BLACK N WHITE

Learn Today Lead Tomorrow

jag....

BCA 3RD SEMSESTER

DATABASE MANAGEMENT SYSTEM DCA2102

(set:- II)

1.) **Question :- Explain various storage devices and their characteristics.**

Answer :- There are mainly two different type of storage device . first ◇ Primary storage device (Internal Storage) & ◇ Secondary Storage device (External Storage) or also some different type of ◇ Internal Storage device RAM & ROM .

1. On the basis of volatility of information

- **Volatile memory:** It requires constant power to maintain the stored information. Volatile memory is typically used only for primary storage. The best example is RAM.
- **Non-volatile memory:** It can retain data even if the power supply is switched off. It is suitable for long-term or permanent storage of information. That is why it is used for secondary, tertiary and off-line storage.
- **Dynamic memory:** It is also a volatile memory. It is dynamic because each memory cell quickly loses its charge. So, it must be refreshed for hundreds of times each second.

2. On the basis of ability to access non-contiguous information

- **Random access:** It means that data in any storage location can be accessed in the same amount of time (generally very small amount of time). That is why random access memory is very much suited for primary storage.
- **Sequential access:** It means that data in different storage locations can't be accessed in the same time. The access time varies from one file to another file. In these cases access time consists of seek time (e.g. to position the read/write head correctly) and rotational delay (e.g. time taken to rotate the medium in such a way that the required file appears below the read/write head).

3. On the basis of ability to change information

- **Read/write storage**, or **mutable storage**: In this case information can be erased and fresh information can be rewritten for any number of times. Every computer must have at least some amount of storage of this kind.
- **Read only storage:** In this type of memory, data is recorded at the manufacturing and it is stored permanently. The information from the memory can be read but fresh information cannot be written. This is a **write once storage** (WORM). These are also called **immutable storage**. Immutable storage is used for tertiary and off-line storage. Examples include CD-ROM.
- **Slow write, fast read storage:** In this type of memory data can be overwritten for multiple times. But the write operation will be much slower than read operation. Examples include CD-RW.

4. On the basis of address-ability of information

- **Location addressable storage:** In this storage each memory location will have a unique address and data can be accessed by its numerical memory address. In modern computers, this method is followed in case of primary memory. This method is very efficient but very burdensome for human beings.

- **File system storage:** In this case information is divided into files of variable length, and a particular file is selected from human readable directory and file name. The operating system of a computer will provide the file system abstraction to make the operation more understandable. In modern computers, secondary, tertiary and off-line storage uses this file system.
- **Content addressable storage:** In this case each individually accessible unit of information is selected with a hash value or a short identifier. Here, there is no significance of memory address in which information is stored. Content addressable storage can be implemented either using software or hardware. Hardware is faster but more expensive.

5. On the basis of capacity and performance

- **Storage capacity:** It is the total amount of information that a storage device or medium can hold. It is expressed as a quantity of bits or bytes (e.g. 10.4 megabytes)
- **Storage density:** It refers to the compactness of stored information. It is the storage capacity of a medium divided with a unit of length, area or volume (e.g. 1.2 megabytes per square centimetre)
- **Latency:** It is the time taken to access a particular location in storage. It is generally 1 nanosecond for primary storage, 1 millisecond for secondary storage and 1 second for tertiary storage. We can also differentiate as read latency and write latency.
- **Throughput:** It is the rate at which information can be read from or written to the storage. In computer storage, throughput is usually expressed in terms of megabytes per second or MB/second.

2.) Question :- Explain the important properties of transactions that a DBMS must ensure to maintain data in the face of concurrent access and system failures .

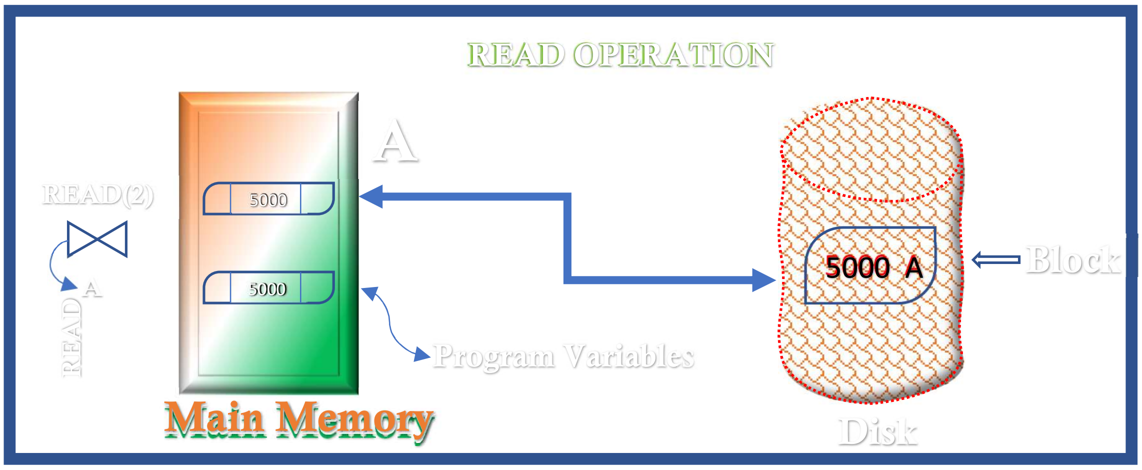
Answer :- A transaction is a set of changes that must all be made together. It is a program unit whose execution mayor may not change the contents of a database. Transaction is executed as a single unit. If the database was in consistent state before a transaction, then after execution of the transaction also, the database must be in a consistent. For example, a transfer of money from one bank account to another requires two changes to the database both must succeed or fail together.

⇨ **Process of Transaction**

The transaction is executed as a series of reads and writes of database objects, which are explained below:

⇨ **Read Operation**

To read a database object, it is first brought into main memory from disk, and then its value is copied into a program variable



To write a database object, an in-memory copy of the object is first modified and then written to disk.

Transaction Properties

There are four important properties of transaction that a DBMS must ensure to maintain data in the case of concurrent access and system failures. These are:

Atomicity: (all or nothing)

A transaction is said to be atomic if a transaction always executes all its actions in one step or not executes any actions at all It means either all or none of the transactions operations are performed.

Consistency: (No violation of integrity constraints)

A transaction must preserve the *consistency* of a database after the execution. The DBMS assumes that this property holds for each transaction. Ensuring this property of a transaction is the responsibility of the user.

Isolation: (concurrent changes invisible)

The transactions must behave as if they are executed in isolation. It means that if several transactions are executed concurrently the results must be same as if they were executed serially in some order. The data used during the execution of a transaction cannot be used by a second transaction until the first one is completed.

3.) Question :- What do you mean by cardinality ratio? What are its types? Explain by giving a suitable example.

Answer :- In the view of databases, cardinality refers to the uniqueness of data values that are contained in a column. High cardinality is nothing but the column contains a large percentage of totally unique values. Low cardinality is nothing but the column which has a lot of “repeats” in its data range.

Cardinality between the tables can be of type one-to-one, many-to-one or many-to-many.

Mapping Cardinality

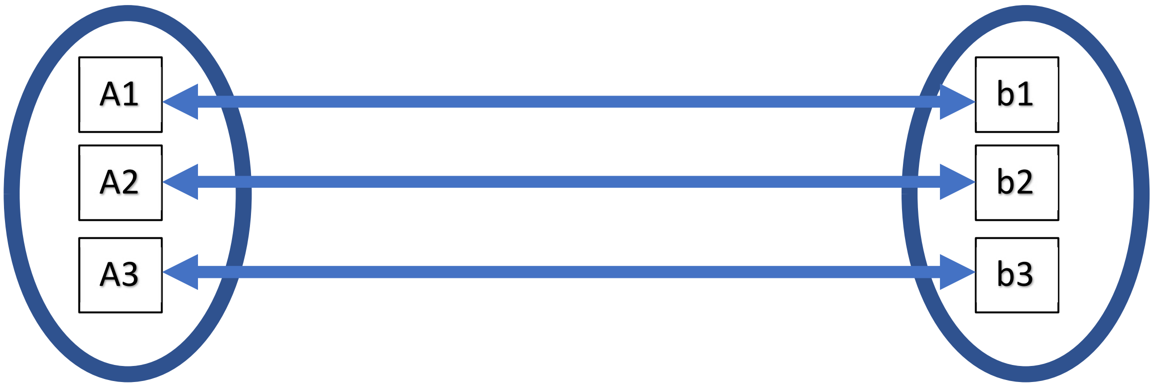
It is expressed as the number of entities to which another entity can be associated via a relationship set.

For binary relationship set there are entity set A and B then the mapping cardinality can be one of the following –

- One-to-one
- One-to-many
- Many-to-one
- Many-to-many

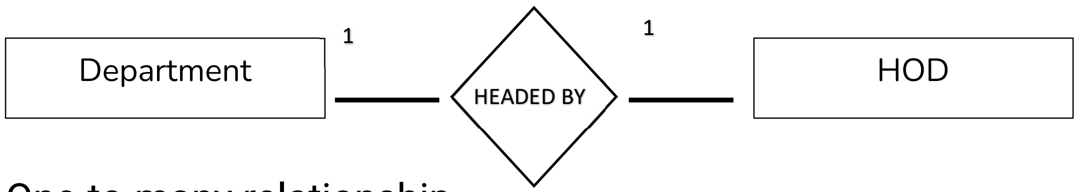
One-to-one relationship

One entity of A is associated with one entity of B.



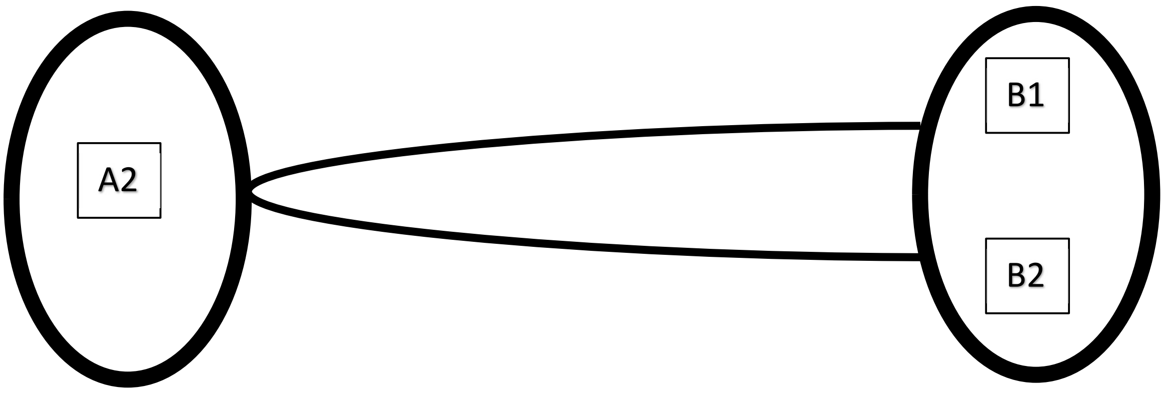
Example

Given below is an example of the one-to-one relationship in the mapping cardinality. Here, one department has one head of the department (HOD).



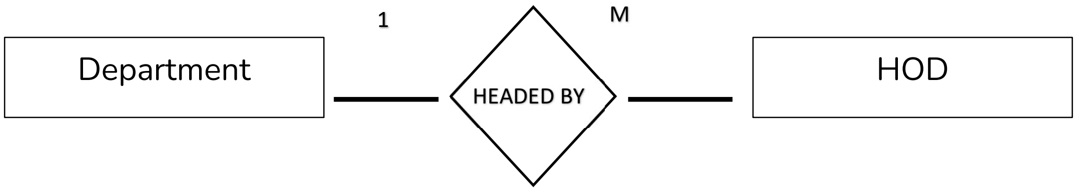
One-to-many relationship

An entity set A is associated with any number of entities in B with a possibility of zero and entity in B is associated with at most one entity in A



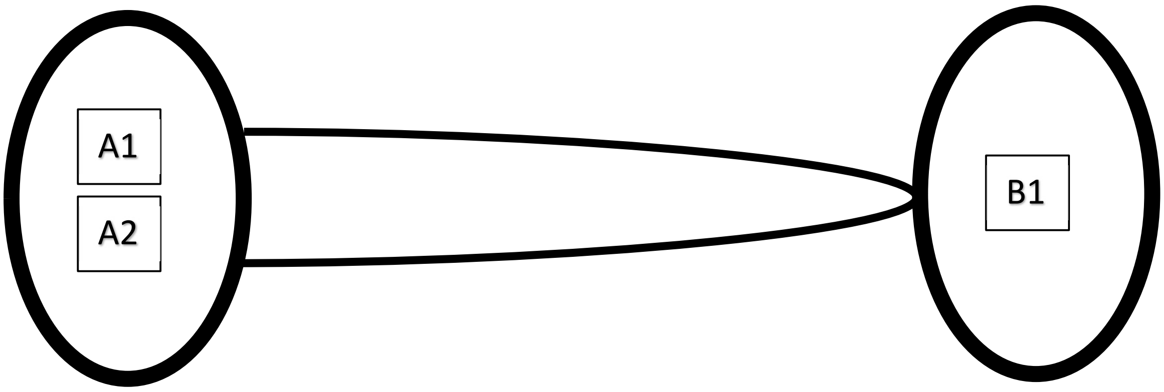
Example

Given below is an example of the one-to-many relationship in the mapping cardinality. Here, one department has many faculties.



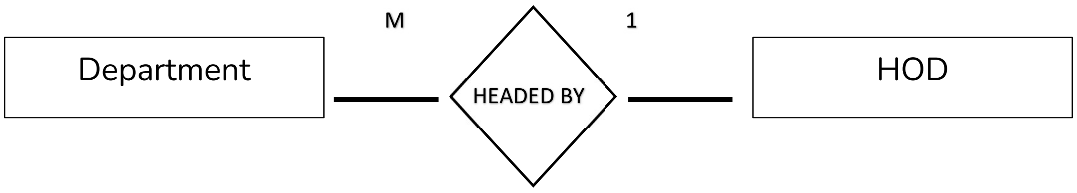
Many-to-one relationship

An entity set A is associated with at most one entity in B and an entity set in B can be associated with any number of entities in A with a possibility of zero.



Example

Given below is an example of the many-to-one relationship in the mapping cardinality. Here, many faculties work in one department.

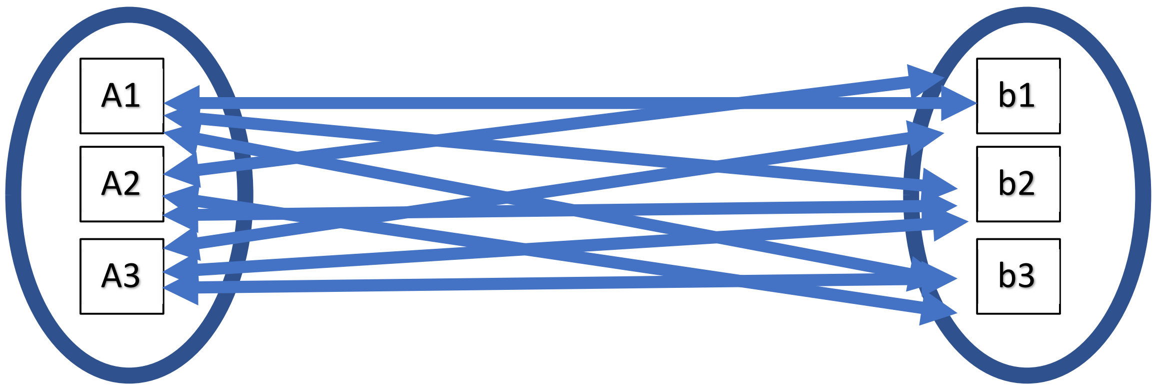


Many-to-many relationship

Many entities of A are associated with many entities of B.

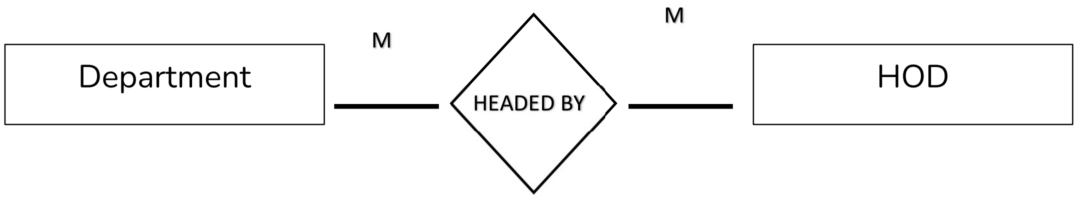
An entity in A is associated with many entities of B and an entity in B is associated with many entities of A.

Many to many=many to one + one to many



Example

Given below is an example of the many-to-many relationship in the mapping cardinality. Here, many employees work on many projects



(set:- II)

4.) Question :- What do you mean by Normalization? How BCNF is different from 3NF?

Answer :- The first goal during data normalization is to detect and remove all duplicate data by logically grouping data redundancies together. Whenever a piece of data is dependent on another, the two should be stored in proximity within that data set.

By getting rid of all anomalies and organizing unstructured data into a structured form, normalization greatly improves the usability of a data set. Data can be visualized more easily, insights could be extracted more efficiently, and information can be updated more quickly. As redundancies are merged together, the risk of errors and duplicates further making data even more disorganized is reduced. On top of all that, a normalized database takes less space, getting rid of many disk space problems, and increasing its overall performance significantly.

Boyce-Codd Normal Form (BCNF)

A higher version of the 3NF, the Boyce-Codd Normal Form is used to address the anomalies which might result if one more than one candidate key exists. Also known as 3.5 Normal Form, the BCNF must be in 3NF and in all functional dependencies (X → Y), X should be a super key.

Third normal form (3NF)

Tables in 3NF must be in 2NF and have no transitive functional dependencies on the primary key.

The following two NFs also exist but are rarely used:

5.) Question :- Explain Join strategies for parallel processing.

Answer :- Parallel processing can be described as a class of techniques which enables the system to achieve simultaneous data-processing tasks to increase the computational speed of a computer system.

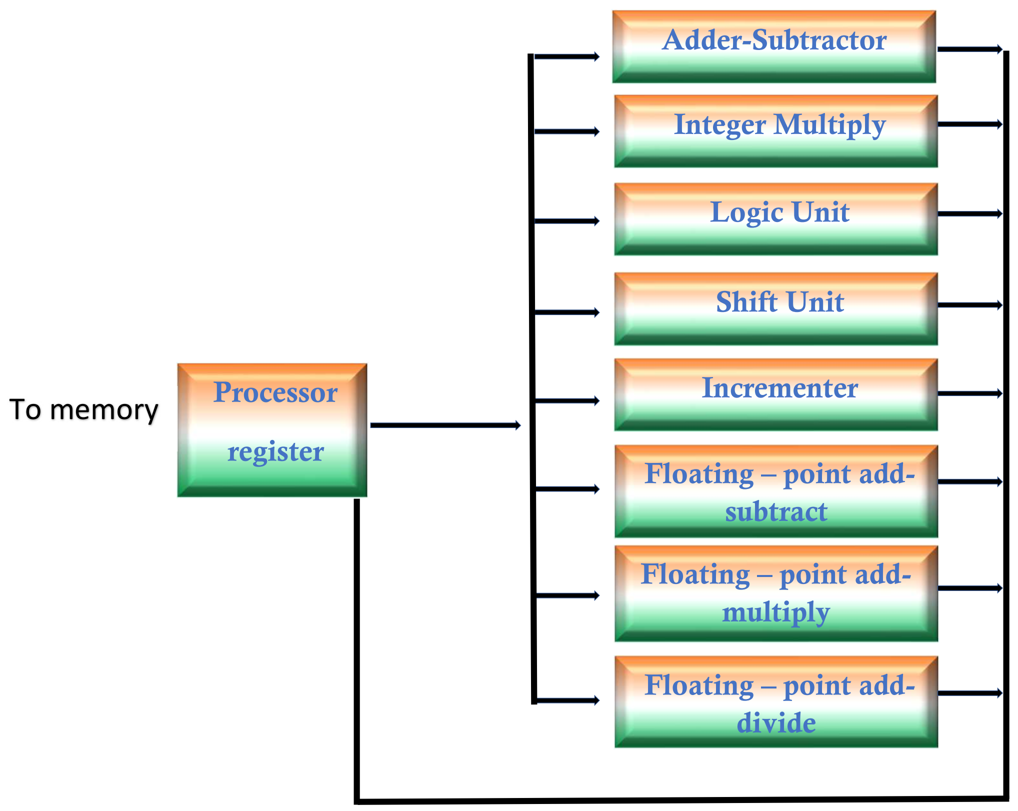
A parallel processing system can carry out simultaneous data-processing to achieve faster execution time. For instance, while an instruction is being processed in the ALU component of the CPU, the next instruction can be read from memory.

The primary purpose of parallel processing is to enhance the computer processing capability and increase its throughput, i.e. the amount of processing that can be accomplished during a given interval of time.

A parallel processing system can be achieved by having a multiplicity of functional units that perform identical or different operations simultaneously. The data can be distributed among various multiple functional units.

The following diagram shows one possible way of separating the execution unit into eight functional units operating in parallel.

The operation performed in each functional unit is indicated in each block if the diagram:



- The adder and integer multiplier performs the arithmetic operation with integer numbers.
- The floating-point operations are separated into three circuits operating in parallel.
- The logic, shift, and increment operations can be performed concurrently on different data. All units are independent of each other, so one number can be shifted while another number is being incremented.

6.) Question :- What are the features of Object -Oriented System? How it is different from RDBMS?

Answer :- An object-oriented database (OOD) is a database system that can work with complex data objects — that is, objects that mirror those used in object-oriented programming languages.

In object-oriented programming, *everything* is an object, and many objects are quite complex, having different properties and methods. An object-oriented database management system works in concert with an object-oriented programming language to facilitate the storage and retrieval of object-oriented data.

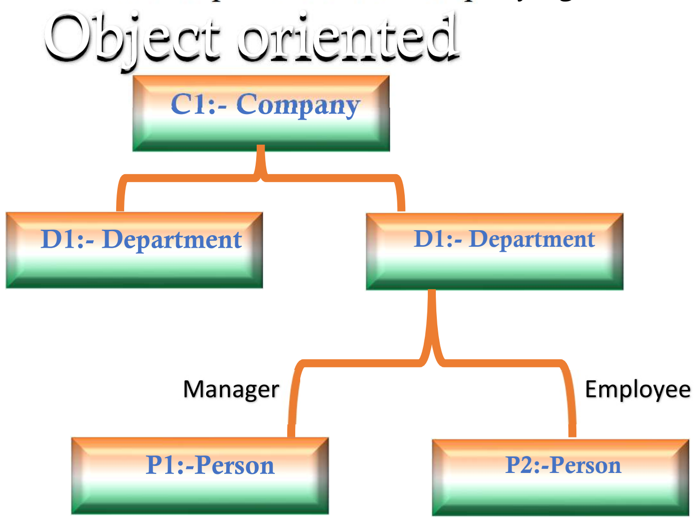
You might be thinking, “Wait, I use objects in my programming all the time. And I use a database. So, does that mean the database I use is an OOD?” Probably not, and the reason has to do with one of the main features of OOD: object data persistence.

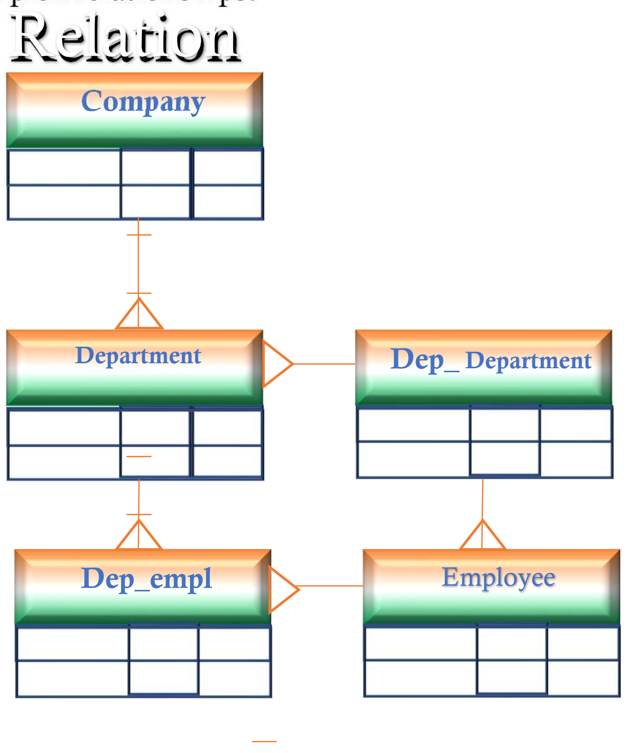
Relational database management systems (RDBMS) work with tables, with each row in the table representing a record. The columns in a row represent the attributes of an individual record. Associations between records (“A Company has many Employees. An Employee belongs to a Company”) are facilitated with foreign keys in one table referencing IDs in another table. These associations make up the “relational” part of relational databases.

Data values stored in relational databases are atomic and primitive. By primitive, we mean that they are types like characters, text strings, numbers, and hashes. Even though MySQL and SQLite support the JSON (JavaScript Object Notation) data type, that’s not the same as supporting objects in the sense that OODs do.

Contrast this with the OOD, which typically stores and manages objects directly on the database server’s disk. There are no tables, no rows, no columns, no foreign keys. There are only objects.

Associations between objects in an OOD can also be established and persist, which can lead to powerful and fast querying of data across complex relationships.


Object oriented
C1:- Company
D1:- Department
D1:- Department
Manager
Employee
P1:-Person
P2:-Person


Relation
Company
Department
Dep_ Department
Dep_empl
Employee