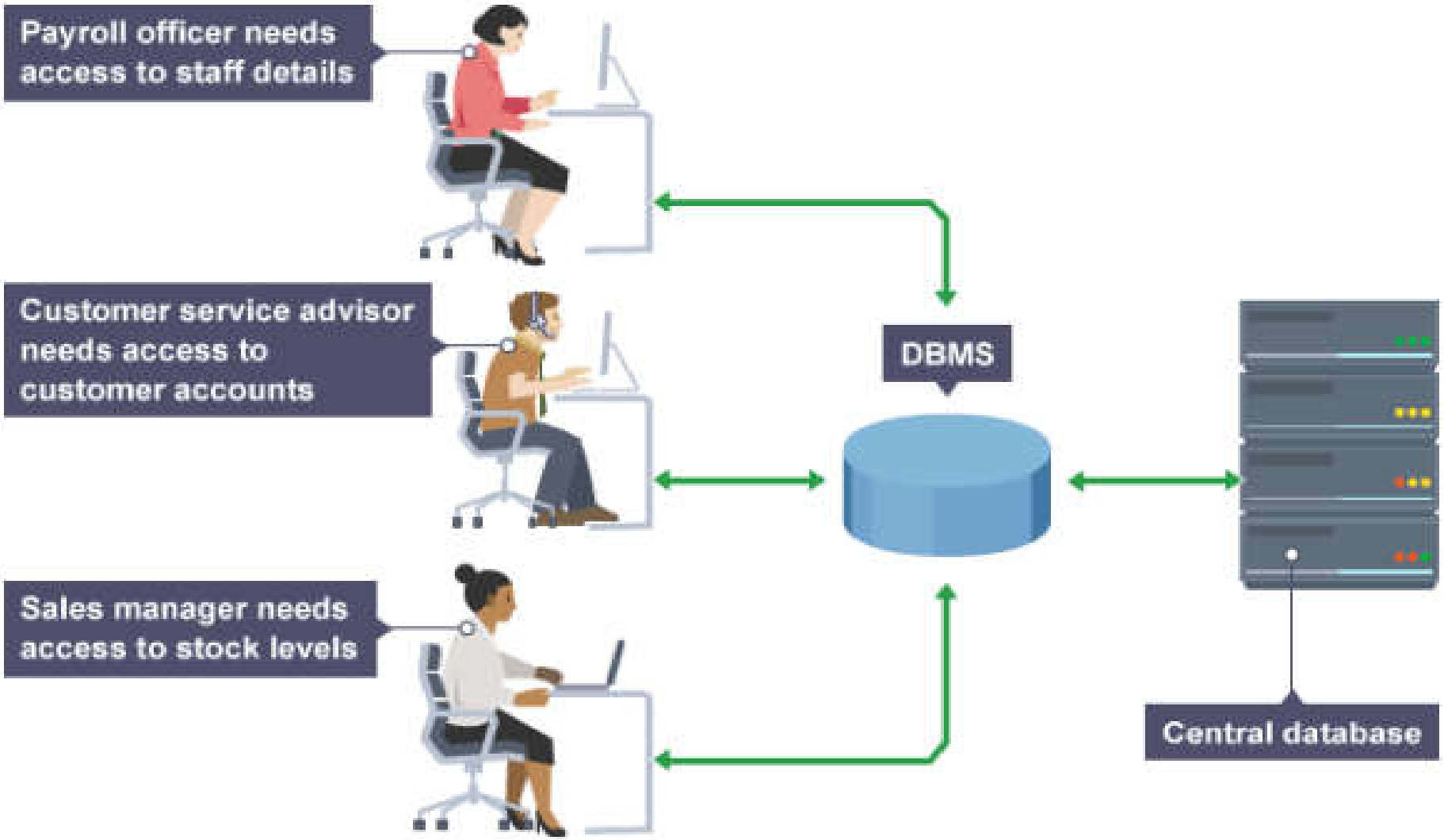
- **Logical level**.
- Describes *what* data are stored in the database, and what relationships exist among those data.
- Describes the entire database in terms of a small number of relatively simple structures. This is referred to as **physical data independence**.
- Database administrators must decide what information to keep in the database and use the logical level of abstraction.

View level

- The highest level of abstraction describes only part of the entire database.
- Many users of the database system do not need all this information; instead, they need to access only a part of the database.
- View level of abstraction exists to simplify their interaction with system.
- The system may provide many views for the same database.



The three levels of data abstraction.


Payroll officer needs access to staff details
Customer service advisor needs access to customer accounts
Sales manager needs access to stock levels
DBMS
Central database

Example of View of Data

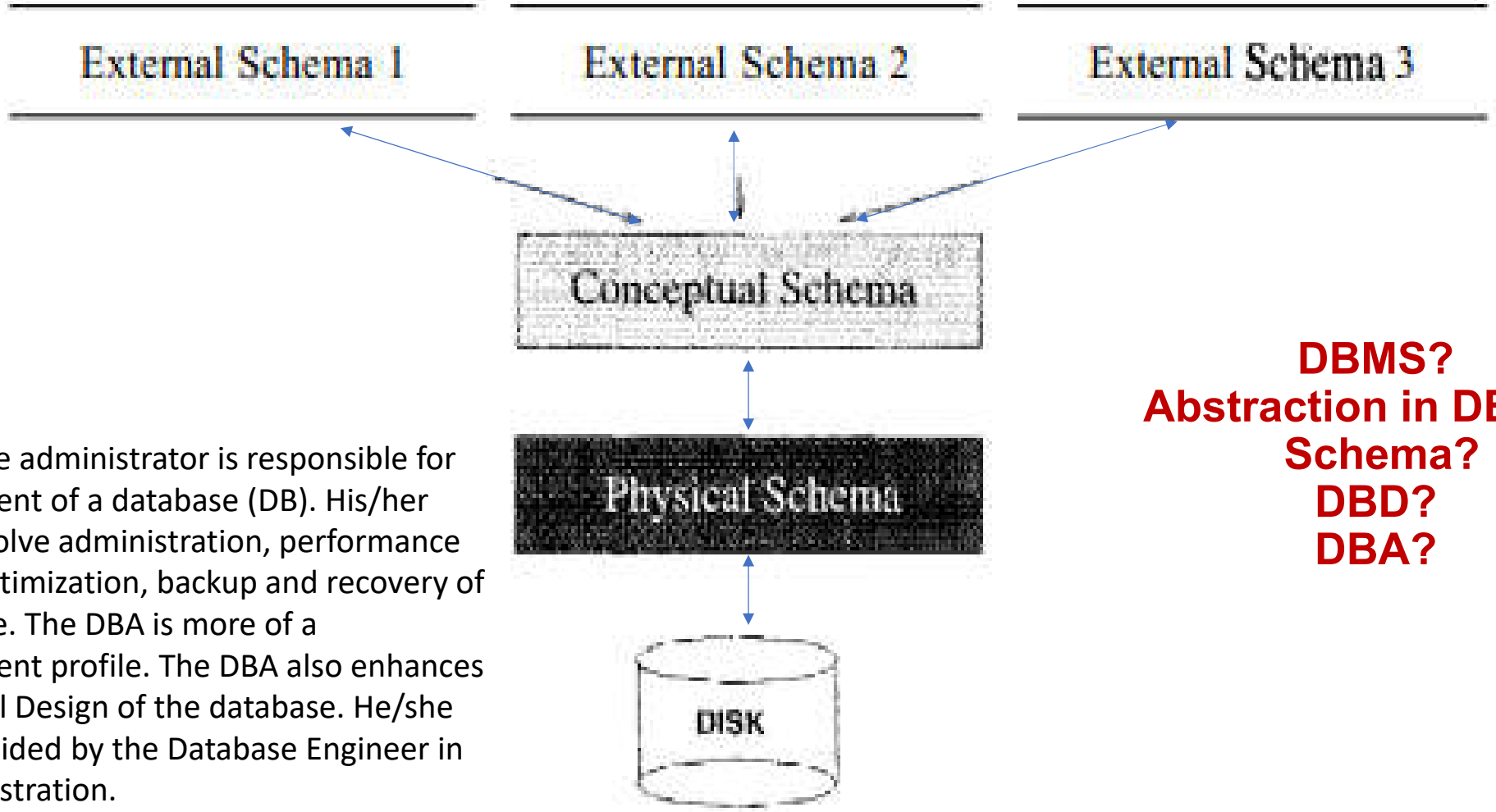
- *department*, with fields *dept name*, *building*, and *budget*
- *course*, with fields *course id*, *title*, *dept name*, and *credits*
- *student*, with fields *ID*, *name*, *dept name*, and *tot cred*
- *instructor, with fields ID, name, dept name, salary*

```
type instructor = record
                    ID : char (5);
                    name : char (20);
                    dept_name : char (20);
                    salary : numeric (8,2);
                  end;
```

Instances and Schemas

- The collection of information stored in the database at a particular moment is called an **instance** of the database.
- The overall design of the database is called the database **schema**.
- The **physical schema** describes the database design at the physical level.
- The **logical schema** describes the database design at the logical level.
- A database may also have several schemas at the view level called as **subschemas**, that describe different views of the database.

Levels of Abstraction in a DBMS



A database administrator is responsible for management of a database (DB). His/her duties involve administration, performance tuning, optimization, backup and recovery of a database. The DBA is more of a management profile. The DBA also enhances the Logical Design of the database. He/she may be guided by the Database Engineer in his administration.

DBMS?
Abstraction in DBMS?
Schema?
DBD?
DBA?

DATABASE ADMINISTRATOR	DATABASE ENGINEER
DBA is responsible for managing the database.	Database Engineer is responsible for developing the database.
defines the management aspect of the profile.	defines the technical aspects of the profile.
handles performance and security of a Database.	handles physical and logical models of a Database.
develops and manages the recovery plan and back ups.	identifies and handles the errors in a database system.
responsible for the system's interaction with the front end users.	Database Engineers are not concerned much with the end users.
DBA is guided by the Database Engineer for effective management of the databases.	The Database Engineer guides the DBA for the effective management of the databases.
The overall role of a DBA is generally narrow.	The overall role of Database Engineer is relatively broader.

Database Users and Administrators

- **Naive users** are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

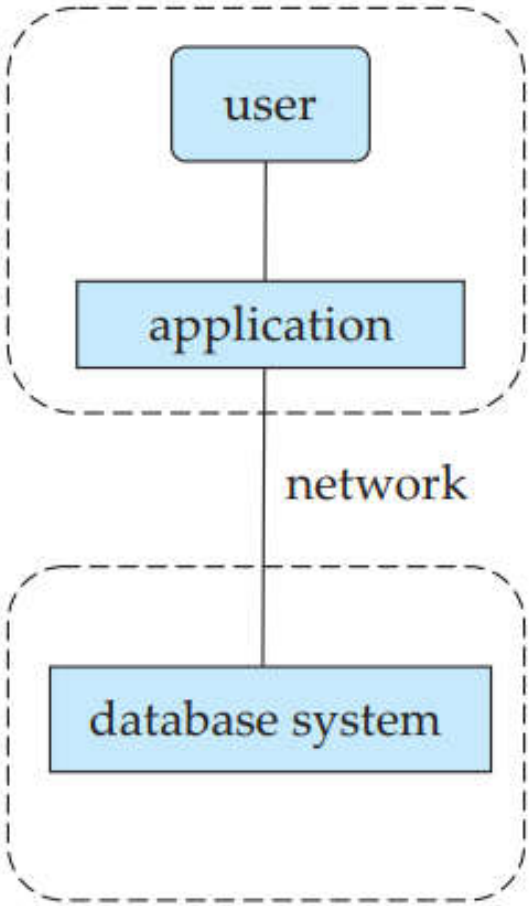
- **Application programmers** are computer professionals who write application programs. **Rapid application development (RAD)** tools are tools that enable an application programmer to construct forms and reports with minimal programming effort.

- **Sophisticated users** interact with the system without writing programs. Instead, they form their requests either using a database query language or by using tools such as data analysis software.

- **Specialized users** are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.

- **Database Administrator:** A person who has such central control over the system (to access data and program that access those data) is called a **database administrator (DBA)**. Functions of DBA:
- **Schema definition, Storage structure and access-method definition, Schema and physical-organization modification, Granting of authorization for data access, Routine maintenance**

2 Tier DBMS Architectue



(a) Two-tier architecture

client

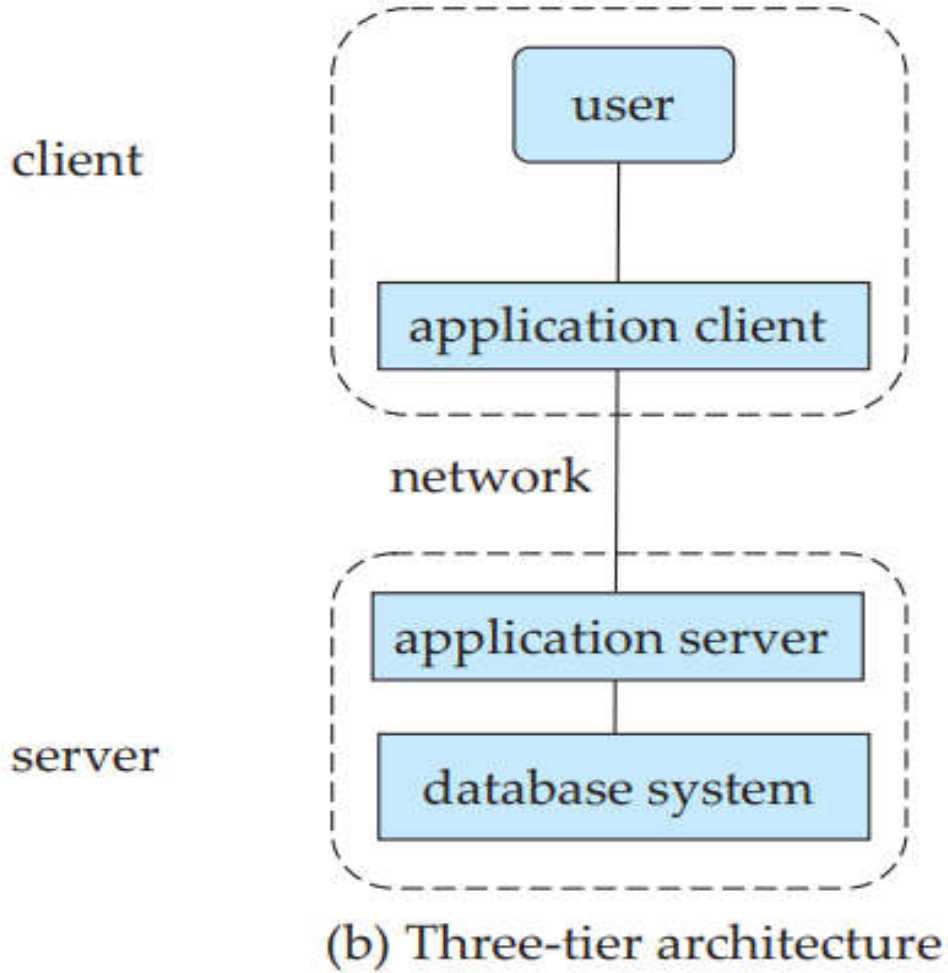
Eg: Banking system, Railway system etc

Advantages: simple, limited client, easy maintenance

Disadvantages: scalability, security

server

3 Tier DBMS Architectue



(b) Three-tier architecture

Eg: web application, gmail data etc

Advantages: data independence(Abstraction), security

Disadvantages: maintenance is difficult

Data Models

A collection of **conceptual tools for describing data, data relationships, data semantics(data organization) and consistency constraints.**

A data model provides a way to describe the design of a database at the physical, logical, and view levels.

Relational Model

The relational model uses a collection of tables to represent both data and the relationships among those data.

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

- Each table has multiple columns, and each column has a unique name.
- Tables are also known as **relations**.
- Each table contains records of a particular type.
- Each record type defines a fixed number of fields, or attributes.
- The columns of the table correspond to the attributes of the record type.

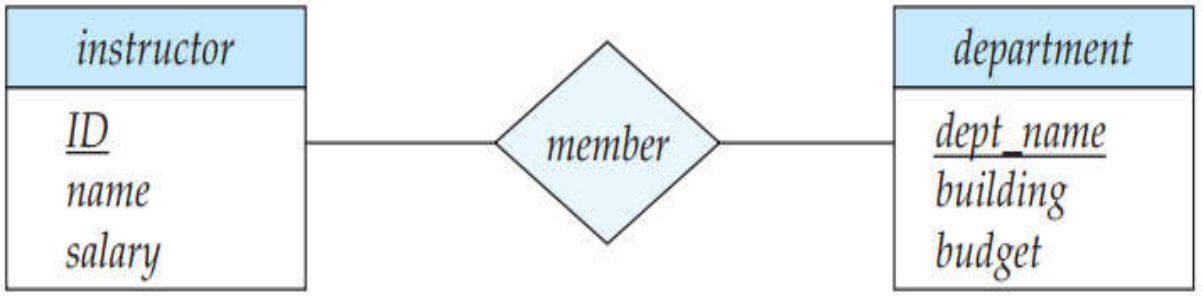
- **Entity-Relationship Model**. The entity-relationship (E-R) data model uses a collection of basic objects, called *entities*, and *relationships* among these objects. An entity is a “thing”/“object” in the real world that is distinguishable from other objects.

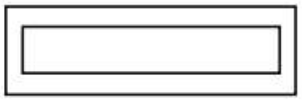
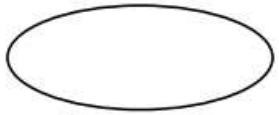
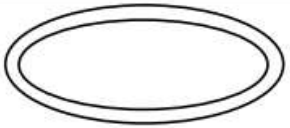
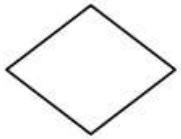
- **Object-Based Data Model**.
  - Object-oriented programming has become the dominant software-development methodology.
  - Object based approach led to extending the E-R model with notions of encapsulation, methods (functions), and object identity.
  - The object-relational data model combines features of the object-oriented data model and relational data model.

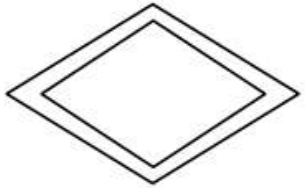
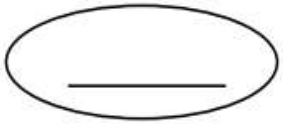
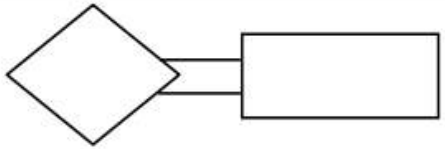
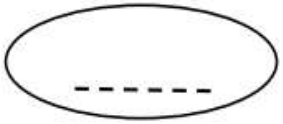
- **Semi structured Data Model**.
  - Permits the specification of data where individual data items of the same type may have different sets of attributes.
  - The **XML,EMAIL,CSV FILE** is widely used to represent semi structured data.

The Entity-Relationship (E-R) data model uses a collection of basic objects, called *entities*, and *relationships* among these objects.

- An entity is a “thing” or “object” in real world, posses physical existence.
- Entities are described in a database by a set of **attributes(**The attribute is used to describe the property of an entity.)
- A **relationship** is used to describe the relation between entities.
- It is used to construct logical/conceptual view of the data.(eg: construction of home)
- **Draw Student-Course ER model ???????**

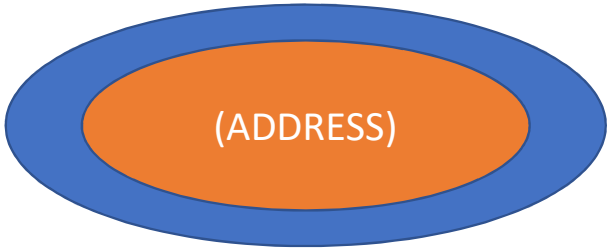


Sr. No	Symbol	Meaning
1		Set of Entity
2		Weak Entity Set
3		Attribute
4		Multivalued Attribute
5		Derived Attribute
6		Relationship

Sr. No	Symbol	Meaning
7		Weak entity Relationship
8		Primary key Attribute
9		Relationship and total participation of entities
10		Weak Entity's Discrimination Attribute

Types of Attributes and Representation

- Single valued (PRN/RN)
- Multivalued (address)
- Simple (cannot be divided further, eg: age)
- Composite/Compound (can be divided further, eg: name)
- Stored (DOB)
- Derived (age from DOB, course end date from start and duration)
- Key (unique value in Attribute dig:_____)
- Non key
- Required (eg: Name of student dig: *)
- Optional
- Complex (compound+multivalued) eg: address



Participation

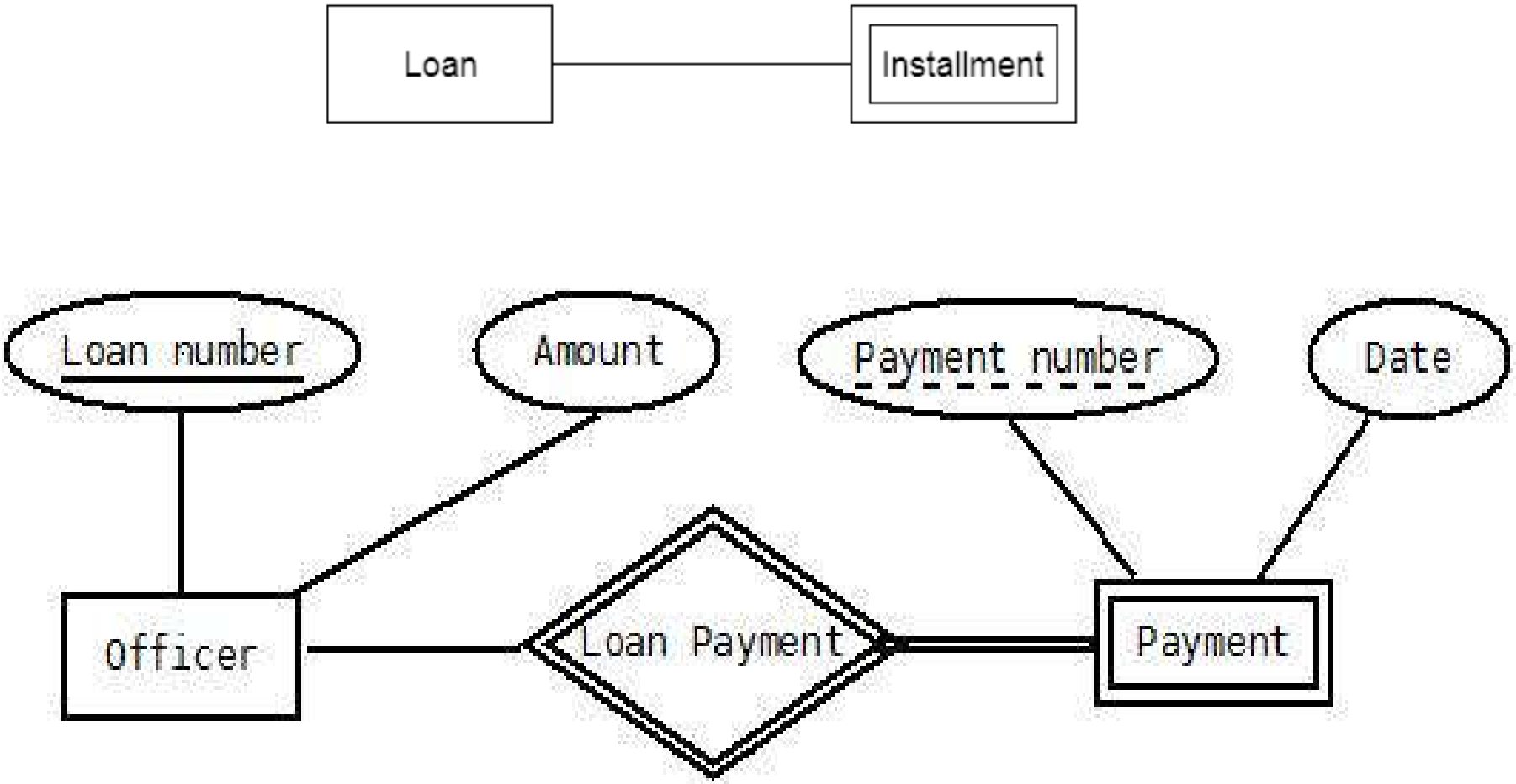
- Partial/optional participation: when ***not all entities*** are involved in the relationship (we represent it by a ***single line***)

"each employee works at zero or one factory"
"each factory has zero or more employees working at it"

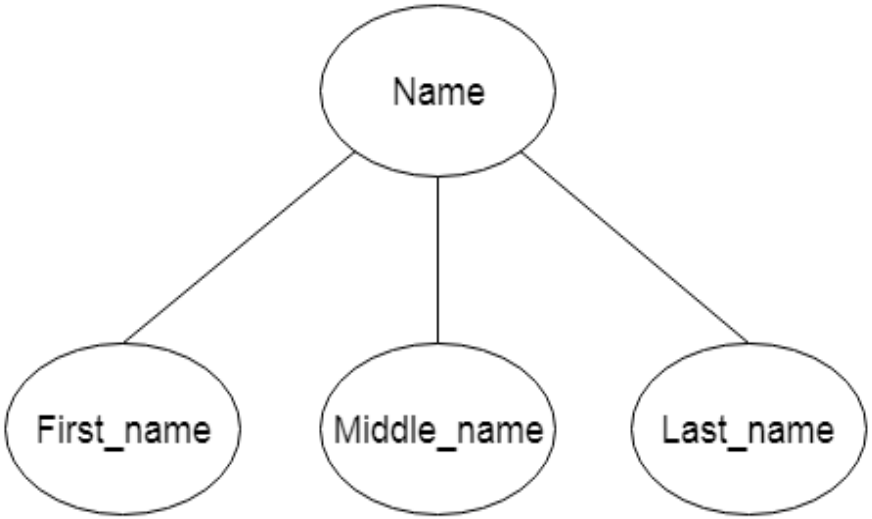
- Total/mandatory participation: when ***each entity*** in the entity set is involved in the relationship (we represent it by a ***double line***)

"each employee manages *zero or one* factories"
"each factory has *exactly one* employee managing it"

Weak Entity: An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own.

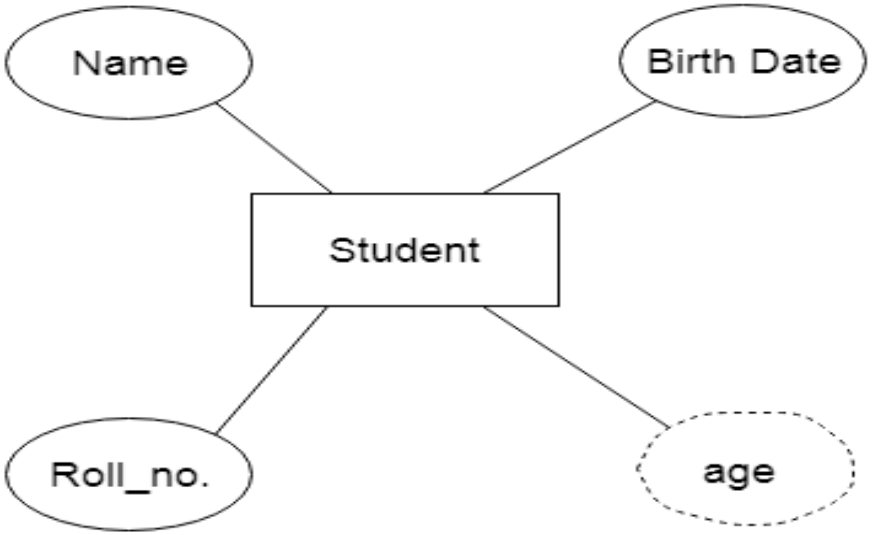


- **Composite Attribute:** An attribute that composed of many other attributes is known as a composite attribute



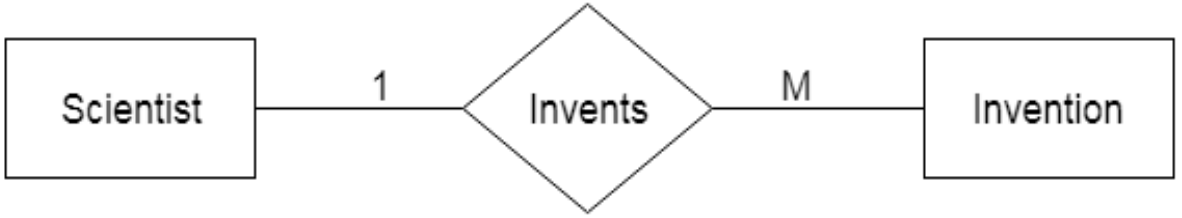
- **Multivalued Attribute:** An attribute can have more than one value

- **Derived Attribute:** An attribute that can be derived from other attribute is known as a derived attribute

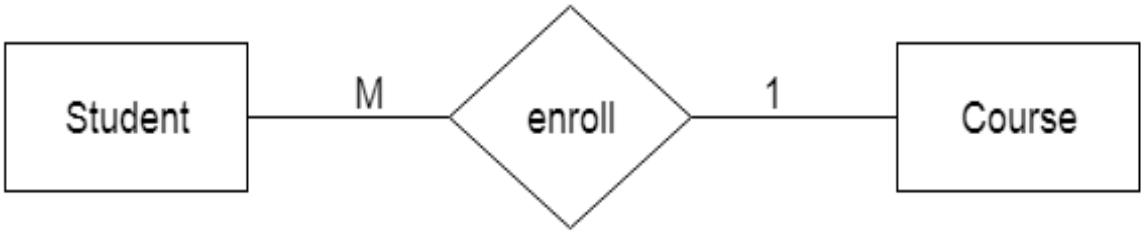


- **One-to-One Relationship:** When only one instance of an entity is associated with the relationship

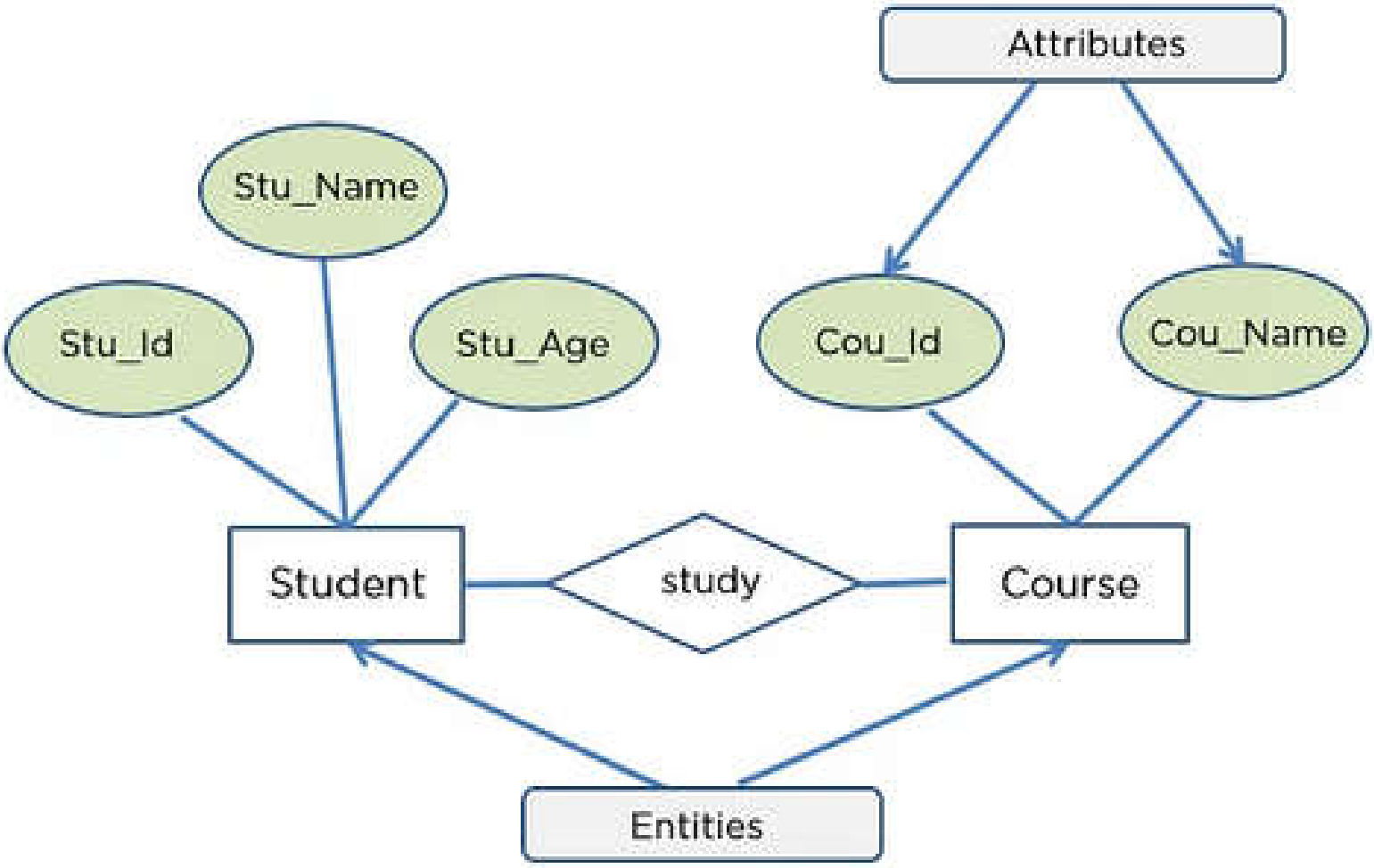
- **One-to-many relationship:** When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship

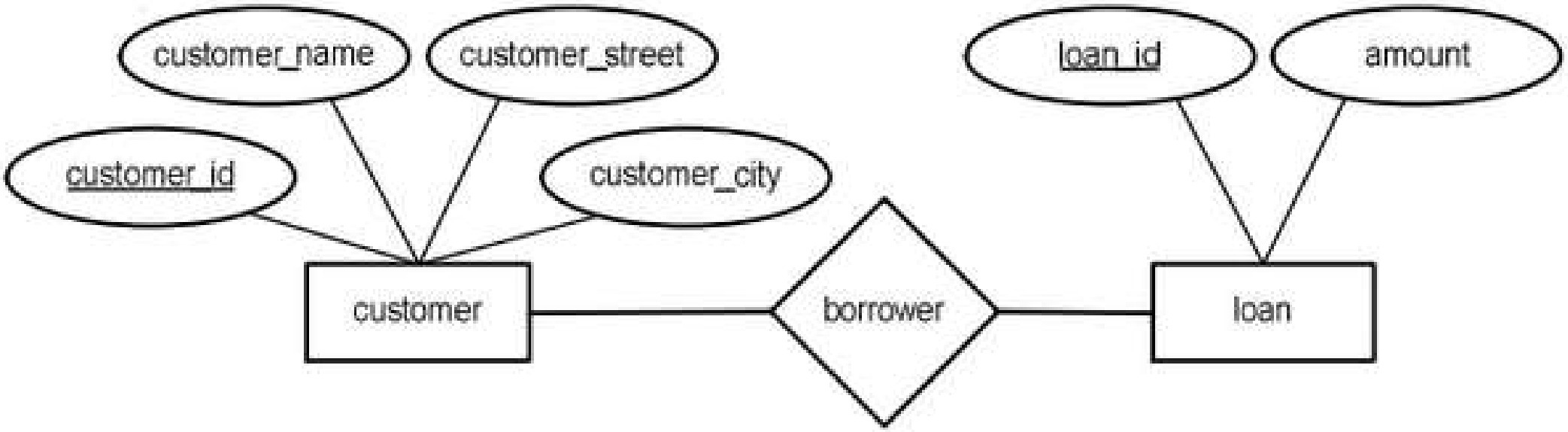


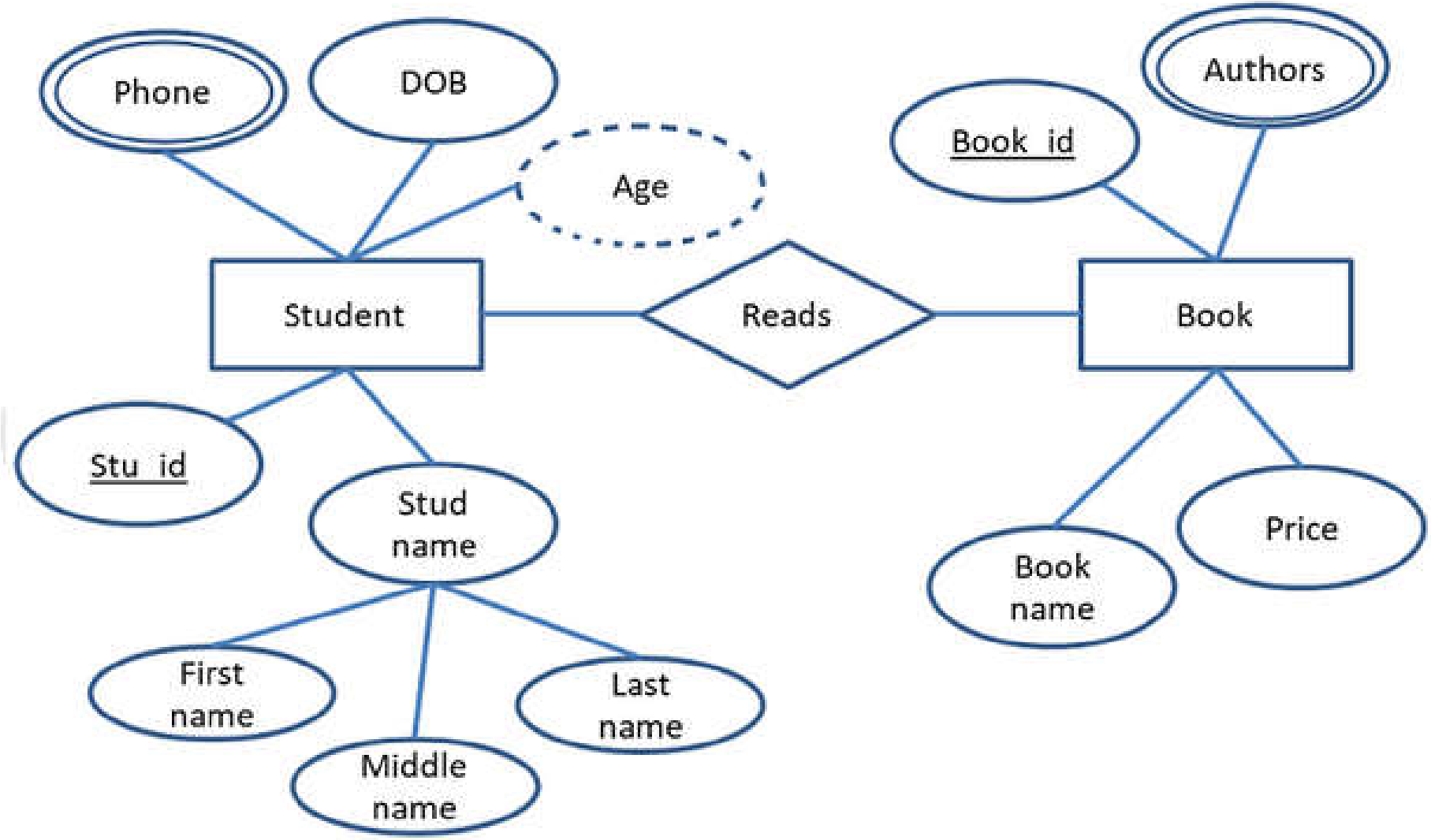
- **Many-to-one relationship:** When more than one instance of the entity on the left, and only one instance of an entity on the right



- **Many-to-many relationship:** When more than one instance of the entity on the left, and more than one instance of an entity on the right associates


Attributes
Stu_Name
Stu_Id
Stu_Age
Cou_Id
Cou_Name
Student
study
Course
Entities


customer_name
customer_street
loan_id
amount
customer_id
customer_city
customer
borrower
loan


Phone
DOB
Age
Authors
Book_id
Student
Reads
Book
Stu_id
Stud name
Price
Book name
First name
Last name
Middle name

- ER Diagram for the Airline Reservation System
- ER Diagram for the University Management System
- ER Diagram for Employee Management System
- ER Diagram for the Railway Reservation System
- ER Diagram for the Library Management System
- ER Diagram for the Hotel Management System
- ER Diagram for the College Management System
- ER Diagram for the Bank
- ER Diagram for the Security Management System
- ER Diagram for the Production Management System
- ER Diagram for the Product Management System
- ER Diagram for the Grocery Management System

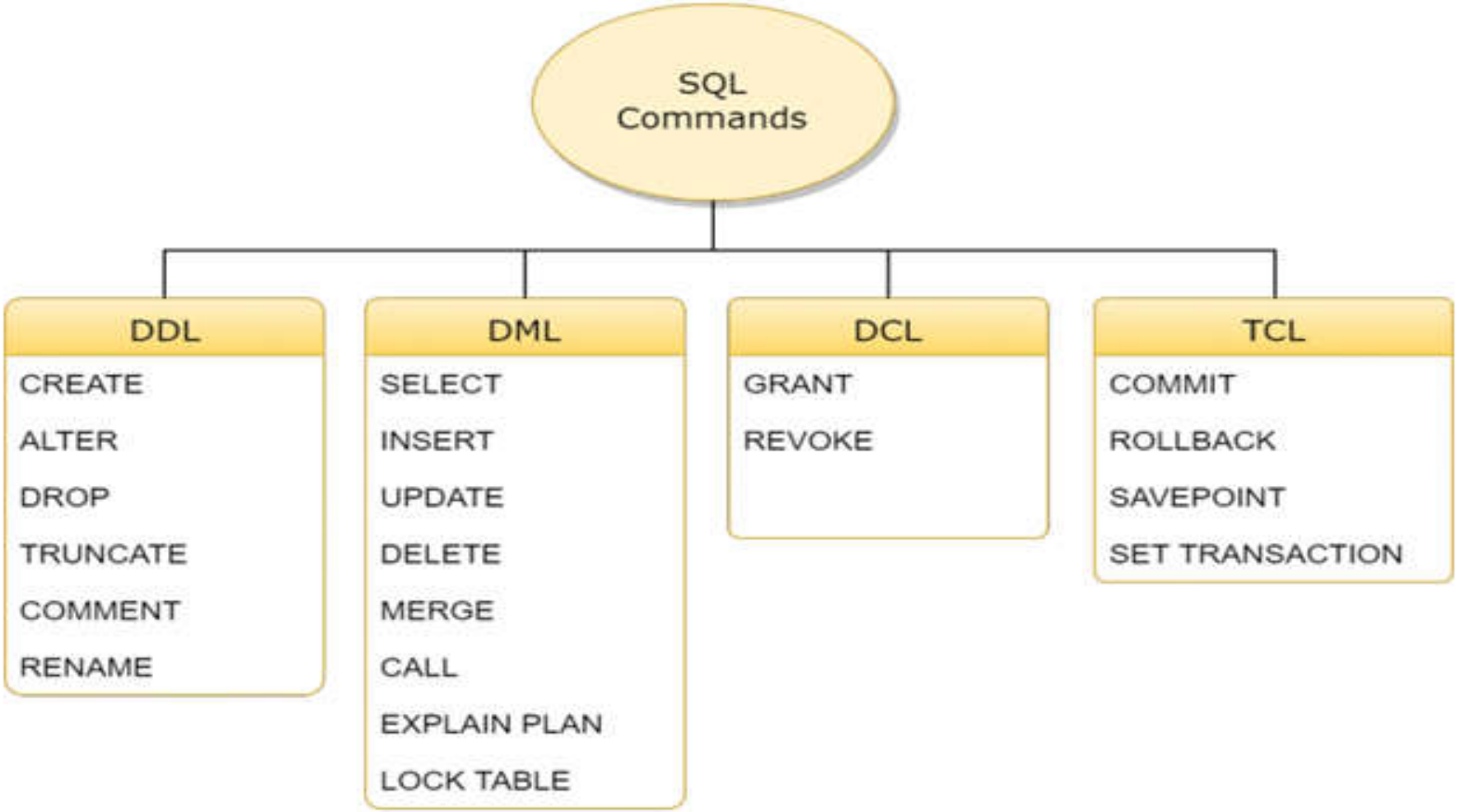
Database Languages

A database system provides a **Data-Definition Language (DDL)** to specify the database schema and a **Data-Manipulation Language (DML)** to express database queries and up-dates.

A **Data-Manipulation Language (DML)** is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:

- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information from the database
- Modification of information stored in the database

- **Procedural DMLs** require a user to specify ***what*** data are needed and ***how*** to get those data.
- **Declarative DMLs** (**Nonprocedural DMLs**) require a user to specify ***what*** data are needed ***without*** specifying how to get those data.


SQL
Commands
DDL
CREATE
ALTER
DROP
TRUNCATE
COMMENT
RENAME
DML
SELECT
INSERT
UPDATE
DELETE
MERGE
CALL
EXPLAIN PLAN
LOCK TABLE
DCL
GRANT
REVOKE
TCL
COMMIT
ROLLBACK
SAVEPOINT
SET TRANSACTION

DDL Statements-

DDL is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

- CREATE – used to create a database table
- ALTER - alters the structure of the existing database
- DROP - delete objects from the database.
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object.

Data Definition Language (DDL)- CREATE

CREATE It is used to create a new table in the database.

Syntax:

CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

Example:

CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHA
R2(100), DOB DATE);

Data Definition Language (DDL)- Drop

Drop: It is used to delete both the structure and record stored in the table.

Syntax:

DROP TABLE ;

Example:

DROP TABLE EMPLOYEE;

Data Definition Language (DDL)- **ALTER**

ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

ALTER TABLE table_name ADD column_name COLUMN-definition;

ALTER TABLE MODIFY(COLUMN DEFINITION....);

Example:

ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));

ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

Data Definition Language (DDL)- **TRUNCATE**

TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Syntax:

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE TABLE EMPLOYEE;
```

DML Statements-

DML is short name of Data Manipulation Language which deals with manipulation on data

- SELECT - retrieve data from a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - Delete all records from a database table
- MERGE - UPSERT operation (insert or update)
- CALL - call a PL/SQL or Java subprogram
- EXPLAIN PLAN - interpretation of the data access path
- LOCK TABLE - concurrency Control

Data Manipulation Language - **INSERT**

INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

```
INSERT INTO TABLE_NAME    (col1, col2, col3,.... col N)
VALUES (value1, value2, value3, .... valueN);
```

OR

```
INSERT INTO TABLE_NAME    VALUES (value1, value2, value3, .... valueN);
```

Example:

```
INSERT INTO XYZ (Author, Subject) VALUES ("Sonoo", "DBMS");
```

Data Manipulation Language - **UPDATE**

Update: This command is used to **update or modify** the value of a column in the table.

Syntax:

```
UPDATE table_name SET [column_name1= value1,...column_n
ameN = valueN] [WHERE CONDITION]
```

Example:

```
UPDATE students
SET User_Name = 'Sonoo'
WHERE Student_Id = '3'
```

Data Control Language - **Grant**

GRANT: It is used to give user access privileges to a database.

Example:

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOT
HER_USER;
```

REVOKE: It is used to take back permissions from the user.

Example:

```
REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

Transaction Control Language - COMMIT

Commit: Commit command is used to save all the transactions to the database.

Syntex:

```
COMMIT;
```

Example:

```
DELETE FROM CUSTOMERS
WHERE AGE = 25;
COMMIT;
```

Transaction Control Language - **Rollback**

Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntex:

```
ROLLBACK;
```

Example:

```
DELETE FROM CUSTOMERS
WHERE AGE = 25;
ROLLBACK;
```

SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntex:

```
SAVEPOINT SAVEPOINT_NA
ME;
```

Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

SELECT

a. **SELECT:** This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

```
SELECT expressions    FROM TABLES    WHERE conditions;
```

Example:

```
SELECT emp_name  FROM employee  WHERE age > 20;
```

Data-Manipulation Language for Relational Database

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	building	budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

select *instructor.name*
from *instructor*
where *instructor.dept_name* = 'History';

El Said Califieri

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	building	budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

select *instructor.ID*, *department.dept_name*
from *instructor*, *department*
where *instructor.dept_name= department.dept_name*
and *department.budget* > 95000;

(12121, Finance),
(45565, Computer Science),
(10101, Computer Science),
(83821,Computer Science),
(76543, Finance)

Key (One of the Attribute in table)

Used for Unique identification of tuple in the table

Candidate key (Collection of UID,RN,LIC NO, EMAIL ID,PAN) (*Unique AND may be Null)

Primary key (Most appropriate key from Candidate key eg: RN/REG NO) (*Unique + Not null). In a single Table there will be only one Primary key.

Foreign key (attribute or set of attribute that refers to the primary key of same or different table) FK maintains Referential Integrity. In a single table there can be more than one FK. A foreign key is applied to a column of one table which references the primary key of a column in another table.

IF TABLE IS ALREADY AVAILABLE then….?

NAME CAN BE DIFFERENT

alter table department add constraint fk foreign key (sid) references student(sid));

student

Department

pk

fk (foreign key)

pk

sid	sname	sage
1	A	22
2	B	21
3	A	22

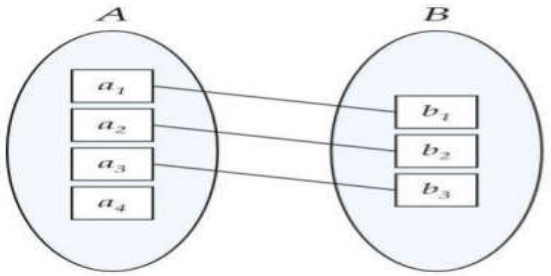
Referenced (base table)

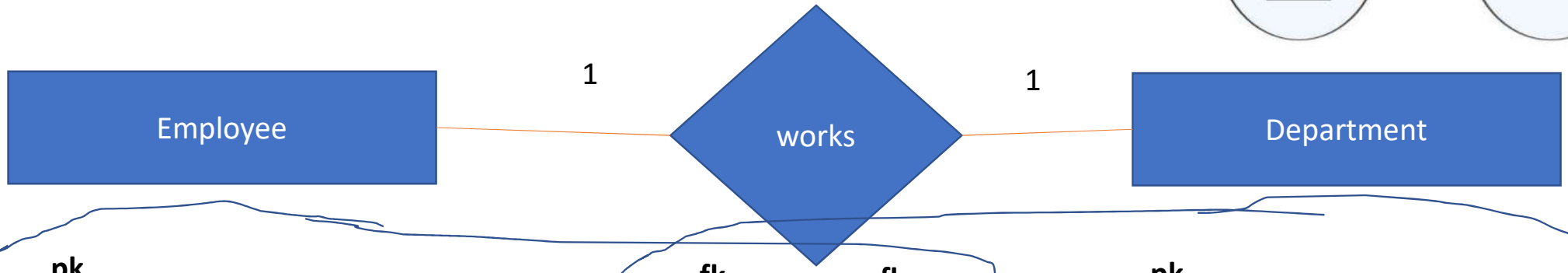
did	dname	dloc	sid
D1	IT	Delhi	1
D2	ETC	Mumbai	2
D3	MECH	Shirdi	3

Referencing table

create table department (did varchar(11) primary key not null, dname varchar(22), dloc varchar(11),sid int references student(sid));

Relationship(Cardinality) (1:1)





pk

eid	Ename	Eage
E1	A	22
E2	B	21
E3	A	22

Referenced (base table)

fk fk

eid	did
E1	D1
E2	D3
E3	D2

Referencing table

pk

did	Dname	dloc
D1	IT	Delhi
D2	ETC	Mumbai
D3	MECH	Shirdi

Referenced (base table)

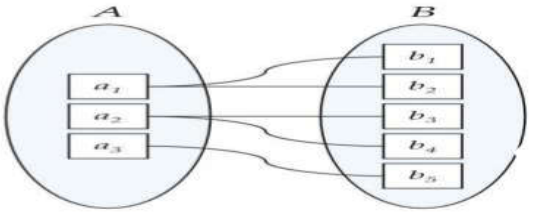
Which employee works in which department
(Descriptive attribute can be added to table)
NO REPETTTION OF VALUES IN THE TABLE

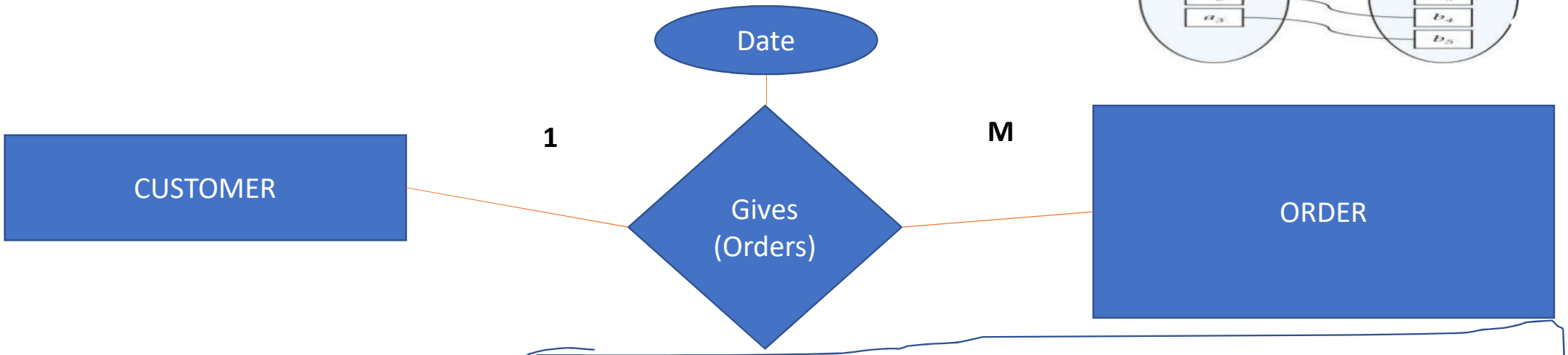
Primary key ? (eid/did)
Merging of table ? (Yes)

Merged table result

eid	Ename	Eage	did
E1	A	22	D1
E2	B	21	D3
E3	A	22	D2

Relationship (Cardinality)(1:M)





Cid	Cname	Cloc
C1	A	Delhi
C2	B	Mumbai
C3	C	Shirdi

Referenced (base table)

Cid	Ono	Date
C1	O1	11/7/22
C1	O2	22/8/22
C2	O3	15/8/22
C2	O4	22/8/22

Referencing table

Ono	Oname	Ocost
O1	Shirt	1000
O2	Jeans	2000
O3	T shirt	500

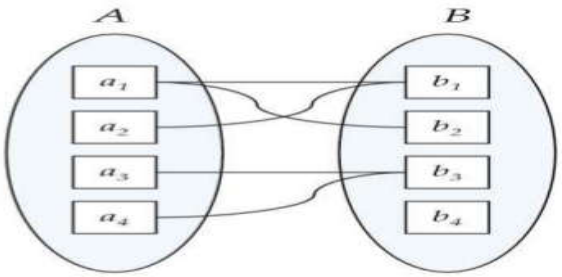
Referenced (base table)

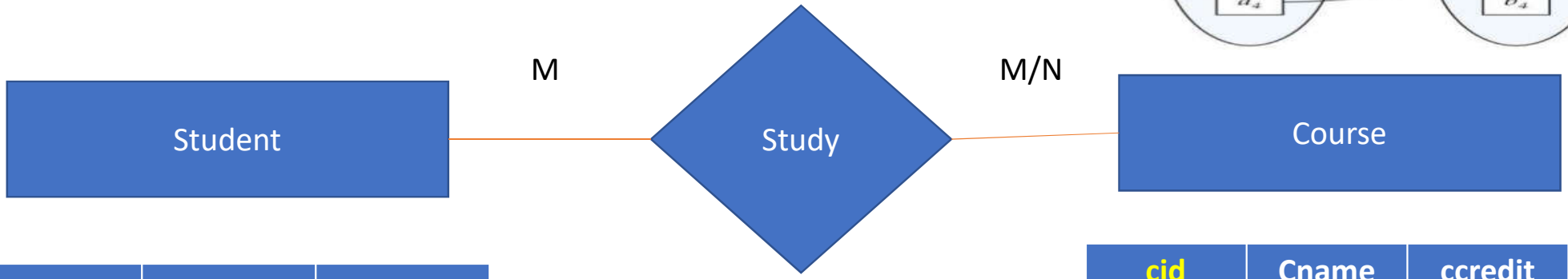
Primary key (Ono in relation table, since it is from 'M' table)?
Merging of table []?

Merged table result

Cid	Ono	Date	Oname	Ocost
C1	O1	11/7/22	Shirt	1000
C1	O2	22/8/22	Jeans	2000
C2	O3	15/8/22	T shirt	500

Relationship (M:M/N)





Rollno	Name	Age
1	A	11
2	B	33
3	C	22

Referenced (base table)

Rollno	cid
1	C1
2	C1
3	C2
1	C2
2	C3

Referencing table

cid	Cname	ccredit
C1	Math	3
C2	Phy	4
C3	Chem	5

Referenced (base table)

Primary key(Rollno/cid) not possible individually but combinely CAN?
Merging of table(NOT POSSIBLE) ?

Constraints

- Constraints are the rules enforced on the data columns of a table.
- These are used to limit the type of data that can go into a table.
- This ensures the accuracy and reliability of the data in the database.
- The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table.

•**NOT NULL Constraint** – Ensures that a column cannot have NULL value.

```
CREATE TABLE STUDENT(
ROLL_NO INT NOT NULL,
STU_NAME VARCHAR (35) NOT NULL,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (235),
PRIMARY KEY (ROLL_NO)
);
```

- **DEFAULT Constraint** – Provides a default value for a column when none is specified.

```
CREATE TABLE STUDENT(
ROLL_NO    INT   NOT NULL,
STU_NAME VARCHAR (35) NOT NULL,
STU_AGE INT NOT NULL,
EXAM_FEE INT   DEFAULT 10000,
STU_ADDRESS VARCHAR (35) ,
PRIMARY KEY (ROLL_NO)
);
```

- **UNIQUE Constraint** – Ensures that all values in a column are different.

```
CREATE TABLE STUDENT(
ROLL_NO INT NOT NULL,
STU_NAME VARCHAR (35) NOT NULL UNIQUE,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (35) UNIQUE,
PRIMARY KEY (ROLL_NO)
);
```

- **PRIMARY Key** – Uniquely identifies each row/record in a database table.

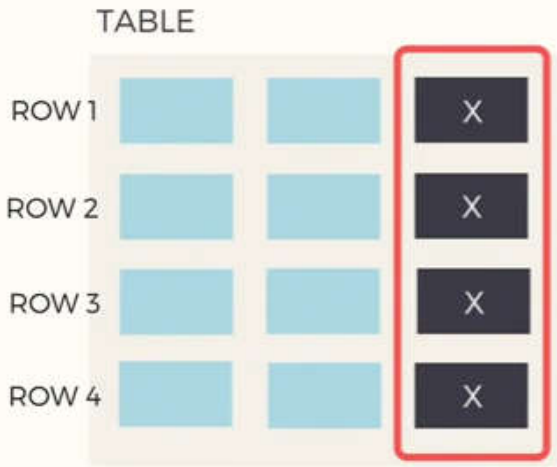
```
CREATE TABLE STUDENT(
ROLL_NO    INT   NOT NULL,
STU_NAME VARCHAR (35)   NOT NULL UNIQUE,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (35) UNIQUE,
PRIMARY KEY (ROLL_NO)
);
```

- **FOREIGN Key** – Uniquely identifies a row/record in any of the given database table.
- **CHECK Constraint** – Ensures that all the values in a column satisfies certain conditions.

```
CREATE TABLE STUDENT(
ROLL_NO    INT   NOT NULL CHECK(ROLL_NO >1000) ,
STU_NAME VARCHAR (35)   NOT NULL,
STU_AGE INT   NOT NULL,
EXAM_FEE INT DEFAULT 10000,
STU_ADDRESS VARCHAR (35) ,
PRIMARY KEY (ROLL_NO)
);
```

normalization is

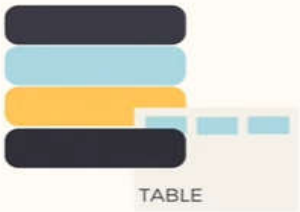
a technique of organizing the data into multiple related tables, to minimize DATA REDUNDANCY.



- Repetition of data increases the size of database.
- Other issues like:
  - Insertion Problems
  - Deletion Problems
  - Updation Problems

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337



issues due to redundancy.

1. Insertion Anomaly
2. Deletion Anomaly
3. Updation Anomaly

Deletion Anomaly

Loss of a related dataset when some other dataset is deleted.

Insertion Anomaly

To insert redundant data for every new row (of Student data in our case) is a data insertion problem or anomaly.

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337

MTRX

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337

Mr. X leaves, and Mr. Y joins as the new HOD for CSE

STUDENTS TABLE

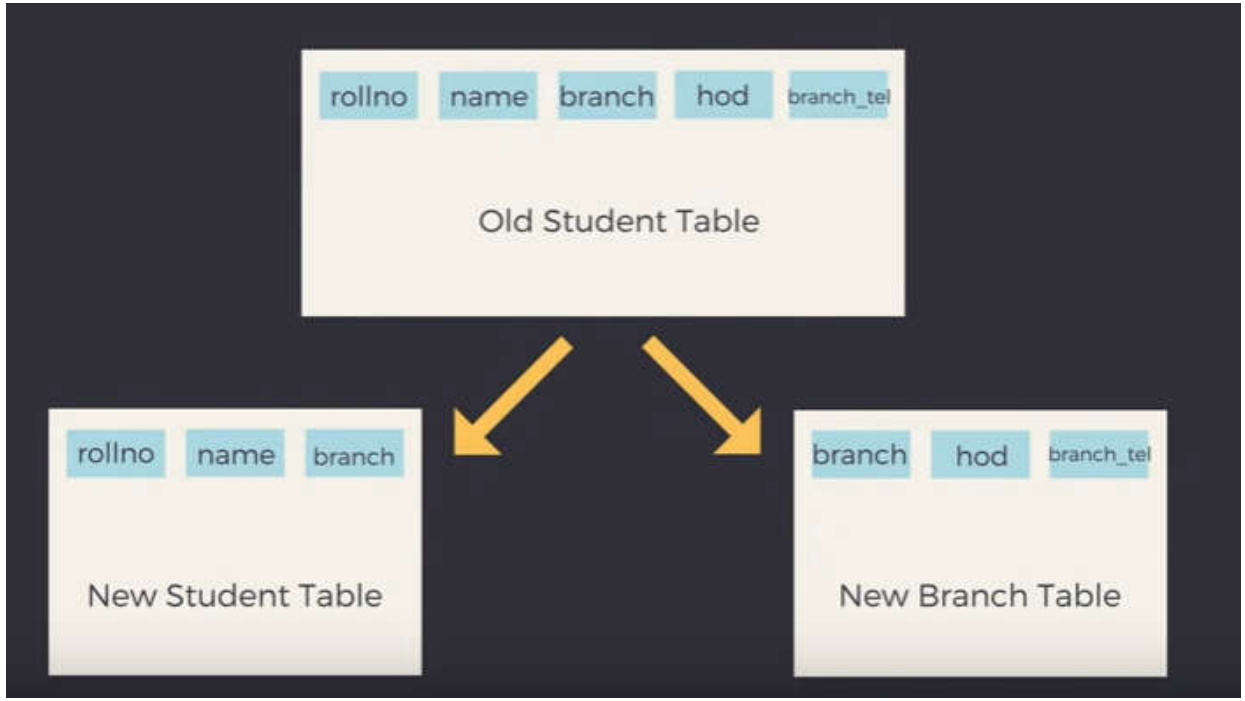
rollno	name	branch	hod	office_tel
1	Akon	CSE	~~Mr. X~~ Mr. Y	53337
2	Bkon	CSE	~~Mr. X~~ Mr. Y	53337
3	Ckon	CSE	~~Mr. X~~ Mr. Y	53337
4	Dkon	CSE	~~Mr. X~~ Mr. Y	53337

STUDENTS TABLE

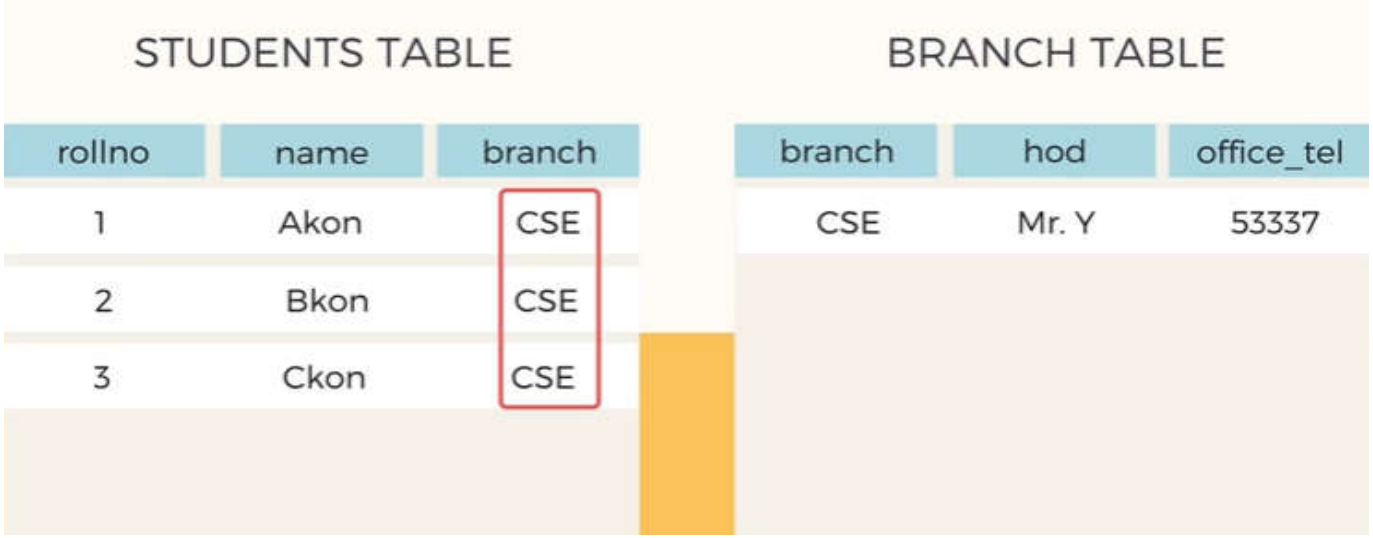
rollno	name	branch	hod	office_tel
1	Akon	CSE	~~Mr. X~~ Mr. Y	53337
2	Bkon	CSE	~~Mr. X~~ Mr. Y	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	~~Mr. X~~ Mr. Y	53337

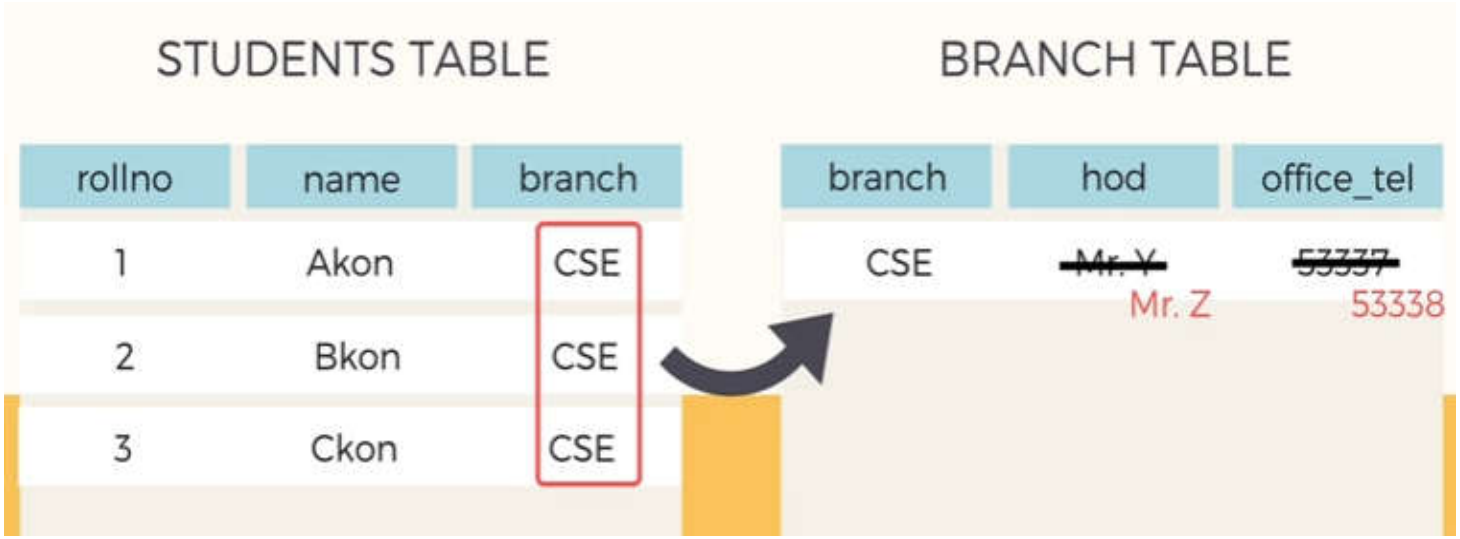
Data Redundancy

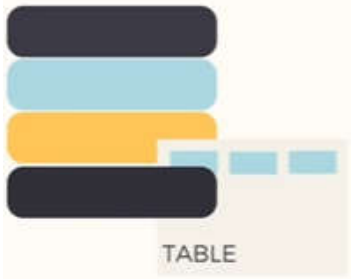
- Repetition of data hence needs extra space.



How Normalization will solve all these problems?


STUDENTS TABLE
rollno
name
branch
1
Akon
CSE
2
Bkon
CSE
3
Ckon
CSE
BRANCH TABLE
branch
hod
office_tel
CSE
Mr. Y
53337


STUDENTS TABLE
rollno
name
branch
1
Akon
CSE
2
Bkon
CSE
3
Ckon
CSE
BRANCH TABLE
branch
hod
office_tel
CSE
Mr. Y
53337
Mr. Z
53338



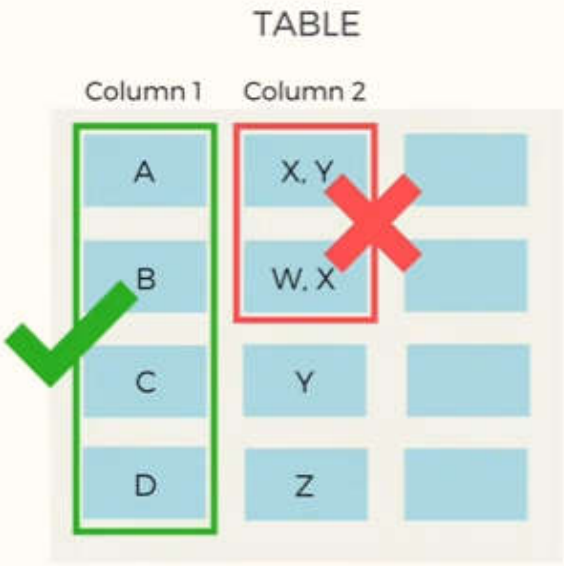
Types of Normalization

1. 1st Normal Form
2. 2nd Normal Form
3. 3rd Normal Form
4. BCNF

1st Normal Form

Every Table in your Database should at least follow the 1st Normal Form, always.

or Stop using Database!



RULE 1

- Each Column should contain atomic values.
- Entries like X, Y and W, X violate this rule.

TABLE

DOB	Name	
26-10-89	A	
13-2-92	SK	
16-11-65	SA	
R	8-9-86	

RULE 2

- A Column should contain values that are of the same type.
- Do not inter-mix different types of values in any column.

TABLE

DOB	Name	Name
26-10-89	A	A
13-2-92	S	K
16-11-65	S	A
8-9-86	R	A

RULE 3

- Each column should have unique name.
- Same names leads to confusion at the time of data retrieval

TABLE

Roll_no	F_Name	L_Name
3	A	A
4	S	K
1	S	A
2	R	A

RULE 4

- Order in which data is saved doesn't matter.
- Using SQL query, you can easily fetch data in any order from a table.

STUDENTS TABLE

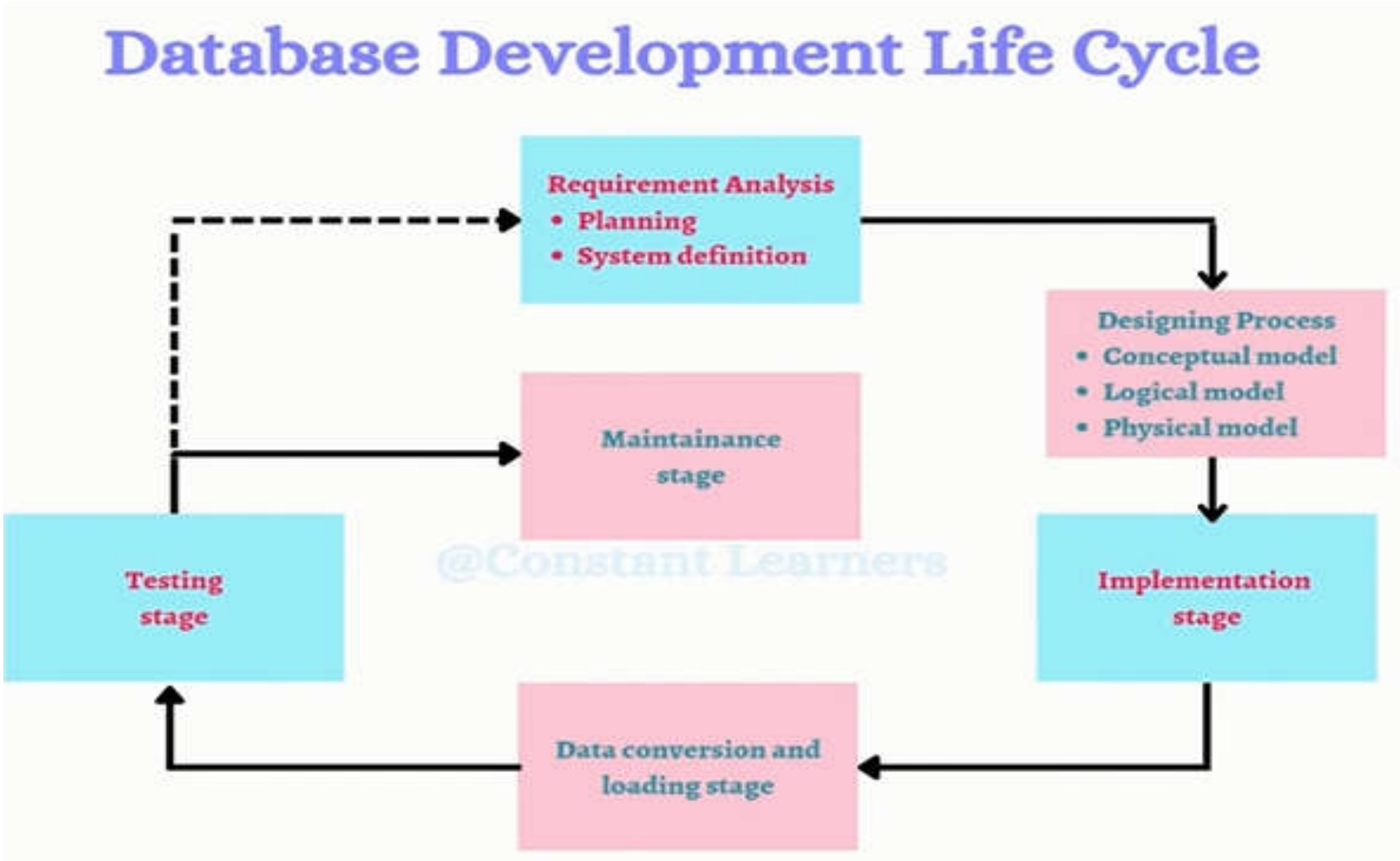
rollno	name	subject
101	Akon	OS, CN
103	Ckon	JAVA
102	Bkon	C, C++

Violation of 1 NF

STUDENTS TABLE

rollno	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	JAVA
102	Bkon	C
102	Bkon	C++

Design Process (Components of DBMS and Overall Structure of DBMS)



1. Requirement analysis

- Identifying and understanding the fundamental requirements
- Develop a comprehensive plan for the DDLC
- Organization's functional requirements

a. Planning

- mapping out the entire DDLC
- organization's requirements
- strong foundation

b. System definition

- build upon the planning stage
- boundaries and scope
- set clear parameters and limitations
- purpose and functionality

2. Designing stage

- Creating a well-structured database
- Different models - Visualize and communicate the database design features

a. Conceptual model
b. Logical model
c. Physical model

a. Conceptual model

- Users view of data
- Organization wants to look at data
- A conceptual schema, highlighting the database entities
- Abstract or superficial view of data
- Easy to create as well as understand

b. Logical model

- Expanding conceptual model
- Without considering specific database management system (DBMS) or physical implementation detail
- Understanding the data needs
- Data elements relate to each other
- What information should be stored
- How the entities relate to one another

Key features of Logical model

- Identify and define the entity relationships
- Specify the attributes that describe the characteristics or properties of each entity
- Assign a primary key
- Establish relationships between entities through foreign keys
- Apply normalization techniques to eliminate redundancy and ensure data integrity

c. Physical model

- Implementing the logical database design
- Software systems, DBMS, the hardware resources and physical implementation aspects
- Theoretical concept into a practical implementation

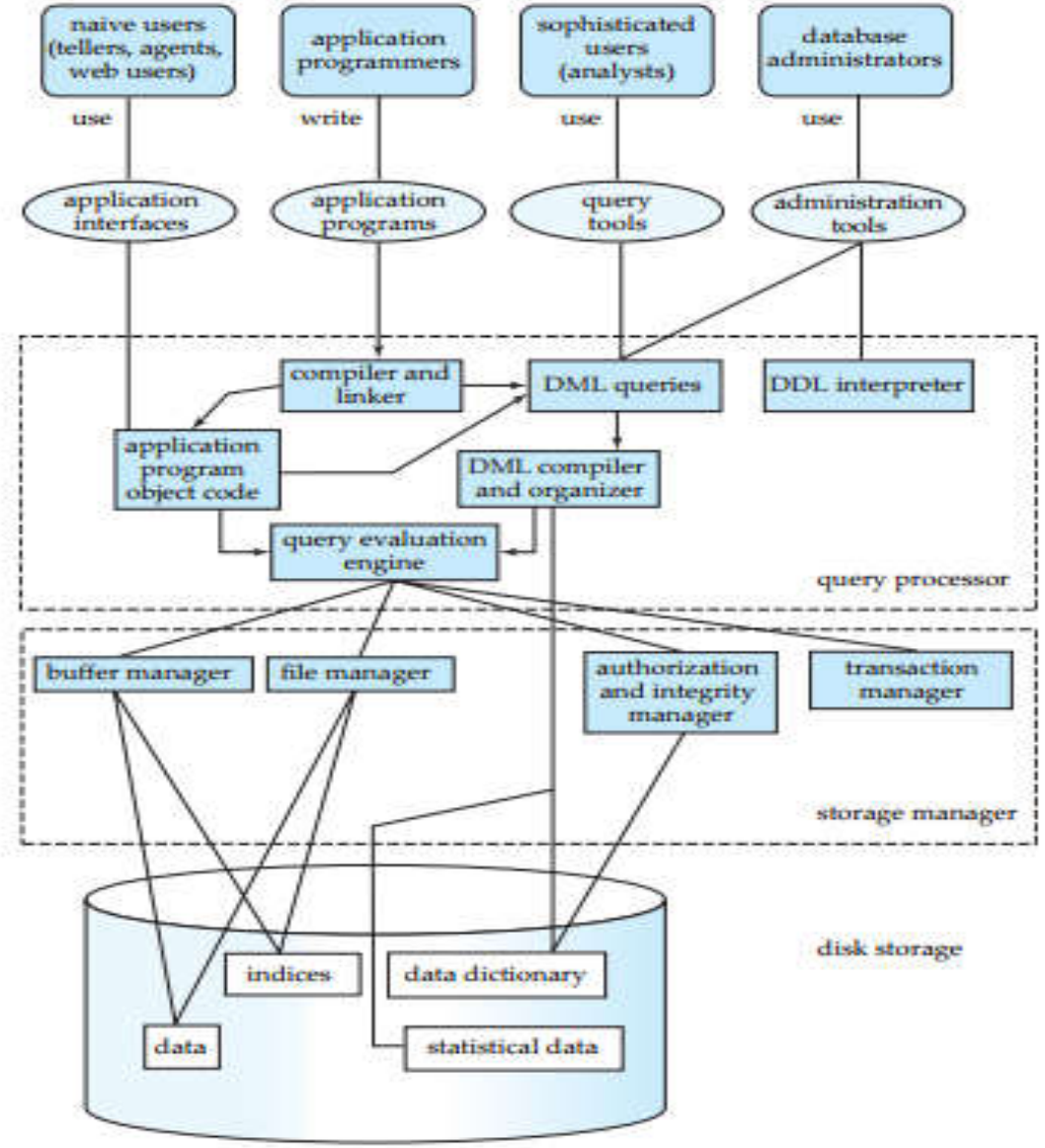
Key features of Physical model

- Specification of columns and tables
- Define the structure of tables, including column names, data types, constraints, etc
- Consider the physical and practical aspects of the database design
- Physical models can be different with respect to different RDBMS, including differences in data types

3. Implementation

- Putting the designed database into action and ensuring that it meets our requirements
- Integration testing - using different data sets
- Convert the data into a format that the computer can understand and process efficiently
- Data manipulation

Design Process (Components of DBMS and Overall Structure of DBMS)



Disk storage

- Data files: actual DB
- Data dictionary: meta data
- Indices: Data access to be fast in accordance with the query processed
- Statistical data: storing stats about the data, to take appropriate decision

Database Users and Administrators

- **Naive users** are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

- **Application programmers** are computer professionals who write application programs. **Rapid Application Development (RAD)** tools are tools that enable an application programmer to construct forms and reports with minimal programming effort.

- **Sophisticated users** interact with the system using a database query language. These queries submitted to Query Processor.

- **Database Administrator:** A person who has such central control over the system (to access data and program that access those data) **or** Handles physical and logical level of database is called a **database administrator (DBA)**.

- **Functions of DBA:**
- **Storage structure and access-method definition**
- **Schema and physical-organization modification**
- **Granting of authorization for data access**
- **Routine maintenance**

The functional components of a database system can be broadly divided into the **Query Processor & Storage Manager.**

- **DDL interpreter**: Interprets DDL statements and records the definitions in the data dictionary(set of table).
- **DML compiler**: Translates DML statements in a query language into low-level instructions that the query evaluation engine understands.
- **Query evaluation engine: E**xecutes low-level instructions generated by the DML compiler.

- Storage manager: role based access control for data retrieval, storage
- Integrity manager: check integrity constraints enforced on data(eg:balance>0)
- Transaction manager: controls concurrent access despite system failure.
- File manager: manages file space, data structure.
- Buffer manager: data transfer from secondary to main memory, decides the data to be in the cache memory to speed up the operation, handles the size of data.

References used

Text Books:

1. Silberschatz A., Korth H., Sudarshan S., "Database System Concepts", McGraw Hill Publishers
2. Connally T, Begg C., "Database Systems", Pearson Education
3. R. P. Mahapatra and Govind Verma, “Database Management Systems”, Khanna Publishing House

Reference Books:

1. Raghurama Krishan, “Database Management Systems”, McGrawHill
2. S.K.Singh, “Database Systems : Concepts, Design and Application”, Pearson, Education
3. Pramod J. Sadalage and Martin Fowler, “NoSQL Distilled”, Addison Wesley
4. Kristina Chodorow, Michael Dirolf, “MangoDB: The Definitive Guide”, O’Reilly Publications

e-Resources: https://nptel.ac.in/courses/106/105/106105175/ ,notes and video sessions of the experts of the respective fields.

Thank you !!!