Advantages of DBMSs.

* Control of data redundancy
* Data consistency
* More information from the same amount of data
* Sharing of data Improved data integrity
* Improved security
* Enforcement of standards
* Economy of scale
* Balance of conflicting requirements
* Improved data accessibility and responsiveness
* Increased productivity
* Improved maintenance through data independence
* Increased concurrency
* Improved backup and recovery services

Disadvantages of DBMSs.

* Complexity
* Size
* Cost of DBMSs
* Additional hardware costs
* Cost of conversion
* Performance
* Higher impact of a failure

A transaction is an action or series of actions that are being performed by a single user or application program, which reads or updates the contents of the database.

Now let us take the example of a certain simple transaction. Suppose any worker transfers Rs 1000 from X’s account to Y’s account. This given small and simple transaction involves various low-level tasks.

**X’s Account**

Open\_Account(X)

Old\_Bank\_Balance = X.balance

New\_Bank\_Balance = Old\_Bank\_Balance – 1000

A.balance = New\_Bank\_Balance

Close\_Bank\_Account(X)

**Y’s Account**

Open\_Account(Y)

Old\_Bank\_Balance = Y.balance

New\_Bank\_Balance = Old\_Bank\_Balance + 1000

B.balance = New\_Bank\_Balance

Close\_Bank\_Account(Y)

**Isolation**

This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of the database state. Transactions occur independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed.

### Durability:

This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

State diag from slde

* **Active −**This is the state in which a transaction is being executed. Thus, it is like the initial state of any given transaction.
* **Partially Committed −**A transaction is in its partially committed state whenever it executes the final operation.
* **Failed −**In case any check made by a database recovery system fails, then that transaction is in a failed state. Remember that a failed transaction can not proceed further.
* **Aborted −**In case any check fails, leading the transaction to a failed state, the recovery manager then rolls all its write operations back on the database so that it can bring the DB (database) back to the original state (the state where it actually was prior to the transaction execution). The transactions in this state are known to be aborted. A DB recovery module can actually select one of these two operations after the abortion of a transaction –
  + Re-start
  + Kill the transaction
* **Committed −**We can say that a transaction is committed in case it actually executes all of its operations successfully. In such a case, all of its effects are now established permanently on the DB system.

SQL statements for Tran control

* COMMIT − to save the changes.
* ROLLBACK − to roll back the changes.
* SAVEPOINT − creates points within the groups of transactions in which to ROLLBACK.
* SET TRANSACTION − Places a name on a transaction.

**Difference between Shared Lock and Exclusive Lock :**

|  |  |  |
| --- | --- | --- |
| S.No. | Shared Lock | Exclusive Lock |
| 1. | Lock mode is read only operation. | Lock mode is read as well as write operation. |
| 2. | Shared lock can be placed on objects that do not have an exclusive lock already placed on them. | Exclusive lock can only be placed on objects that do no have any other kind of lock. |
| 3. | Prevents others from updating the data. | Prevents others from reading or updating the data. |
| 4. | Issued when transaction wants to read item that do not have an exclusive lock. | Issued when transaction wants to update unlocked item. |
| 5. | Any number of transaction can hold shared lock on an item. | Exclusive lock can be hold by only one transaction. |
| 6. | S-lock is requested using lock-S instruction. | X-lock is requested using lock-X instruction. |

THREE LEVEL ARCHITECTURE

**EXTERNAL LEVEL**

This is the **users’ view** of the database. This level describes that part of the database that is relevant to each user. The external view includes only those entities, attributes, and relationships in the ‘real world’ that the user is interested in. Other entities, attributes, or relationships that are not of interest may be represented in the database, but the user will be unaware of them.

For example, one user may view dates in the form (day, month, year), while another may view dates as (year, month, day).

**Conceptual Level**

The **community view** of the database. This level describes what data is stored in the database and the relationships among the data.

The conceptual level represents:

* all entities, their attributes, and their relationships;
* the constraints on the data;
* semantic information about the data;
* security and integrity information

For instance, the description of an entity should contain only data types of attributes (for example, integer, real, character) and their length (such as the maximum number of digits or characters), but not any storage considerations, such as the number of bytes occupied.

**Internal Level**

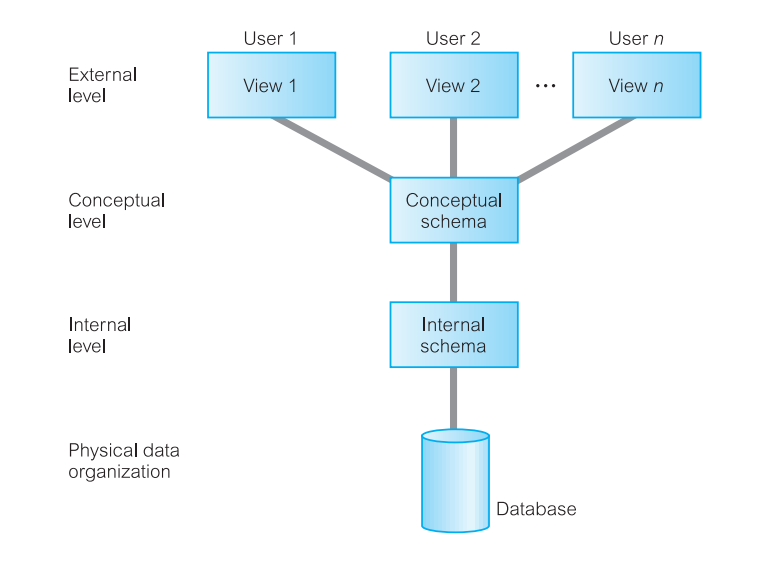
The **physical representation** of the database on the computer. This level describes how the data is stored in the database.

The internal level is concerned with such things as:

* storage space allocation for data and indexes;
* record descriptions for storage (with stored sizes for data items);
* record placement;
* data compression and data encryption techniques.

**Extra:**

Below the internal level there is a **physical level** that may be managed by the operating system under the direction of the DBMS. However, the functions of the DBMS and the operating system at the physical level are not clear-cut and vary from system to system.



**Functions/Objectives/Advantages of DBMS**

**Data storage, retrieval, and update:** A DBMS must furnish users with the ability to store, retrieve, and update data in the database.

**A user-accessible catalog:**  A DBMS must furnish a catalog in which descriptions of data items are stored and which is accessible to users.

**Transaction support:** A DBMS must furnish a mechanism which will ensure either that all the updates corresponding to a given transaction are made or that none of them is made.

**Concurrency control services:** A DBMS must furnish a mechanism to ensure that the database is updated correctly when multiple users are updating the database concurrently.

**Recovery services:** A DBMS must furnish a mechanism for recovering the database in the event that the database is damaged in any way

**Authorization services**: A DBMS must furnish a mechanism to ensure that only authorized users can access the database

**Support for data communication:** A DBMS must be capable of integrating with communication software.

**Integrity services:** A DBMS must furnish a means to ensure that both the data in the database and changes to the data follow certain rules.

**Services to promote data independence:** A DBMS must include facilities to support the independence of programs from the actual structure of the database.

**Utility services:** A DBMS should provide a set of utility services

**Superkey:**

An attribute, or set of attributes, that uniquely identifies a tuple within a relation.

**Candidate key**:

A superkey such that no proper subset is a superkey within the relation.

**Composite key:**

When a key consists of more than one attribute, we call it a composite key.

**Primary key:**

The candidate key that is selected to identify tuples uniquely within the relation.

**Foreign key:**

An attribute, or set of attributes, within one relation that matches the candidate key of some (possibly the same) relation.

**Integrity Constraints**:

**Null:**

Represents a value for an attribute that is currently unknown or is not applicable for this tuple.

**Entity integrity:**

In a base relation, no attribute of a primary key can be null.

**Referential integrity:**

If a foreign key exists in a relation, either the foreign key value must match a candidate key value of some tuple in its home relation or the foreign key value must be wholly null.

**General constraints:**

Additional rules specified by the users or database administrators of a database that define or constrain some aspect of the enterprise.

**Base relation:**

A named relation corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database.

**View:**

The dynamic result of one or more relational operations operating on the base relations to produce another relation. A view is a virtual relation that does not necessarily exist in the database but can be produced upon request by a particular user, at the time of request.