Unit 1

Introduction to DBMS

Subject Importance

- A **database management system** is **important** because it manages **data** efficiently and allows users to perform multiple tasks with ease.
- **Data** can be categorized and structured to suit the needs of the company or organization.
- **Data** is entered into the **system** and accessed on a routine basis by assigned users.

Importance

- **Database** helps in keeping the files in a systematic manner.
- It helps in managing large amount of information in small time.
- A **database** is an organized collection of data.
- A relational **database**, more restrictively, is a collection of schemas, tables, queries, reports, views, and other elements manages.

Top 10 Enterprise Database Systems of 2019

- Oracle **Database**.
- Oracle began its journey in 1979 as the first commercially available relational **database** management system (RDBMS). ...
- Microsoft SQL Server. ...
- IBM DB2. ...
- SAP Sybase ASE. ...
- PostgreSQL. ...
- MariaDB Enterprise. ...
- MySQL. ...
- Teradata.
- IBM Informix

Jobs in world

- **Database analysts and data administrators .**
- **Oracle Database Administrator.**
- **Database Administrator.**
- **Data Science Engineer .**
- **MongoDB Developer.**
- **Microsoft SQL Server Database Administration.**
- **Oracle Applications DBA.**
- **Database Administrator - IT Security Design & Implementation.**
- **PL-SQL Developer - SQL/Data Migration**

Introduction

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases can be very large.
- Databases touch all aspects of our lives

Introduction

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems

Advantages of a DBMS over file-processing Systems

- Explain advantage of DBMS (System) over conventional file processing system.

What are the benefits of database management system?

- **Advantages of Database Management System**
- Reducing Data Redundancy.
- The file based data management systems contained multiple files that were stored in many different locations in a system or even across multiple systems. ...
- Sharing of Data. ...
- Data Integrity. ...
- Data Security. ...
- Privacy. ...
- Backup and Recovery. ...
- Data Consistency.

Drawbacks of File system

- **Data redundancy(Duplication)** : Data redundancy refers to the duplication of data, lets say we are managing the data of a college where a student is enrolled for two courses,

the same student details in such case will be stored twice, which will take more storage than needed.

- Data redundancy often **leads to higher storage costs and poor access time**.

Drawbacks of File system .. Continue

- **Data inconsistency:**
- Data redundancy leads to data inconsistency, lets take the same example that we have taken above,
- a student is enrolled for two courses and we have student address stored twice, now lets say student requests to change his address, if the address is changed at one place and not on all the records then this can lead to data inconsistency.

Drawbacks of File system .. Continue

- **Difficulty in accessing Data:**
- In Classical file organization the data is stored in the files.
- Whenever data has to be retrieved as per the requirements then a new application program has to be written.
- This is tedious process.

Drawbacks of File system .. Continue

- **Data Isolation:**
- Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

Drawbacks of File system .. Continue

- **Integrity Problem:**
- The data values stored in the database must satisfy certain types of consistency constraints.
- For example, the balance of a bank account **may never fall below a prescribed amount(Threshold Value)**. These constraints are enforced in the system by adding appropriate code in the various application programs.
- However, 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.

Drawbacks of File system .. Continue

- **Atomicity issues:**
- Atomicity of a transaction refers to “All or nothing”, which means either all the operations in a transaction executes or none.

- For example: Lets say Steve transfers 100$ to Negan’s account. This transaction consists multiple operations such as debit 100$ from Steve’s account, credit 100$ to Negan’s account.

- like any other device, a computer system can fail lets say it fails after first operation then in that case Steve’s account would have been debited by 100$ but the amount was not credited to Negan’s account, in such case the rollback of operation should occur to maintain the atomicity of transaction. It is **difficult to achieve atomicity in file processing systems**.

Drawbacks of File system .. Continue

- **Data concurrency:**
- Concurrent access to data means **more than one user is accessing the same data at the same time**.
- Anomalies occur when changes made by one user gets lost because of changes made by other user.
- File system does not provide any procedure to stop anomalies. Whereas DBMS provides a locking system to stop anomalies to occur.

Drawbacks of File system .. Continue

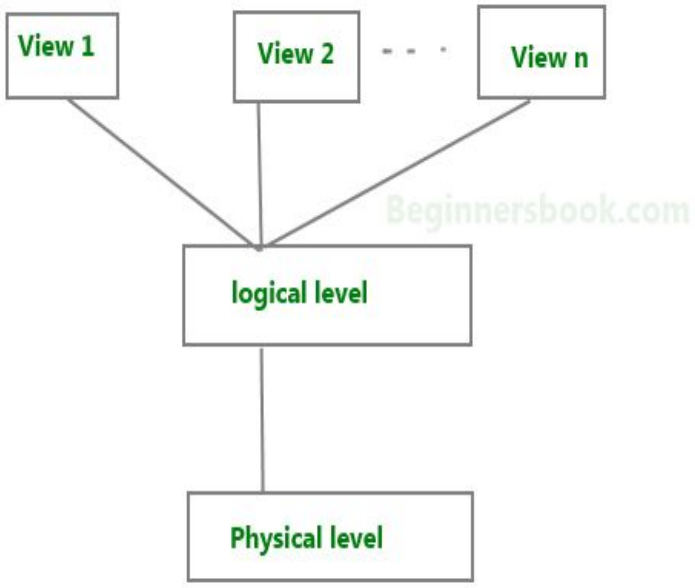
- **Data Security:**
- Data should be secured from unauthorized access,
- for example a student in a college should not be able to see the payroll details of the teachers, such kind of security constraints are difficult to apply in file processing systems.

Summary Drawbacks of File system over the DBMS

- Data Redundancy
- Data Inconsistency
- Difficulty in accessing the data
- Data Isolation
- Integration Problem
- Atomicity Issues
- Data Concurrency
- Data Security

Data Abstraction

- Database systems are made-up of complex data structures.
- To ease the user interaction with database, the developers hide internal irrelevant details from users.
- This process of hiding irrelevant details from user is called data abstraction.



Three Levels of data abstraction

Levels of Abstraction

- **Physical level**:
- This is the lowest level of data abstraction.
- **It describes how data is actually stored in database.**
- You can get the complex data structure details at this level.
- **Example:**
- Let's say we are storing customer information in a customer table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.

Levels of Abstraction…Continue

- **Logical level:**
- This is the middle level of 3-level data abstraction architecture.
- **It describes what data is stored in database.**
- At the logical level these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented.
- The programmers generally work at this level because they are aware of such things about database systems.

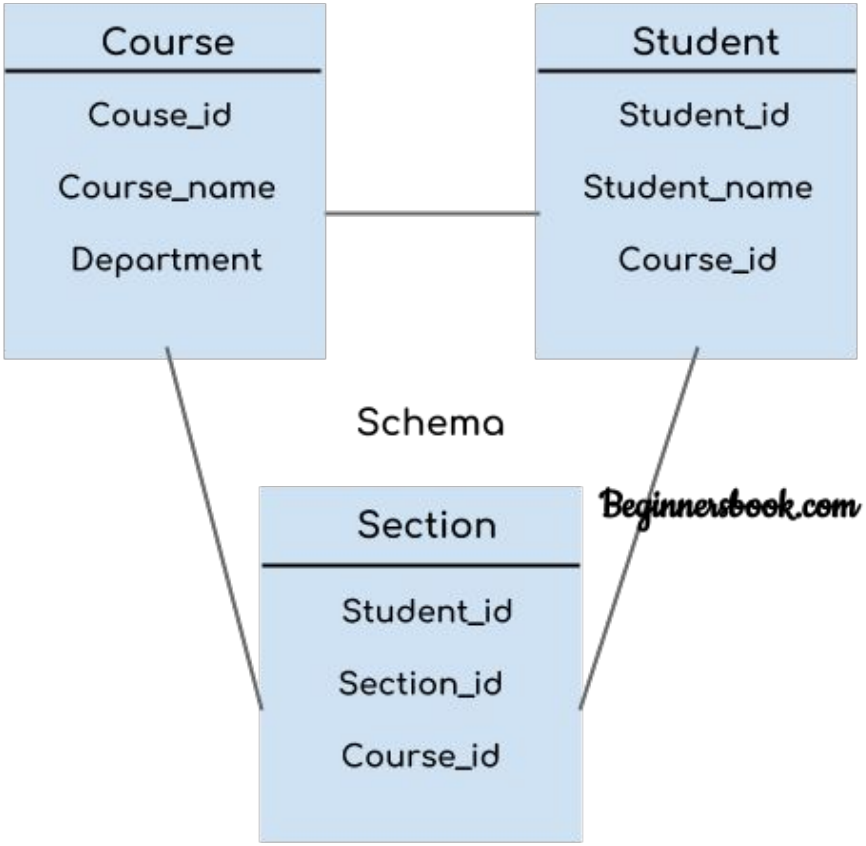
Levels of Abstraction…Continue

- **View level:**
- Application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.
- At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

Instances and Schemas

- **Definition of schema:**
- Design of a database is called the schema.
- Schema is of three types: Physical schema, logical schema and view schema.
- For example: In the following diagram, we have a schema that shows the relationship between three tables: Course, Student and Section.
- The diagram only shows the **design of the database**, it **doesn't show the data present in those tables**. Schema is only a **structural view(design) of a database** as shown in the diagram below.

Instances and Schemas...Continue



Instances and Schemas…Continue

- The design of a database at physical level is called **physical schema**, how the data stored in blocks of storage is described at this level.
- Design of database at logical level is called **logical schema**, programmers and database administrators work at this level, at this level data can be described as certain types of data records gets stored in data structures.
- Design of database at view level is called **view schema**. This generally describes end user interaction with database systems.

Instances and Schemas…Continue

Definition of instance:

- The data stored in database at a particular moment of time is called instance of database.
- Database schema defines the variable declarations in tables that belong to a particular database;
- the value of these variables at a moment of time is called the instance of that database.

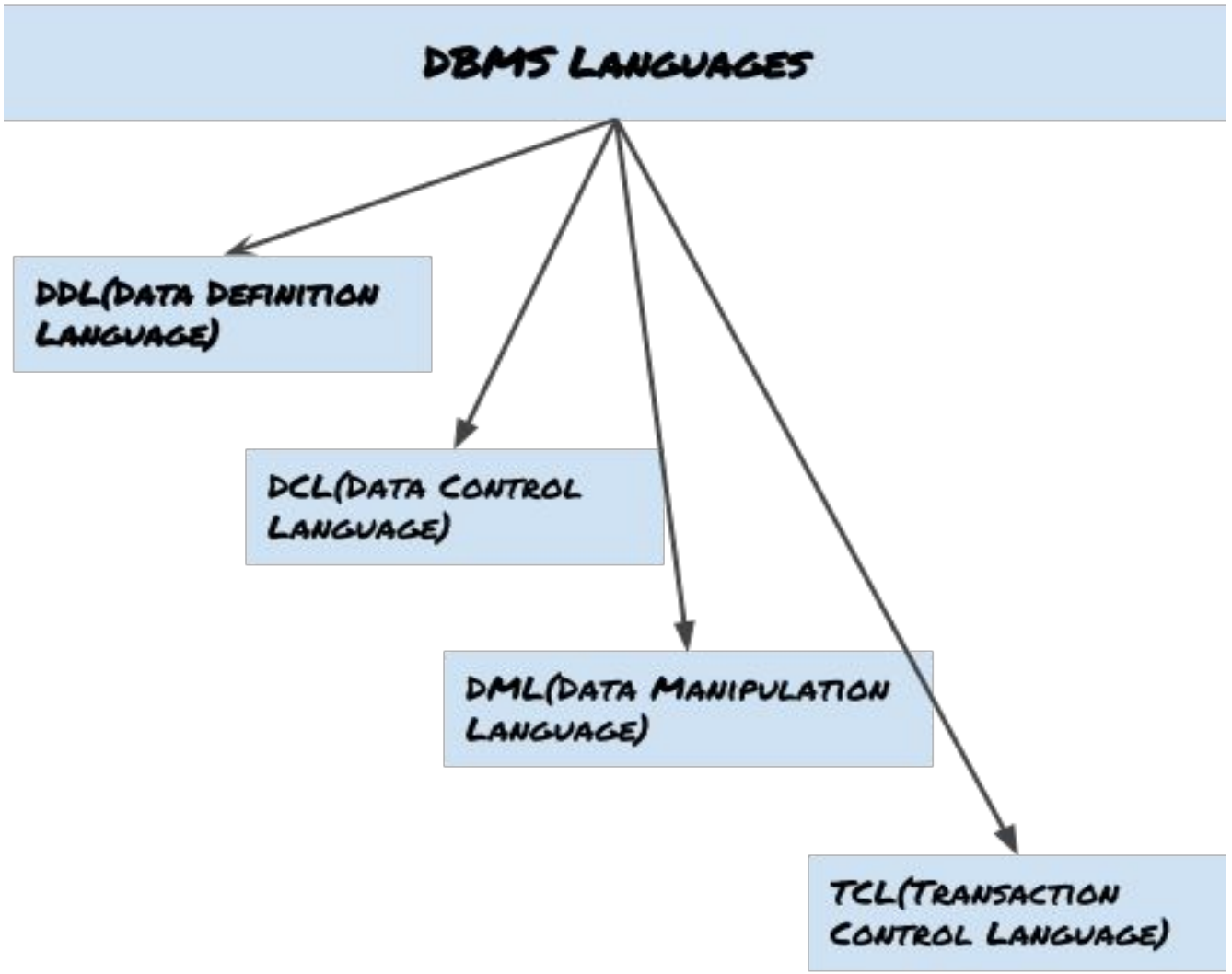
Instances and Schemas…Continue

- For example, lets say we have a single table student in the database, today the table has 100 records, so today the **instance of the database has 100 records**. Lets say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table. In short, at a particular moment the data stored in database is called the instance, that changes over time when we add or delete data from the database.

Database Languages

- Database languages are used to read, update and store data in a database. There are several such languages that can be used for this purpose; one of them is SQL (Structured Query Language).

Database Languages…Continue



Database Languages…Continue

- **Data Definition Language (DDL)**
- DDL is used for specifying the database schema. It is used for creating tables, schema, indexes, constraints etc. in database. Lets see the operations that we can perform on database using DDL:
- To create the database instance – **CREATE**
- To alter the structure of database – **ALTER**
- To drop database instances – **DROP**
- To delete tables in a database instance – **TRUNCATE**
- To rename database instances – **RENAME**
- To drop objects from database such as tables – **DROP**
- **Note-**All of these commands either defines or update the database schema that's why they come under Data Definition language.

Database Languages…Continue

- **Data Manipulation Language (DML)**
- DML is used for accessing and manipulating data in a database. The following operations on database comes under DML:
- To read records from table(s) – **SELECT**
- To insert record(s) into the table(s) – **INSERT**
- Update the data in table(s) – **UPDATE**
- Delete all the records from the table – **DELETE**

Database Languages…Continue

- **Data Control language (DCL)**
- DCL is used for granting and revoking user access on a database –
- To grant access to user – GRANT
- To revoke or cancel access from user – REVOKE
- **Note-In practical data definition language, data manipulation language and data control languages are not separate language, rather they are the parts of a single database language such as SQL.**

Database Languages…Continue

- **Transaction Control Language(TCL)**
- The changes in the database that we made using DML commands are either performed or rollbacked using TCL.
- To persist the changes made by DML commands in database – COMMIT
- To rollback the changes made to the database – ROLLBACK

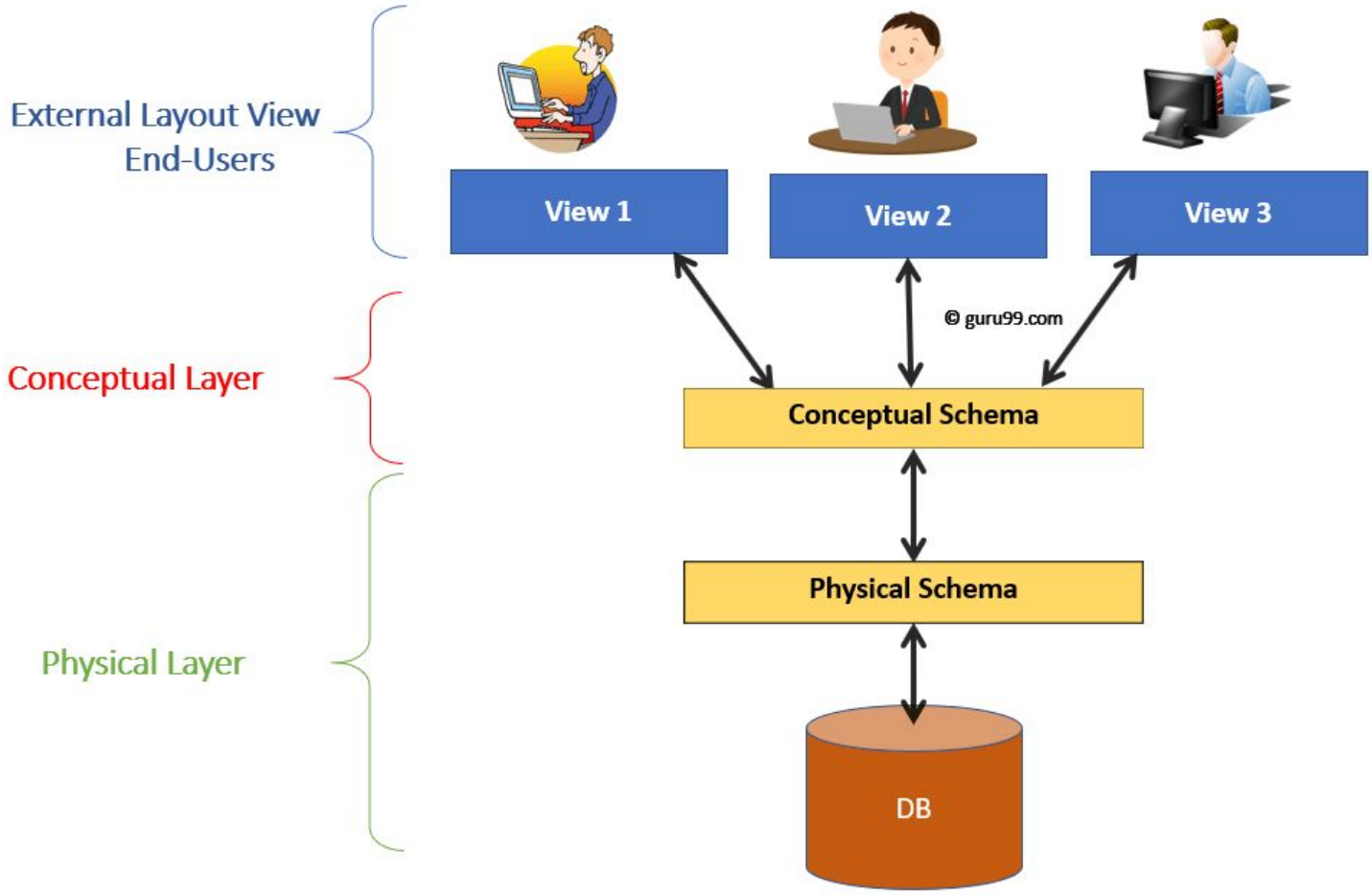
What is Data Independence of DBMS?

- Data Independence is defined as a property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level.
- Data independence helps you to keep data separated from all programs that make use of it.

Data Independence…Continue

- **Types of Data Independence**
- In DBMS there are two types of data independence
- Physical data independence
- Logical data independence.
- **Levels of Database**
- The database has 3 levels as shown in the diagram below
- Physical/Internal
- Conceptual
- External

Data Independence…Continue



Data Independence…Continue

- **Physical Data Independence**
- Physical data independence helps you to separate conceptual levels from the internal/physical levels.
- It allows you to provide a logical description of the database without the need to specify physical structures.
- Compared to Logical Independence, it is easy to achieve physical data independence.
- With Physical independence, you can easily change the physical storage structures or devices without an effect on the conceptual schema.
- Any change done would be absorbed by the mapping between the conceptual and internal levels.

Data Independence…Continue

- **Examples of changes under Physical Data Independence**
- Due to Physical independence, any of the below change will not affect the conceptual layer.
- Using a new storage device like Hard Drive or Magnetic Tapes
- Modifying the file organization technique in the Database
- Switching to different data structures.
- Changing the access method.
- Modifying indexes.
- Changes to compression techniques or hashing algorithms.
- Change of Location of Database from say C drive to D Drive

Data Independence…Continue

- **Logical Data Independence**
- Logical Data Independence is the ability to change the conceptual scheme **without changing External views**
- **External API or programs**
- Any change made will be absorbed by the mapping between external and conceptual levels.
- When compared to Physical Data independence, it is challenging to achieve logical data independence.

Data Independence…Continue

- **Examples of changes under Logical Data Independence**
- Due to Logical independence, any of the below change will not affect the external layer.
- Add/Modify/Delete a **new attribute**, entity or relationship is possible without a **rewrite of existing application programs**
- **Merging two records into one**
- Breaking an existing record into two or more records

Difference between Physical and Logical Data Independence

Logica Data Independence	Physical Data Independence
Logical Data Independence is mainly concerned with the structure or changing the data definition.	Mainly concerned with the storage of the data.
It is difficult as the retrieving of data is mainly dependent on the logical structure of data.	It is easy to retrieve.
Compared to Logic Physical independence it is difficult to achieve logical data independence.	Compared to Logical Independence it is easy to achieve physical data independence.
You need to make changes in the Application program if new fields are added or deleted from the database.	A change in the physical level usually does not need change at the Application program level.
Modification at the logical levels is significant whenever the logical structures of the database are changed.	Modifications made at the internal levels may or may not be needed to improve the performance of the structure.
Concerned with conceptual schema	Concerned with internal schema
Example: Add/Modify/Delete a new attribute	Example: change in compression techniques, hashing algorithms, storage devices, etc

Data models

- **Data Model** is a logical structure of Database. It describes the design of database to reflect entities, attributes, relationship among data, constrains etc.
- **Types of Data Models- 3 Types of Data Model**
- **A>Object based logical Models** – Describe data at the logical and view levels.
- **1.E-R Model**
- **2.Object oriented Model**

Data models Continue…

- **B>Record based logical Models** – Like Object based model, they also describe data at the logical and view levels. These models specify logical structure of database with **records, fields and attributes.**
- **1.Relational Model**-This model is simple and it has all the properties and capabilities required to process data with storage efficiency.
- **2.Hierarchical Model**-A **hierarchical** database model is a data model in which the data are organized into a **tree-like structure**. ... The **hierarchical** database model mandates that each child record has only one parent, whereas each parent record can have one or more child records
- ***3.Network Model*** – Network Model is same as hierarchical model except that it has **graph-like structure** rather than a tree-based structure. Unlike hierarchical model, this model allows each record to have more than one parent record.
- **C>Physical Data Model-** These model are used to describe data at lowest level

Data models Continue…

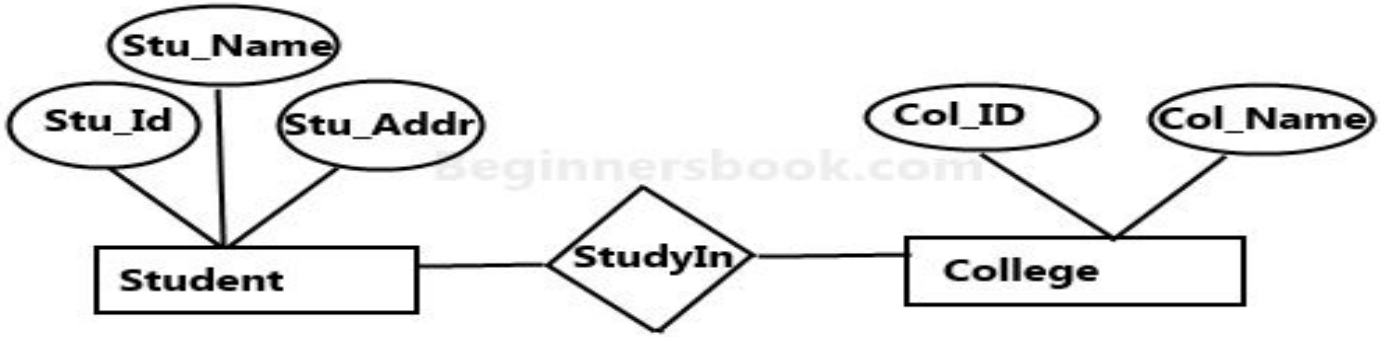
- 1.**Entity–relationship model (ER model)**
- ER Model describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**.
- An ER model is a design or blueprint of a database that can later be implemented as a database.
- The main components of E-R model are: entity set and relationship set.

- **What is an Entity Relationship Diagram (ER Diagram)?**
- An ER diagram shows the relationship among entity sets.
- An entity set is a group of similar entities and these entities can have attributes.
- In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.
- Lets have a look at a simple ER diagram to understand this concept.

Data models Continue…

A simple ER Diagram:

In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name



Sample E-R Diagram

Components of the ER Diagram

- This model is based on three basic concepts:
- Entities
- Attributes
- Relationships

- For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.

WHAT IS ENTITY?

- A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.
- An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.
- **Examples of entities:**
- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

Example of Entities:

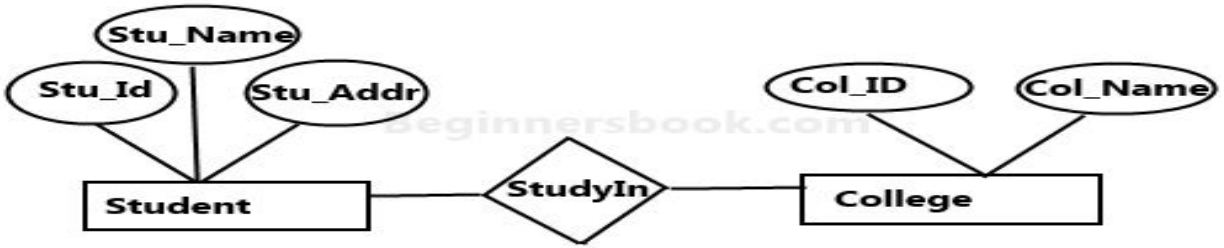
A university may have some departments. All these departments employ various lecturers and offer several programs.

Some courses make up each program. Students register in a particular program and enroll in various courses.

A lecturer from the specific department takes each course, and each lecturer teaches a various group of students.

Relationship

Relationship is nothing but an association among two or more entities. E.g., Student study In College



Sample E-R Diagram

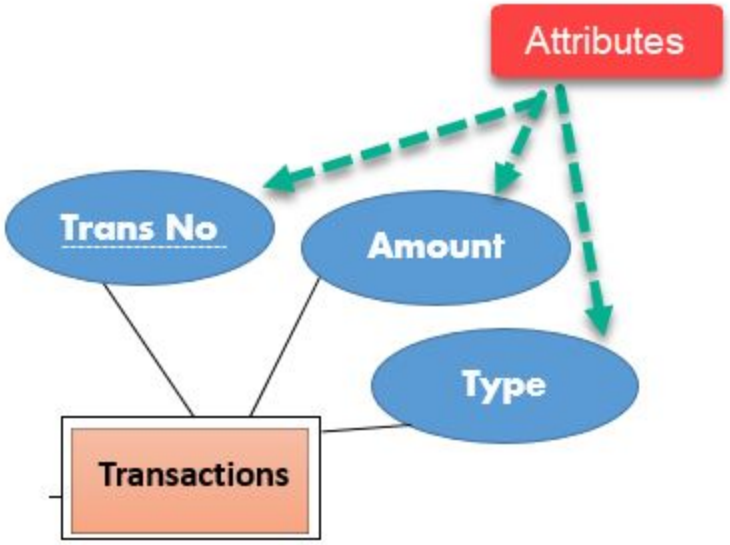
Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

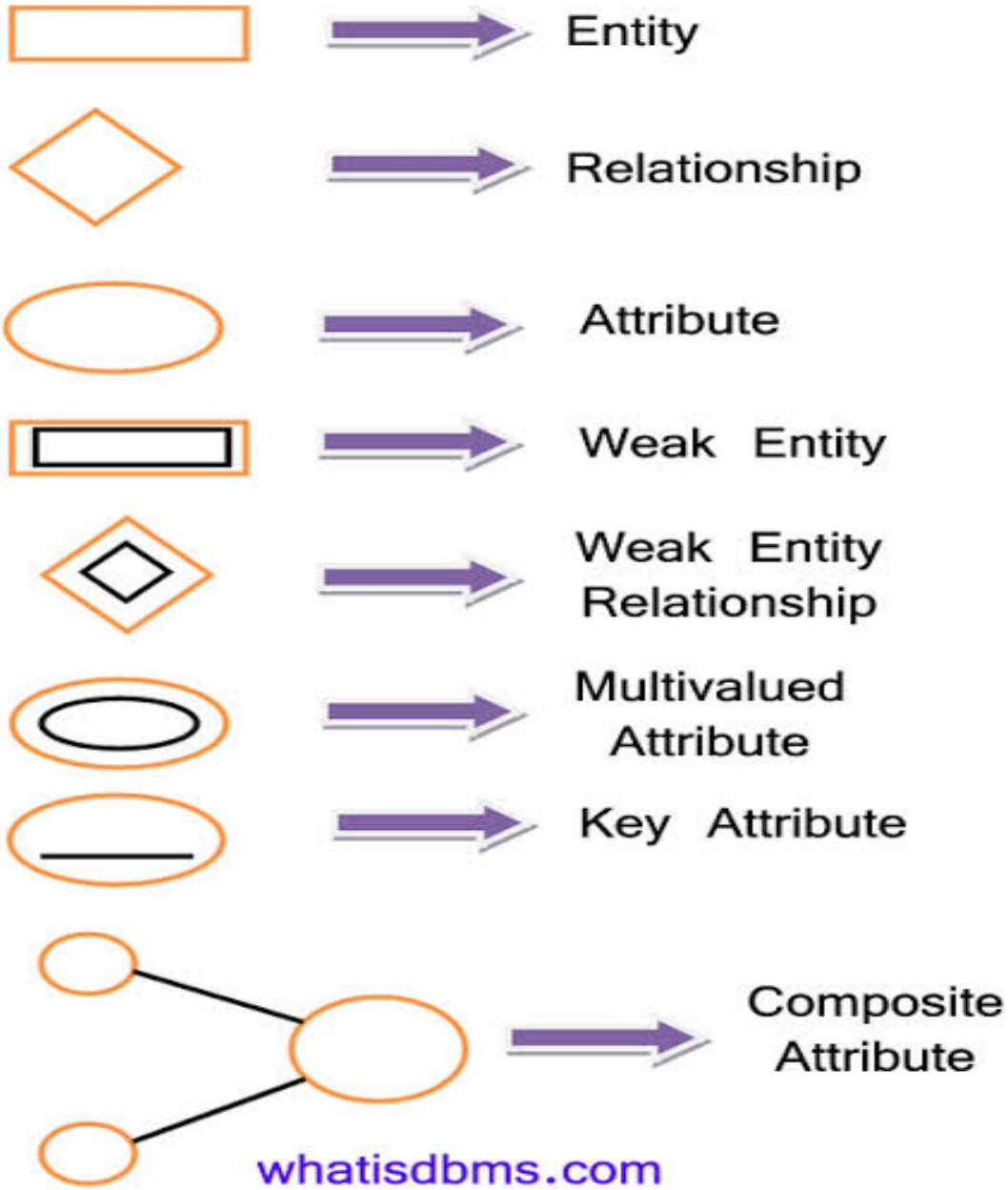
For example:

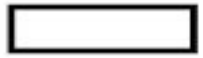
- You are attending this lecture
- I am giving the lecture
- Just like entities, we can classify relationships according to relationship-types:
- A student attends a lecture
- A lecturer is giving a lecture.

Attributes

- It is a single-valued property of either an entity-type or a relationship-type.
- For example, a lecture might have attributes: time, date, duration, place, etc.
- An attribute is represented by an Ellipse




Entity
Relationship
Attribute
Weak Entity
Weak Entity
Relationship
Multivalued
Attribute
Key Attribute
Composite
Attribute
whatisdbms.com



Represents Entity

Represents Attribute

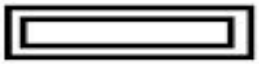
Represents Relationship

Links Attribute(s) to entity set(s) or
Entity set(s) to Relationship set(s)

Represents Multivalued Attributes

Represents Derived Attributes

Represents Total Participation of Entity

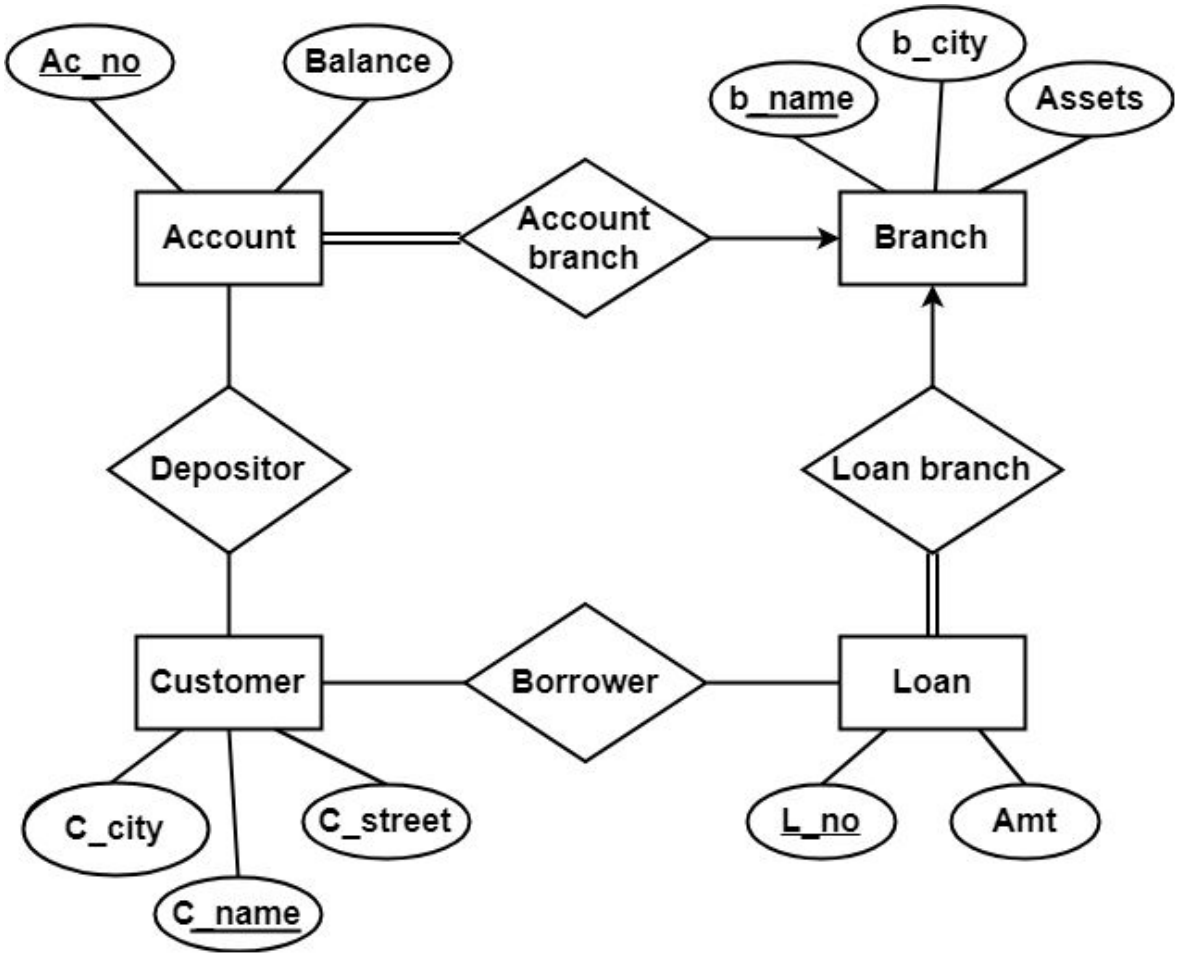


Represents Weak Entity

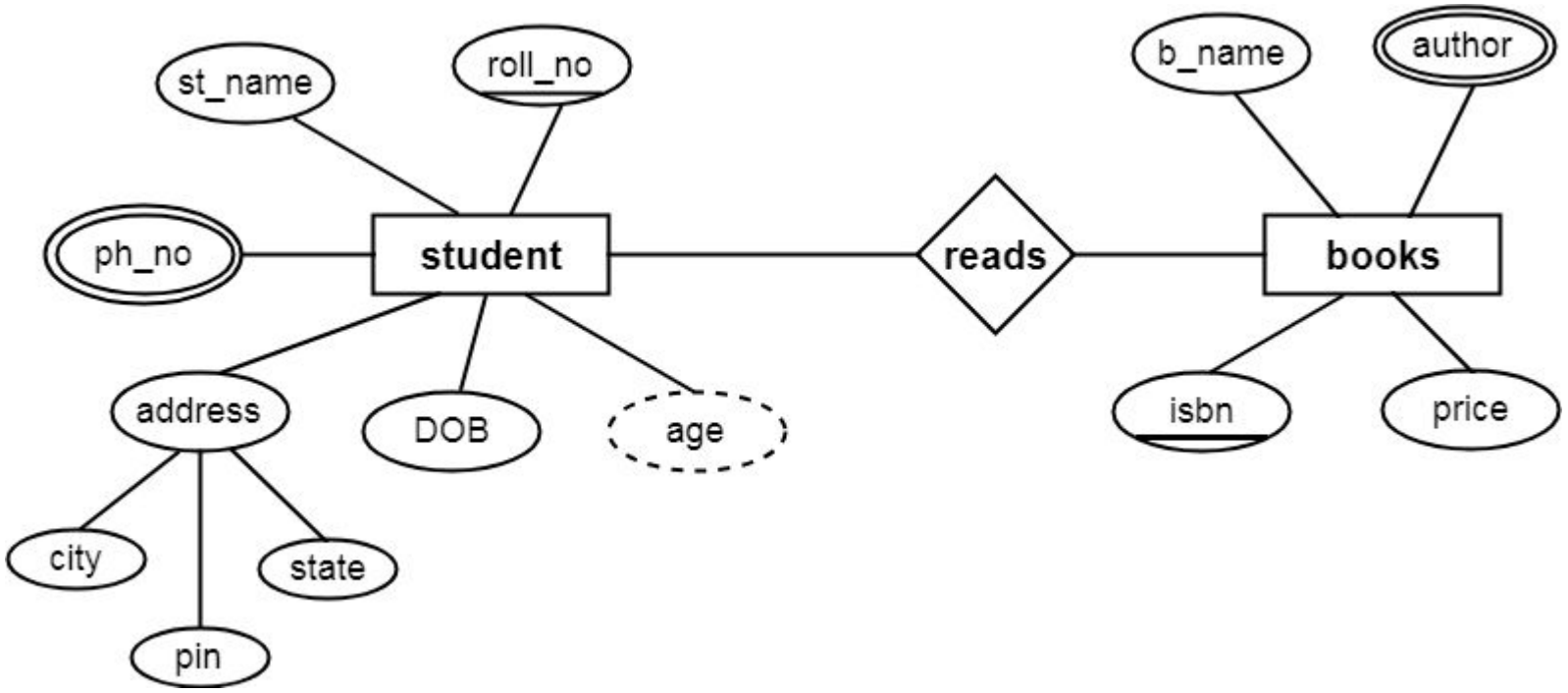
Represents Weak Relationships

Represents Composite Attributes

Represents Key Attributes / Single Valued
Attributes


Ac_no
Balance
b_city
b_name
Assets
Account
Account branch
Branch
Depositor
Loan branch
Customer
Borrower
Loan
C_city
C_street
C_name
L_no
Amt

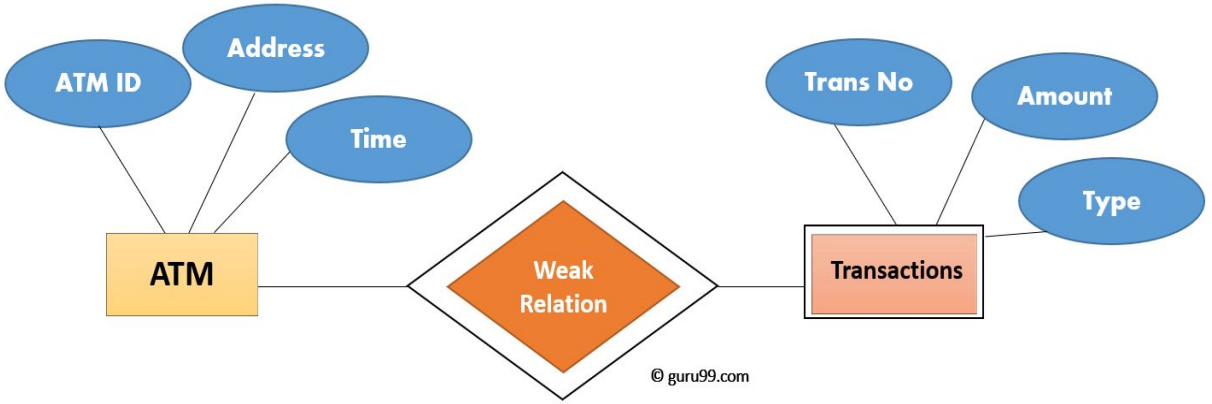
- **Solution-**
-
- Applying the rules that we have learnt, minimum 6 tables will be required-
- Account (Ac_no , Balance , b_name)
- Branch (b_name , b_city , Assets)
- Loan (L_no , Amt , b_name)
- Borrower (C_name , L_no)
- Customer (C_name , C_street , C_city)
- Depositor (C_name , Ac_no)



The double ellipse is used to define **multivalued attributes** and the dashed ellipse is used for **derived attributes**

Weak Entities

A weak entity is a type of entity which doesn't have its key attribute. **It can be identified uniquely by considering the primary key of another entity**. For that, weak entity sets need to have participation.

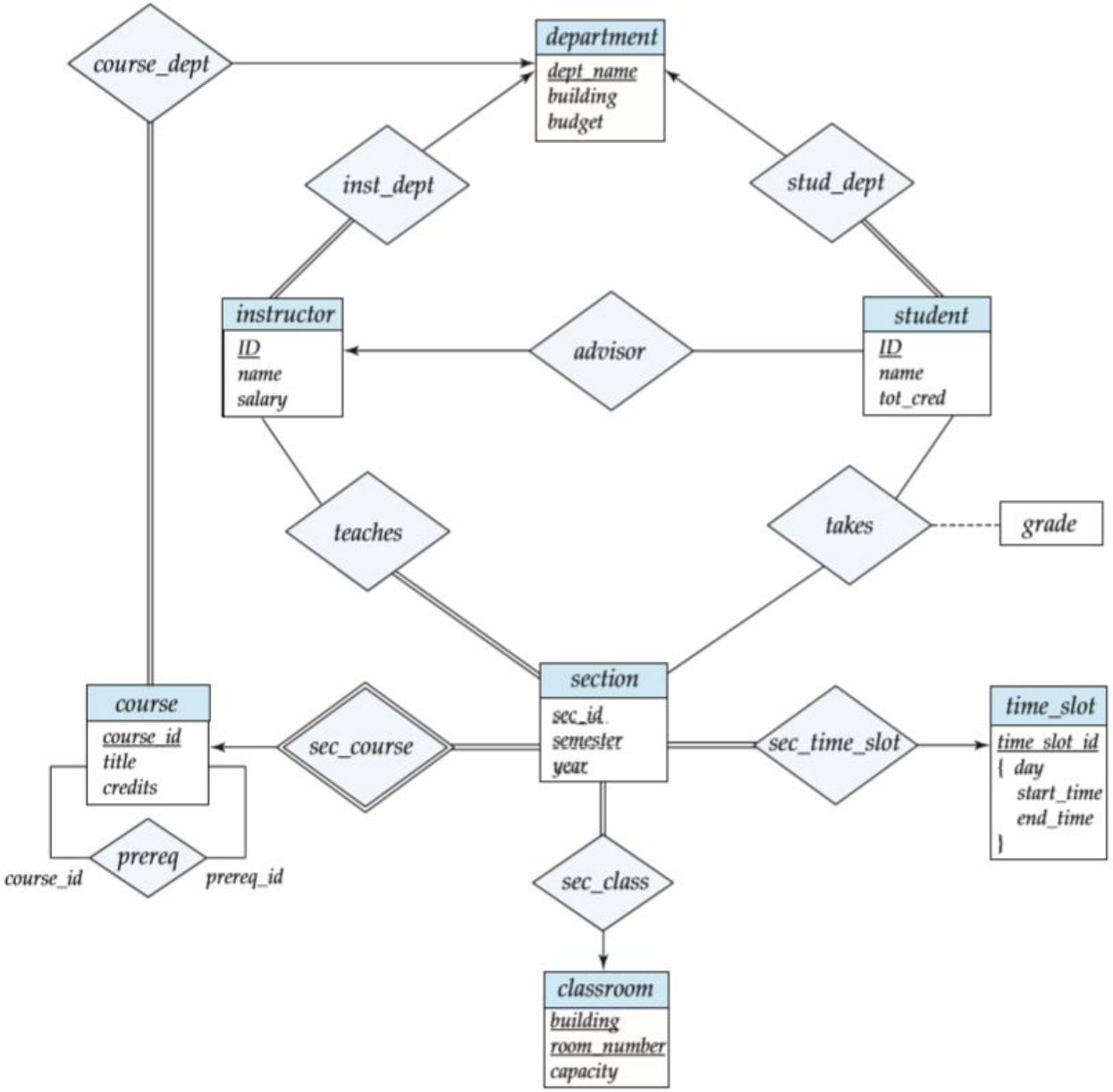


In above example, "Trans No" is a discriminator within a group of transactions in an ATM.

Let's learn more about a weak entity by comparing it with a Strong Entity

Strong Entity Set	Weak Entity Set
Strong entity set **always has a primary key**.	It **does not have** enough attributes to build a primary key.
It is represented by a **rectangle symbol**.	It is represented by a **double rectangle symbol.**
It contains a Primary key represented by the **underline symbol**.	It contains a **Partial Key** which is represented by a **dashed underline symbol**.
The member of a strong entity set is called as **dominant entity set**.	The member of a weak entity set called as a **subordinate entity set.**
Primary Key is one of its attributes which helps to **identify its member**.	In a weak entity set, it is a combination of **primary key and partial key of the strong entity set**.
In the ER diagram the relationship between two strong entity set shown by using a **diamond symbol**.	The relationship between one strong and a weak entity set shown by using the **double diamond symbol.**
The connecting line of the strong entity set with the relationship **is single**.	The line connecting the weak entity set for identifying relationship **is double**.

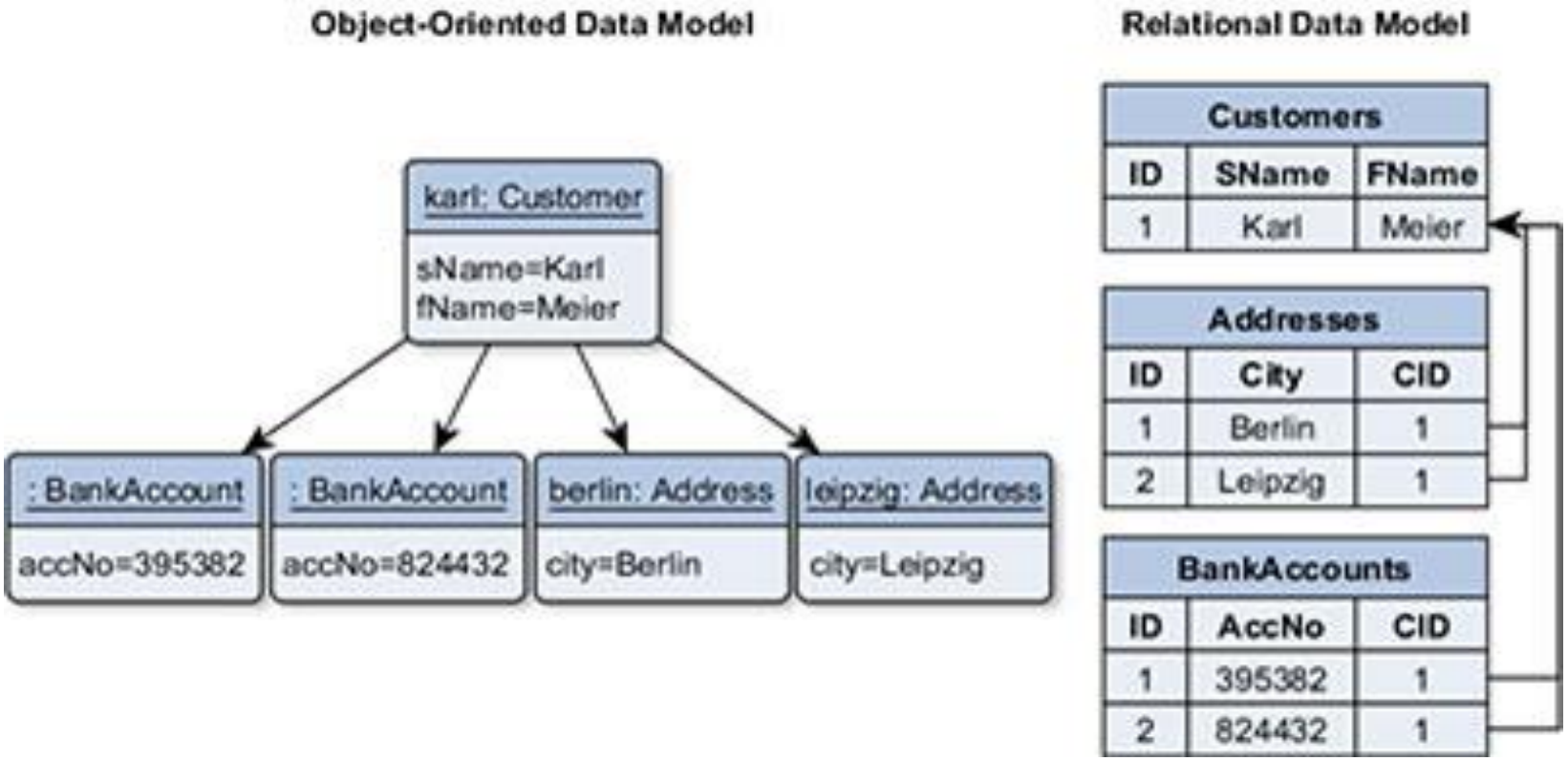
E-R diagram for the University Enterprise



Types of Attributes

Types of Attributes	Description
Simple attribute	Simple attributes **can't be divided any further**. For example, a student's contact number. It is also called an atomic value.
Composite attribute	It is possible **to break down composite attribute**. For example, a student's full name may be further divided into first name, second name, and last name.
Derived attribute	This type of attribute does not include in the physical database. However, their values are **derived from other attributes present in the database**. For example, age should not be stored directly. Instead, it should be derived from the DOB of that employee.
Multivalued attribute	Multivalued attributes can have **more than one values**. For example, a student can have more than one mobile number, email address, etc.

Object Oriented Data Model



Relational Model

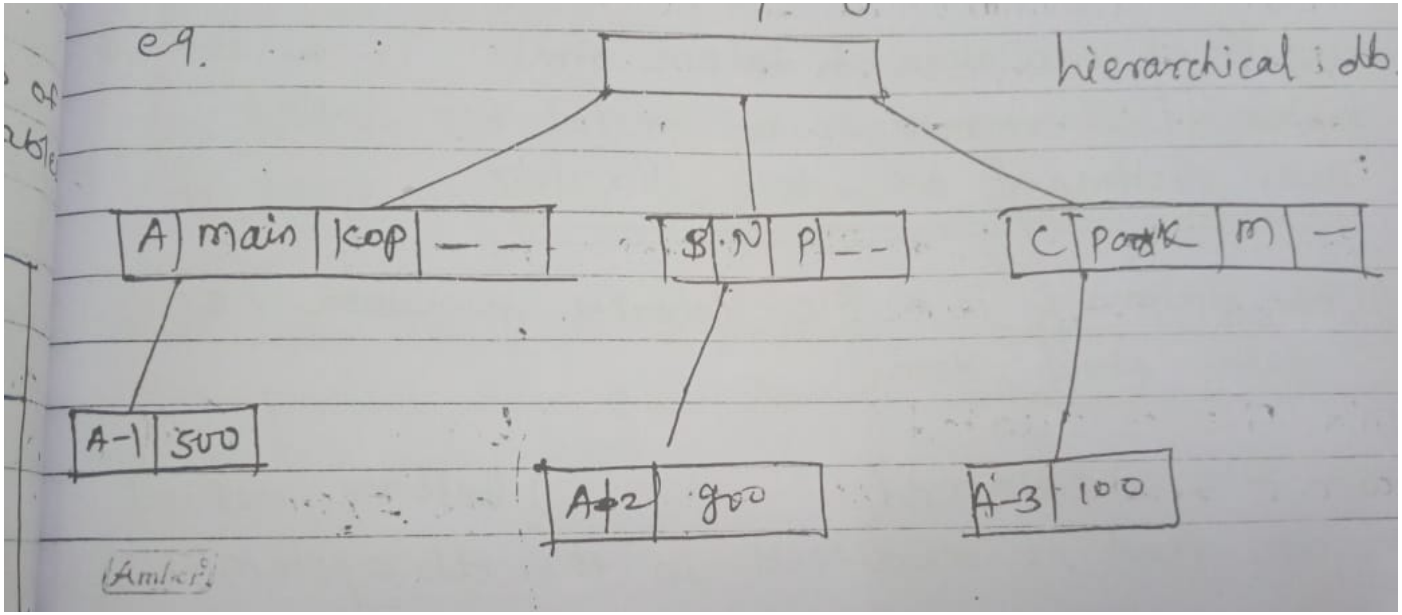
- The relational model uses a collection of tables to represent both data and the relationships among those data.
- Each table has multiple columns, and each column has a unique name.
- Tables are also known as relations.
- The relational model is an example of a record-based model.
- Record-based models are so named because the database is structured in fixed-format records of several types.
- 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

ID	*salary*
12121	90000
22222	95000
33456	87000
83821	92000

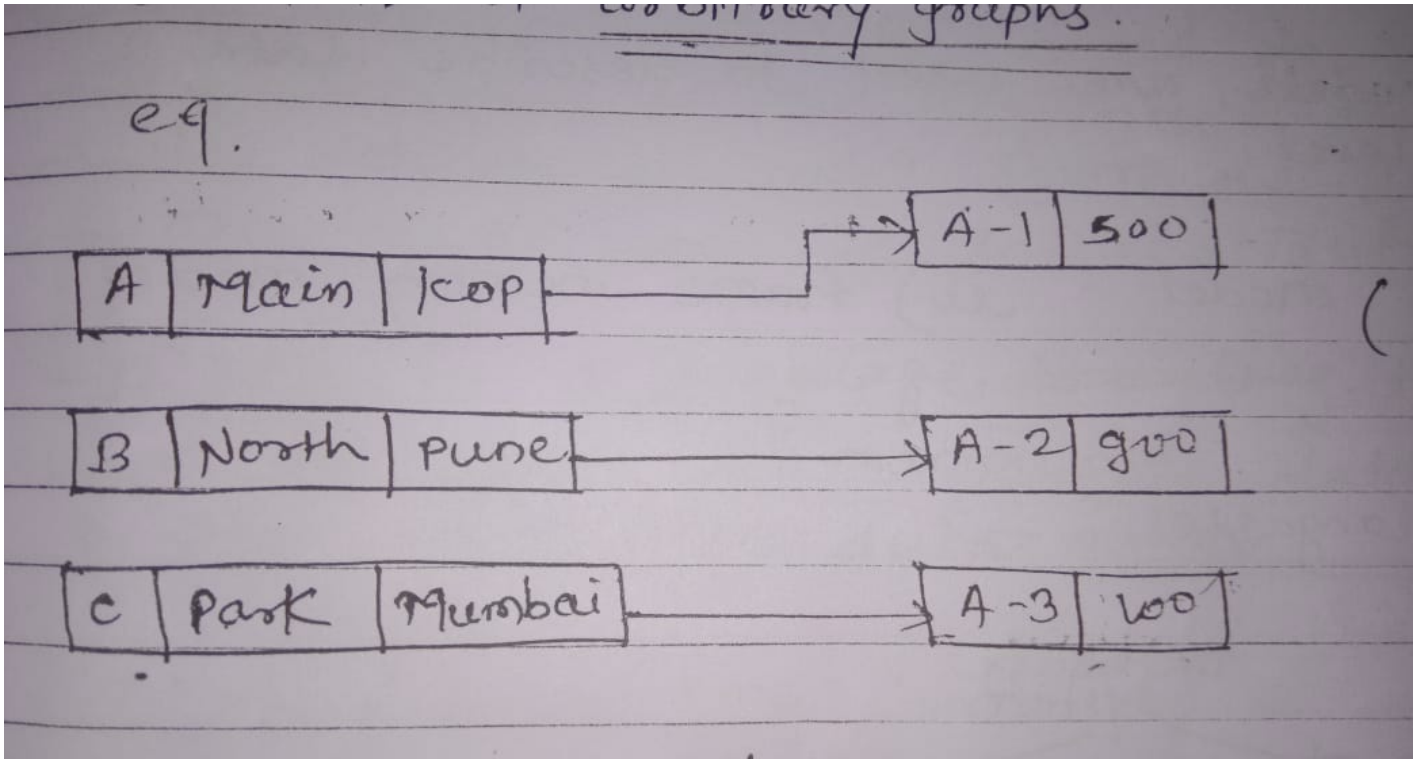
ID	*name*	*dept_name*	*salary*
12121	Wu	Finance	90000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
83821	Brandt	Comp. Sci.	92000

ID	*name*	*salary*	*dept_name*	*building*	*budget*
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
12121	Wu	90000	Finance	Painter	120000
15151	Mozart	40000	Music	Packard	80000
22222	Einstein	95000	Physics	Watson	70000
32343	El Said	60000	History	Painter	50000
33456	Gold	87000	Physics	Watson	70000
45565	Katz	75000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
76543	Singh	80000	Finance	Painter	120000
76766	Crick	72000	Biology	Watson	90000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000

Hierarchical Model



Network Model



Mapping Cardinality

The relationship set *advisor*, between the *instructor* and *student* entity sets may be one-to-one, one-to-many, many-to-one, or many-to-many. To distinguish among these types, we draw either a directed line (→) or an undirected line (—) between the relationship set and the entity set in question, as follows:

- **One-to-one:** We draw a directed line from the relationship set *advisor* to both entity sets *instructor* and *student* (see Figure 7.9a). This indicates that an instructor may advise at most one student, and a student may have at most one advisor.
- **One-to-many:** We draw a directed line from the relationship set *advisor* to the entity set *instructor* and an undirected line to the entity set *student* (see Figure 7.9b). This indicates that an instructor may advise many students, but a student may have at most one advisor.
- **Many-to-one:** We draw an undirected line from the relationship set *advisor* to the entity set *instructor* and a directed line to the entity set *student*. This indicates that an instructor may advise at most one student, but a student may have many advisors.
- **Many-to-many:** We draw an undirected line from the relationship set *advisor* to both entity sets *instructor* and *student* (see Figure 7.9c). This indicates that

→ (One) / —— (Many)

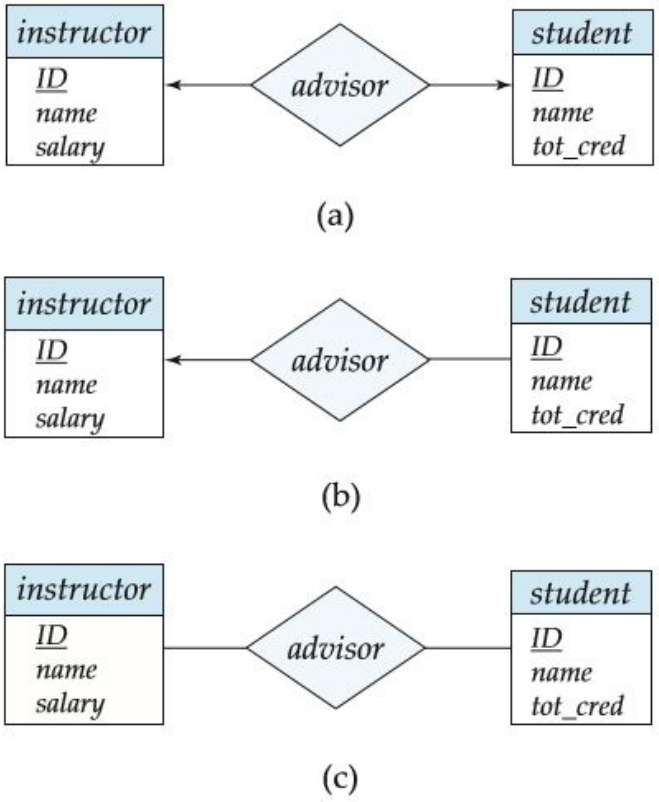
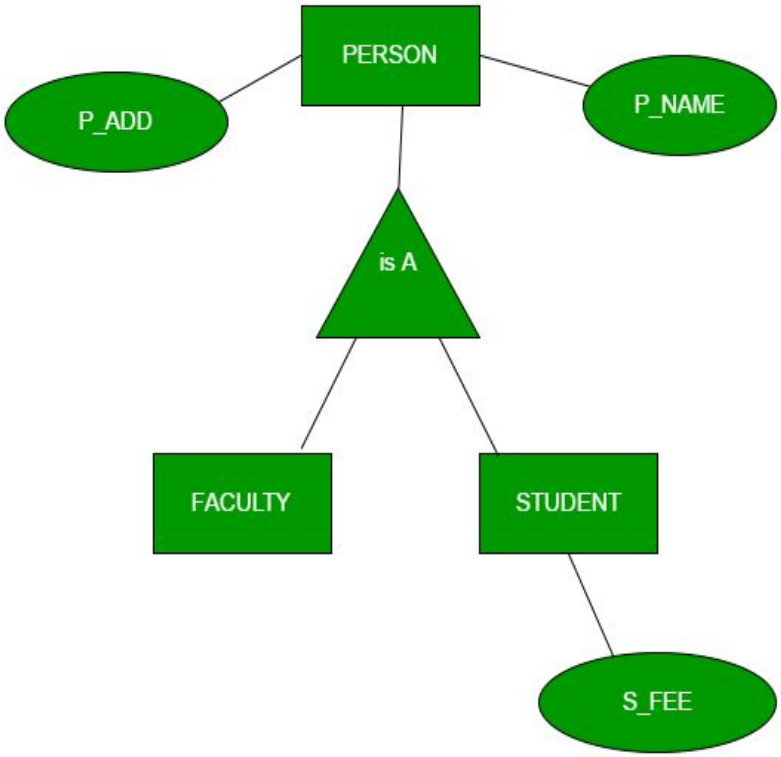
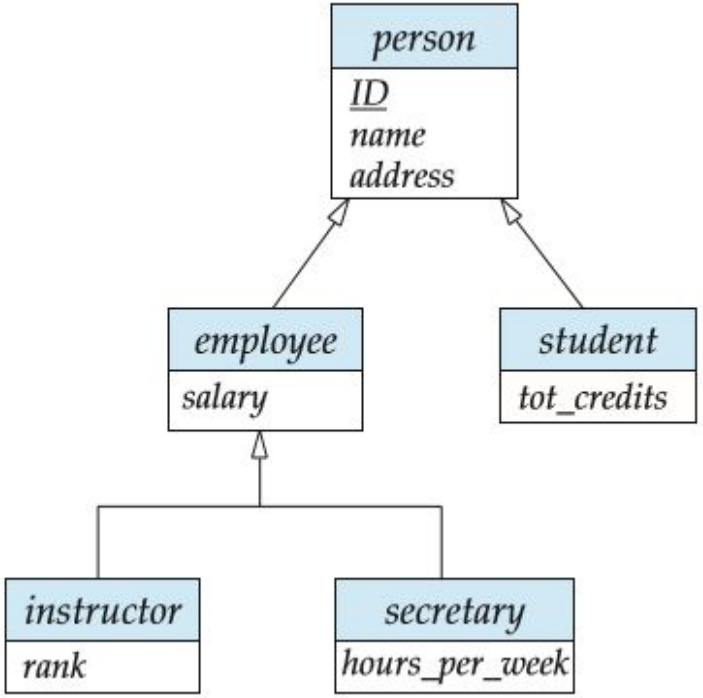


Figure 7.9 Relationships. (a) One-to-one. (b) One-to-many. (c) Many-to-many.

Generalization –

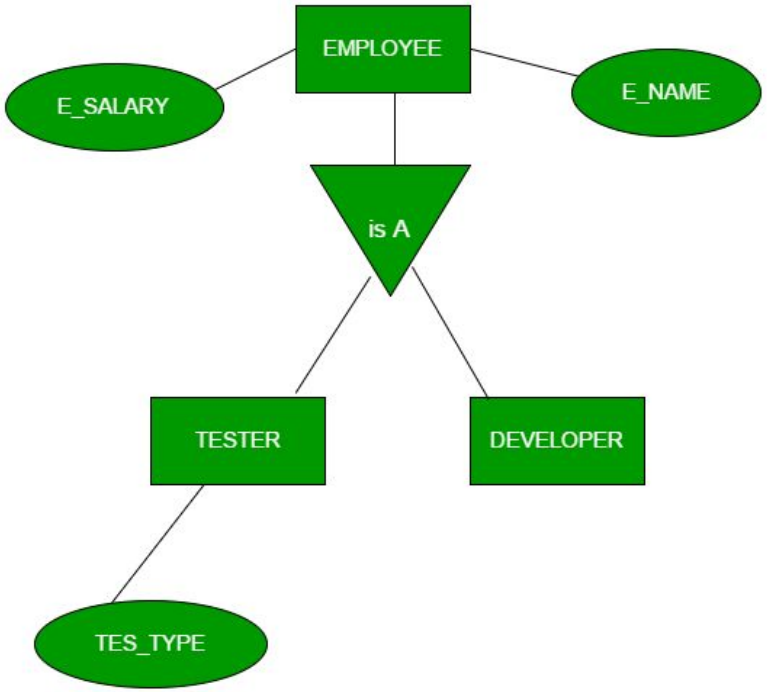
- Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it.
- It is a **bottom-up approach** in which two or more entities can be generalized to a higher level entity if they have some attributes in common.
- For Example, STUDENT and FACULTY can be generalized to a higher level entity called PERSON as shown in Figure . In this case, common attributes like P_NAME, P_ADD become part of higher entity (PERSON) and specialized attributes like S_FEE become part of specialized entity (STUDENT).


PERSON
P_ADD
P_NAME
is A
FACULTY
STUDENT
S_FEE

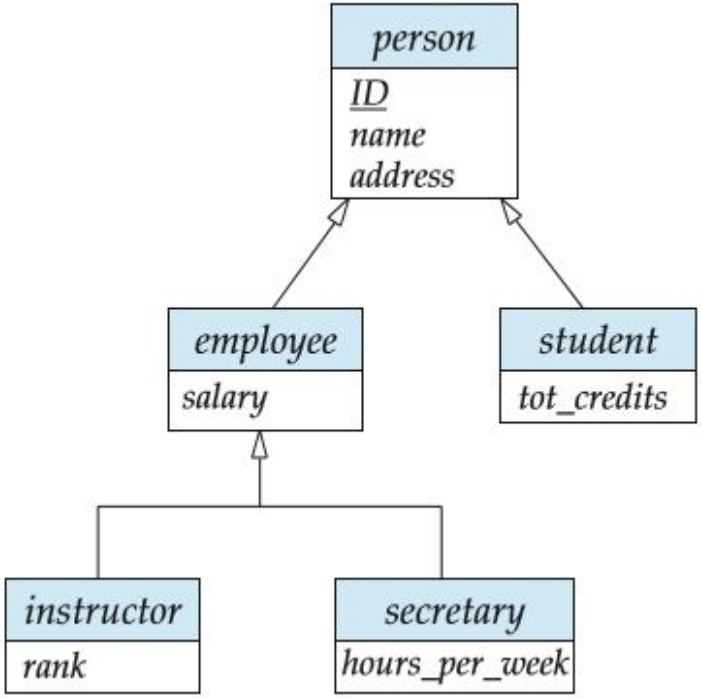

person
ID
name
address
employee
salary
student
tot_credits
instructor
rank
secretary
hours_per_week

Specialization

- An entity is divided into sub-entities based on their characteristics.
- It is a **top-down approach** where higher level entity is specialized into two or more lower level entities.
- For Example, EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc. as shown in Figure . In this case, common attributes like E_NAME, E_SAL etc. become part of higher entity (EMPLOYEE) and specialized attributes like TES_TYPE become part of specialized entity (TESTER).

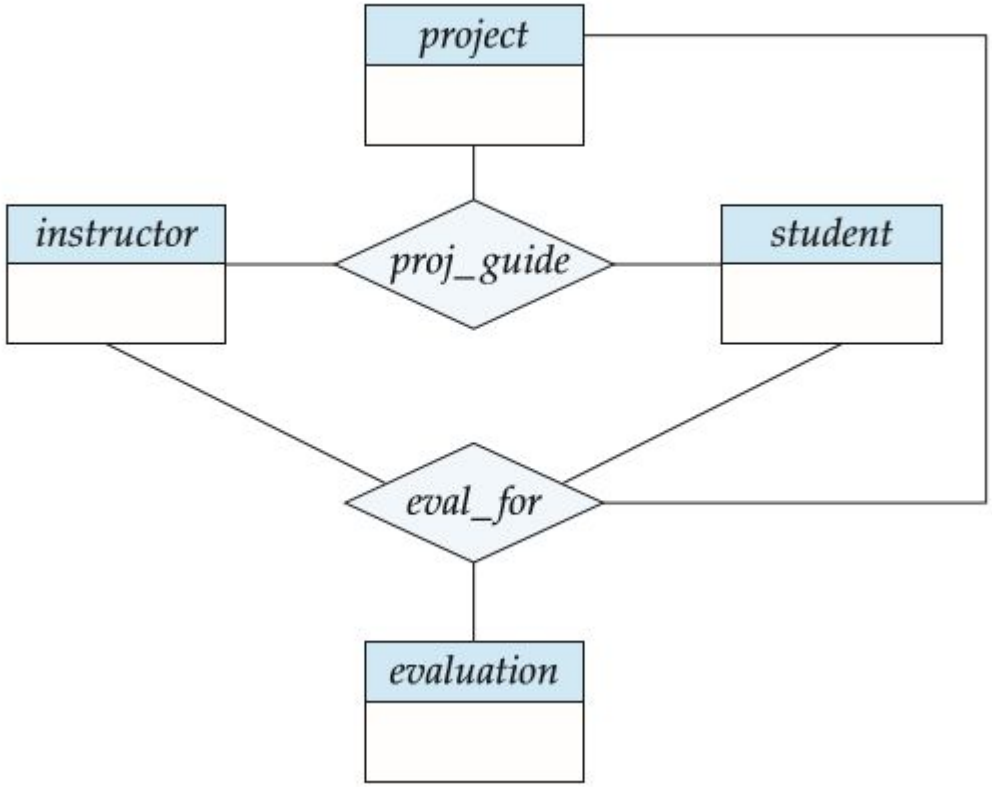


Specialization

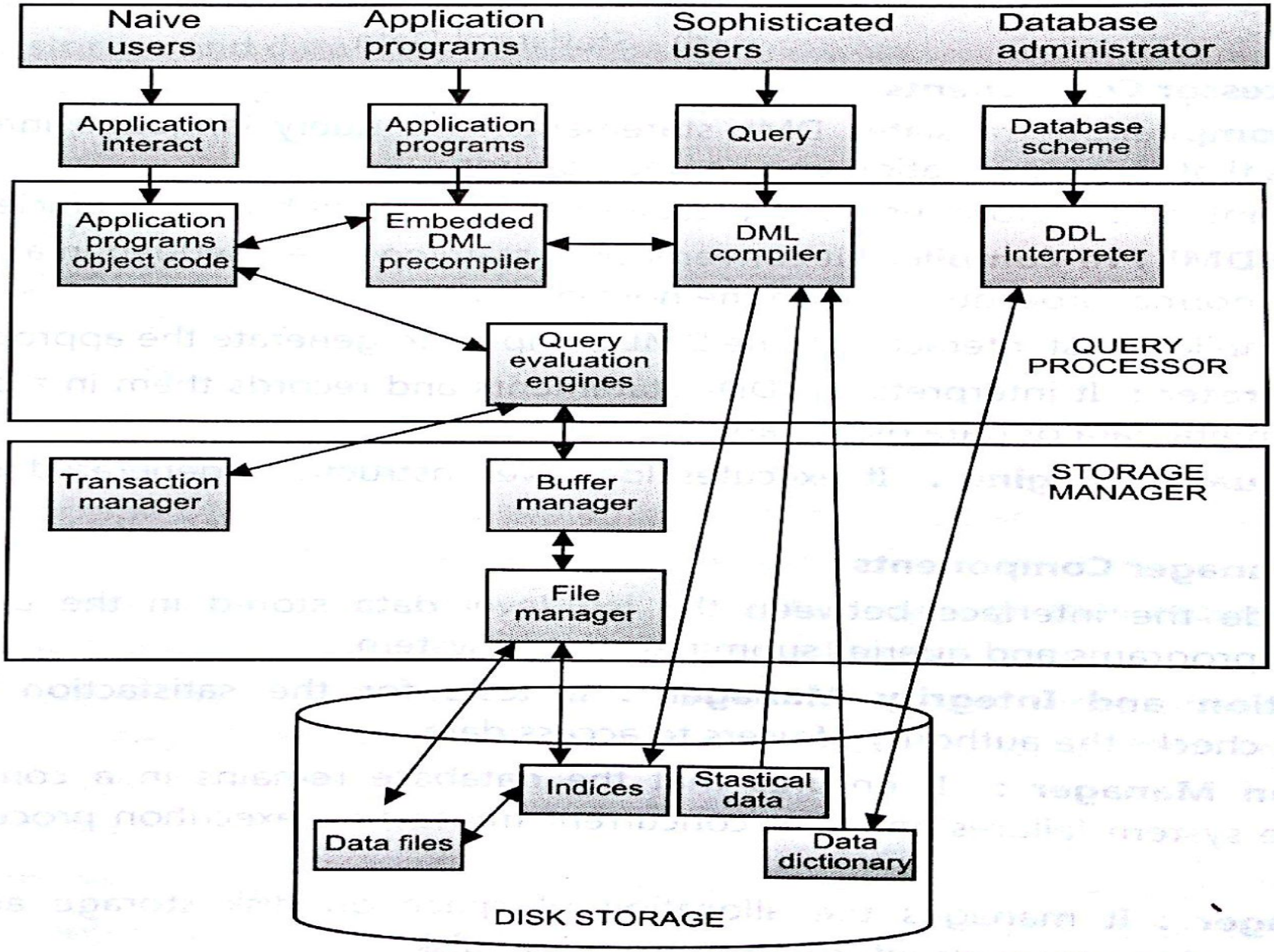


Aggregation

- An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher level entity.
- For Example A binary relationship eval_for between proj_guide and evaluation to represent which (student, project, instructor) combination an evaluation


project
instructor
proj_guide
student
eval_for
evaluation

Overall DBMS-Architecture



System structure

- **Database Users:**
- Users are differentiated by the way they expect to interact with the system:
- **Application programmers:**
  - Application programmers are **computer professionals who write application programs**. Application programmers can choose from many tools to develop user interfaces.
  - Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.
- **Sophisticated users:**
  - Sophisticated users interact with the system without writing programs. Instead, **they form their requests in a database query language.**
  - They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands.

- **Specialized users :**
  - **Specialized users are sophisticated users who write specialized database applications** that do not fit into the traditional data-processing framework.
  - Among these applications are computer-aided design systems, knowledge base and expert systems, systems that store data with complex data types (for example, graphics data and audio data)
- **Naïve users :**
  - Naive users are unsophisticated users who interact with the system by **invoking one of the application programs that have been written previously.**
  - For example, a bank teller who needs to transfer $50 from account A to account B invokes a program called transfer. This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred.

- **Database Administrator:**
- **Coordinates all the activities of the database system**. The database administrator has a good understanding of the enterprise's information resources and needs.
- ***Database administrator's duties include:***
  - **Schema definition:** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
  - **Storage structure and access method definition.**
  - **Schema and physical organization modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
  - **Granting user authority to access the database:** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access.
  - **Specifying integrity constraints.**
  - **Monitoring performance and responding to changes in requirements.**

❖ Query Processor:

- The query processor will accept query from user and solves it by accessing the database.

Parts of Query processor:

- **DDL interpreter**
- This will interprets DDL statements and fetch the definitions in the data dictionary.
- **DML compiler**
- a. This will translates DML statements in a query language into low level instructions that the query evaluation engine understands.
- b. A query can usually be translated into any of a number of alternative evaluation plans for same query result. DML compiler will select best plan for query optimization.
- **Query evaluation engine**
- This engine will execute low-level instructions generated by the DML compiler on DBMS.

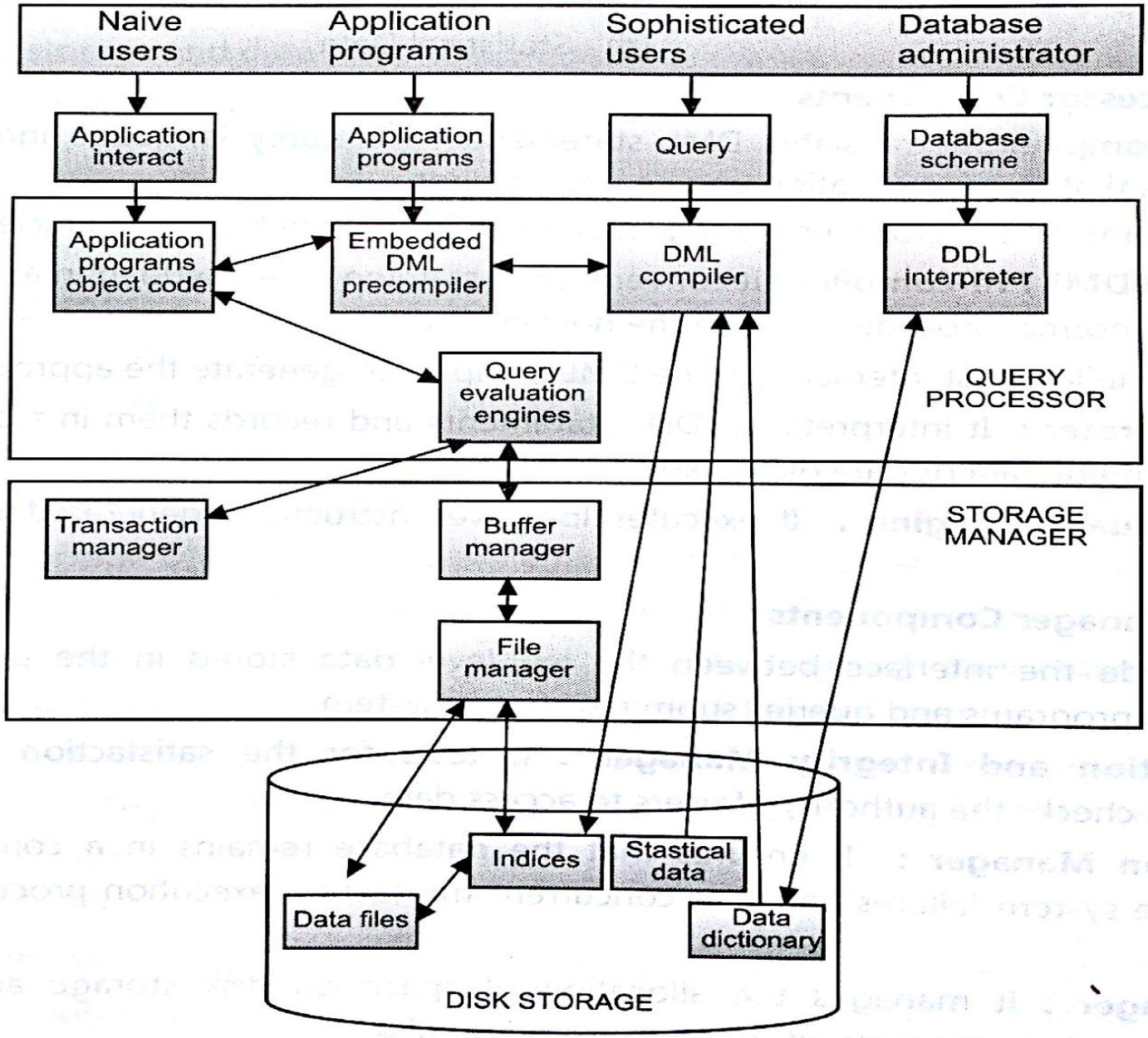
- **Storage Manager/Storage Management:**
- A storage manager is a program module which acts like **interface between the data stored in a database and the application programs and queries submitted to the system.**
- Thus, the storage manager is responsible for storing, retrieving and updating data in the database.
- **The storage manager components include:**
  - **Authorization and integrity manager:** Checks for integrity constraints and **authority of users** to access data.
  - **Transaction manager:** Ensures that the **database remains in a consistent** state although there are system failures.
  - **File manager:** Manages the allocation of **space on disk storage** and the data structures used to represent information stored on disk.
  - **Buffer manager:** It is responsible for **retrieving data from disk storage into main memory**. It enables the database to handle data sizes that are much larger than the size of main memory.
  - **Data structures implemented by storage manager.**
  - **Data files:** Stored in the database itself.
  - **Data dictionary:** Stores metadata about the structure of the database.
  - **Indices:** Provide fast access to data items.

References

- Abraham Silberschatz ,HenryKorth , S.Sudarshan,"Database System concepts",5th Edition ,McGraw Hill International Edition
- http://www.timeconsult.com/TemporalData/TemporalDB.html
- http://punarvasi.com/different-types-of-database-users/

END

Overall DBMS-Architecture



System structure

- **Database Users:**
- Users are differentiated by the way they expect to interact with the system:
- **Application programmers:**
  - Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.
  - Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.
- **Sophisticated users:**
  - Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language.
  - They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands.

- **Specialized users :**
  - Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.
  - Among these applications are computer-aided design systems, knowledge base and expert systems, systems that store data with complex data types (for example, graphics data and audio data)
- **Naïve users :**
  - Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
  - For example, a bank teller who needs to transfer $50 from account A to account B invokes a program called transfer. This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred.

- **Database Administrator:**
- Coordinates all the activities of the database system. The database administrator has a good understanding of the enterprise's information resources and needs.

- Database administrator's duties include:
  - **Schema definition:** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
  - **Storage structure and access method definition.**
  - **Schema and physical organization modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
  - **Granting user authority to access the database:** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access.
  - **Specifying integrity constraints.**
  - **Monitoring performance and responding to changes in requirements.**

❖ **Query Processor:**

- The query processor will accept query from user and solves it by accessing the database.

 Parts of Query processor:

- **DDL interpreter**
- This will interprets DDL statements and fetch the definitions in the data dictionary.
- **DML compiler**
- a. This will translates DML statements in a query language into low level instructions that the query evaluation engine understands.
- b. A query can usually be translated into any of a number of alternative evaluation plans for same query result. DML compiler will select best plan for query optimization.
- **Query evaluation engine**
- This engine will execute low-level instructions generated by the DML compiler on DBMS.

- **Storage Manager/Storage Management:**
- A storage manager is a program module which acts like interface between the data stored in a database and the application programs and queries submitted to the system.
- Thus, the storage manager is responsible for storing, retrieving and updating data in the database.
- **The storage manager components include:**
  - **Authorization and integrity manager:** Checks for integrity constraints and authority of users to access data.
  - **Transaction manager:** Ensures that the database remains in a consistent state although there are system failures.
  - **File manager:** Manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
  - **Buffer manager:** It is responsible for retrieving data from disk storage into main memory. It enables the database to handle data sizes that are much larger than the size of main memory.
  - **Data structures implemented by storage manager.**
  - **Data files:** Stored in the database itself.
  - **Data dictionary:** Stores metadata about the structure of the database.
  - **Indices:** Provide fast access to data items.

❖ Prime and Non Prime Attributes in DBMS with Example

❑ Prime Attributes –

Attribute set that belongs to any candidate key are called Prime Attributes.
(union of all the candidate key attribute)
{CK1 ∪ CK2 ∪ CK3 ∪ ……}
If Prime attribute determined by other attribute set, then more than one candidate key is possible.

For example,
If A is Candidate Key, and X→A, then, X is also Candidate Key

❑ **Non Prime Attribute –** Attribute set does not belongs to any candidate key are called Non Prime Attributes.

END

Modelling Temporal Data

- Temporal database stores data relating to time instances. It offers temporal data types and stores information relating to past, present and future time.
- Temporal databases provide a uniform and systematic way of dealing with historical data.
- More specifically the temporal aspects usually include valid time and transaction time. These attributes can be combined to form bitemporal data.

✔ **Valid time** is the time period during which a fact is true in the real world.(Historical database)

✔ **Transaction time** is the time period during which a fact stored in the database was known.(Rollback database)

✔ **Bitemporal data** combines both Valid and Transaction Time.

- **Example:-**Imagine that we come up with a temporal database storing data about the 18th century. The valid time of these facts is somewhere between 1700 and 1799, where as the transaction time starts when we insert the facts into the database, for example, January 21, 1998.

- Consider data about employees with respect to the real world is stored in table.

- Then, the following table could result:

EmpID	Name	Department	Salary	ValidTimeStart	ValidTimeEnd
10	John	Research	11000	1985	1990
10	John	Sales	11000	1990	1993
10	John	Sales	12000	1993	INF
11	Paul	Research	10000	1988	1995
12	George	Research	10500	1991	INF

- The above valid-time table stores the history of the employees with respect to the real world. The attributes **ValidTimeStart** and **ValidTimeEnd** actually represent a time interval which is closed at its lower and open at its upper bound.

- Thus, we see that during the time period [1985 - 1990), employee John was working in the research department, having a salary of 11000.

- Then he changed to the sales department, still earning 11000. In 1993, he got a salary raise to 12000.

- The upper bound INF denotes that the tuple is valid until further notice. It is now possible to store information about past states.

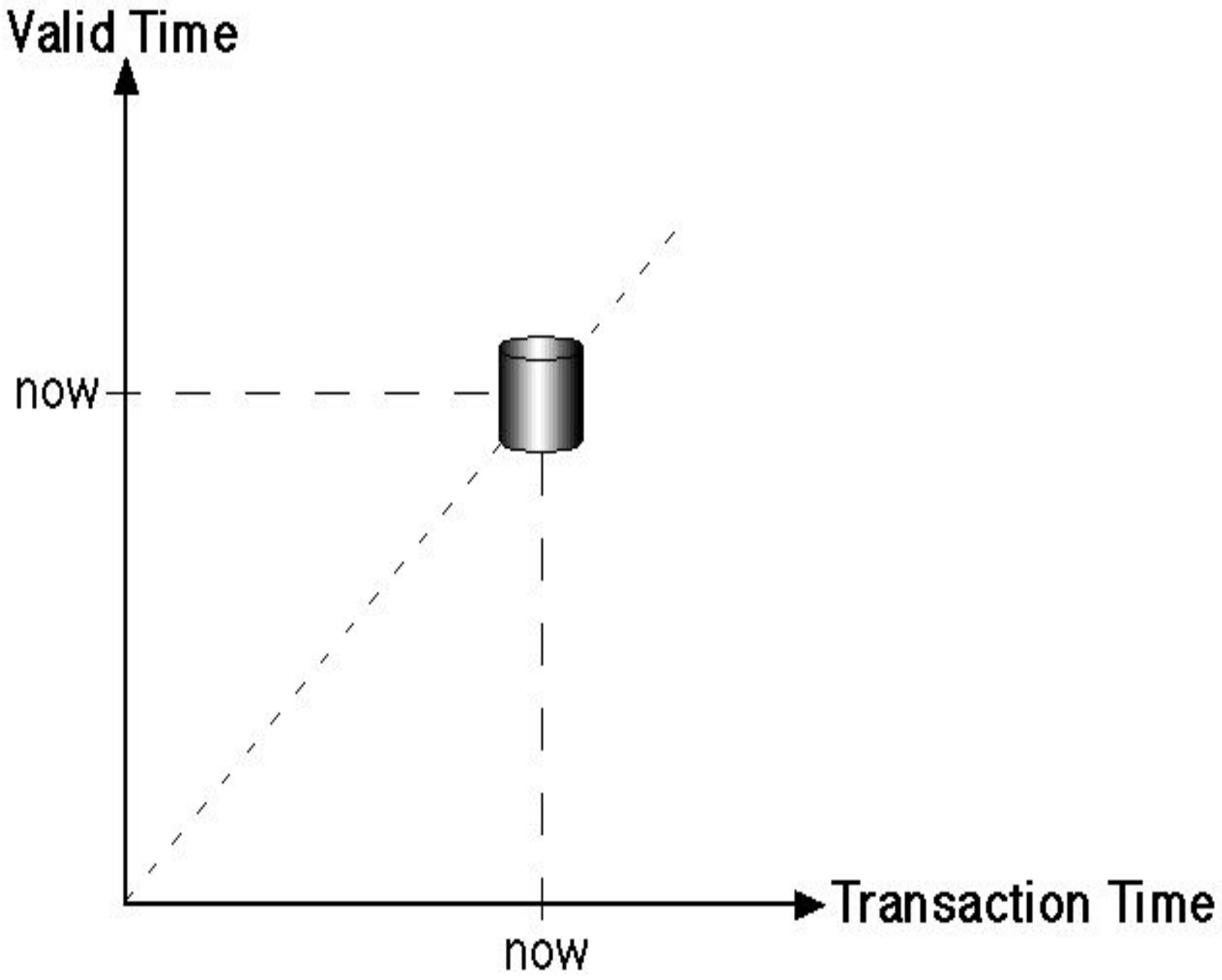
- We see that Paul was employed from 1988 until 1995. In the corresponding non-temporal table, this information was (physically) deleted when Paul left the company.

Different Forms of Temporal Databases

- The two different notions of time - valid time and transaction time - allow the distinction of different forms of temporal databases.
- A **historical database** stores data with respect to valid time, a **rollback database** stores data with respect to transaction time.
- A **bitemporal database** stores data with respect to both valid time and transaction time.

- Commercial DBMS are said to store only a single state of the real world, usually the most recent state. Such databases usually are called **snapshot databases**.

- A snapshot database in the context of valid time and transaction time is depicted in the following picture:


Valid Time
now
now
Transaction Time

- On the other hand, a **bitemporal DBMS** such as TimeDB stores the history of data with respect to both valid time and transaction time.

- The history of when data was stored in the database (transaction time) is limited to past and present database states, since it is managed by the system directly which does not know anything about future states.

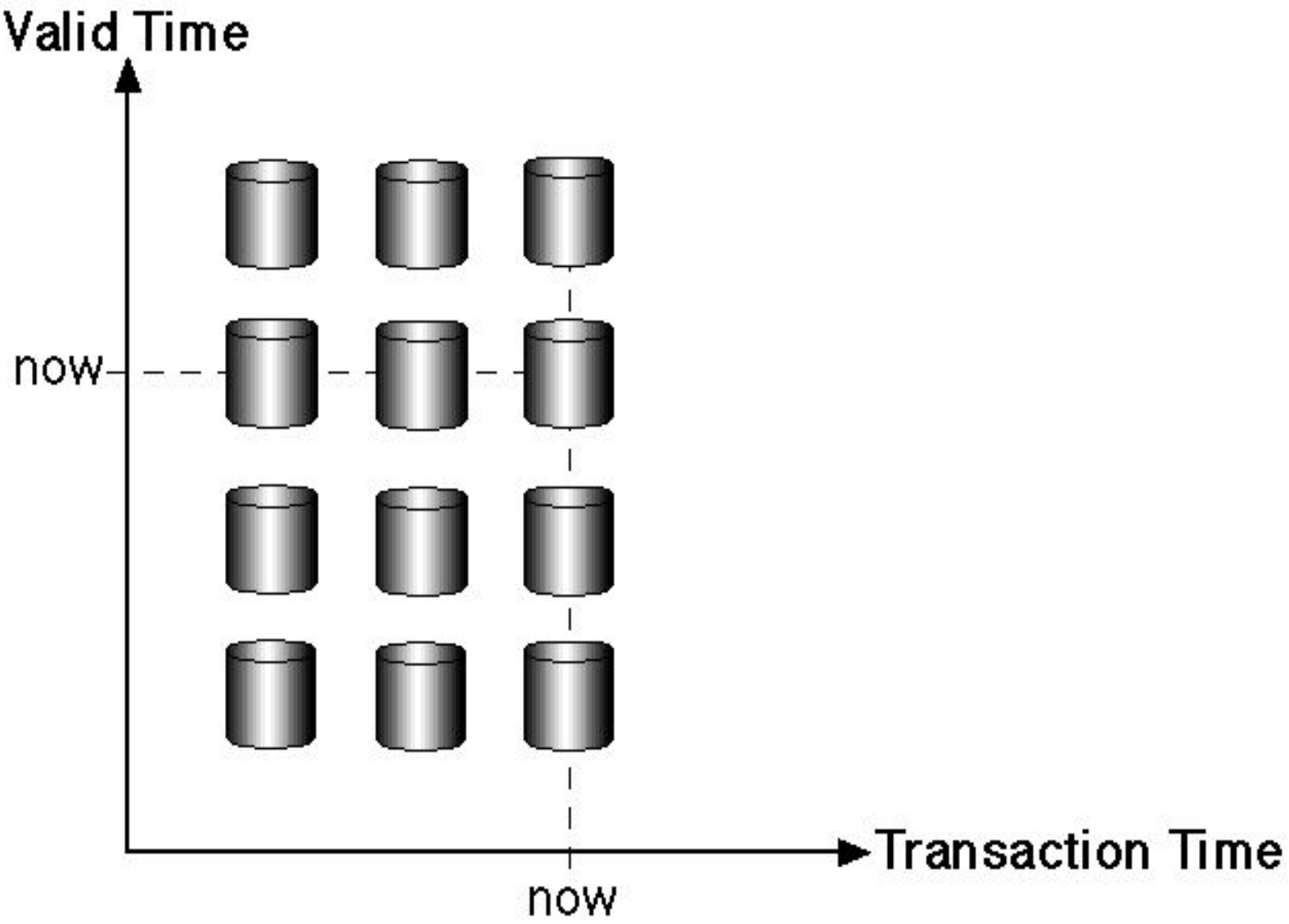
- A table in the bitemporal relational DBMS TimeDB may either be a snapshot table (storing only current data), a valid-time table (storing when the data is valid wrt. the real world), a transaction-time table (storing when the data was recorded in the database) or a bitemporal table (storing both valid time and transaction time).

- An extended version of SQL allows to specify which kind of table is needed when the table is created.

- Existing tables may also be altered (schema versioning). Additionally, it supports **temporal queries**, **temporal modification statements** and **temporal constraints**.

- The states stored in a bitemporal database are shown in fig.

- Of course, a temporal DBMS such as TimeDB does not store each database state separately as depicted in the figure, it stores valid time and/or transaction time for each tuple.


Valid Time
now
Transaction Time
now