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.

You can use this stored data for computing and presentation. In many systems, data independence is an essential function for components of the system.

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 with an effect on the conceptual schema. Any change done would be absorbed by the mapping between the conceptual and internal levels. Physical data independence is achieved by the presence of the internal level of the database and then the transformation from the conceptual level of the database to the internal level.

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

Logical Data Independence

Logical Data Independence is the ability to change the conceptual scheme without changing

1. External views
2. 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. Examples of changes under Logical Data Independence Due to Logical independence, any of the below change will not affect the external layer.

1. Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs
2. Merging two records into one
3. 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

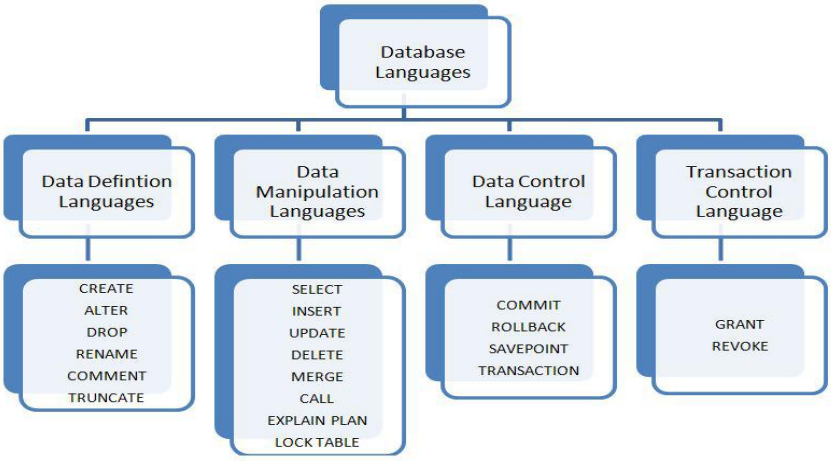
Importance of Data Independence

- Helps you to improve the quality of the data
- Database system maintenance becomes affordable
- Enforcement of standards and improvement in database security
- You don't need to alter data structure in application programs
- Permit developers to focus on the general structure of the Database rather than worrying about the internal implementation
- It allows you to improve state which is undamaged or undivided
- Database incongruity is vastly reduced.
- Easily make modifications in the physical level is needed to improve the performance of the system.

Database Languages in DBMS

Database languages are the languages that provide the facility to specify the database schema and to express database queries and updates. They are further divided into four categories but all are purposed to provide the facilities in their specific ways and the most widely used database language is SQL language.

Let us understand the basics of the **Database Languages**, what's their task are and how they perform it.



1) Data Definition Language

A **Data Definition Language** is a special kind of language that used to specify the database schema by the set of definitions. DDL is also purposed to specify the additional properties of the data.

There is a special type of DDL that specify the storage structure and access methods used by the database system. The hidden details for the implementation of the database schemas from the users are defined by these statements. Also, the data values that are stored in the database are required to satisfy certain **consistency constraints**.

For e.g. – A university is required to fulfill the criteria of the marks of the students never be negative. DDL provides facilities to specify such type of constraints. Every time the database gets updated, these constraints are being checked by the database system. These constraints can refer to the arbitrary predicate for the database. But these arbitrary predicates are costly for the tests. Therefore, in that case, integrity constraints are implemented by the database systems so that these constraints can be tested with less overhead and more accuracy.

- **Domain Constraints**

These are the most basic form of integrity constraint. In the database, they can be easily tested by the system every time a new data item is entered. Domain needs to be related to every attribute such as integer, character, date/time types. These attributes related to domain further acts as constraints on the values associated with it.

- **Referential Integrity**

Referential integrity means the value that is in the one relation must also appear in a certain set of attributes in another relation.
Modifications in the database could turn out to violate the referential integrity and such actions are rejected by the normal procedure.

- **Assertions**

Assertions are the conditions that are needed to be satisfied by the database. The above two constraints, Domain Constraints and Referential Integrity Constraints are the special forms of assertions.
System checks for the validity of the assertion and if it does not fit the validity criteria, no further modifications are done to the database.

- Authorization simply means to provide access or permission to the differentiated users on various data values in the database.

Most common authorizations are:

1. **Read Authorization –** It only allows the user to read the data.
2. **Insert Authorization –** It only allows the user to insert the data.
3. **Update Authorization –** It only allows the user to modify the data.

---

4. **Delete Authorization –** It only allows the user to delete the data.

All of these different types of authorization are mutually exhaustive such that user can be assigned one or some or all of the combination of these types.
DDL acts like any other programming language, in which an input is taken by the user and the output is generated. These outputs are further stored in the Data Dictionary.

2) Data-Manipulation Language

Data Manipulation Language (DML) is a language that provides the facility to access or manipulate the data to the user. There is various type of access, such as, **retrieval, insertion, modification, and deletion** of the information that is stored in the database.

DML are further classified into two types:

- **Procedural DMLs**
It is required from the user to specify what type of data is needed and how one can reach up to that data.
- **Declarative DMLs**
These are also referred to a **Non – Procedural DML** that requires from the user only to specify what type of data is needed and it doesn't specify how to reach up to that data.
Declarative DMLs are easy to learn and use than Procedural DMLs.
But in **Declarative DMLs** when the user doesn't specify how to reach up to that data, this is the responsibility of the database system to figure out itself how to do so.

The **query** is a statement that requests for the retrieval of the information. **Query Language** is the portion of the **DML** that involves information retrieval. Therefore, **Query Language** and **Data Manipulation Language**, these terms are used synonymously, although they are technically incorrect.

3) Data Control Language

Data Control Language provides the facility of the authorization in the database. All the commands used in **DDL** and **DML** can further be authorized with **DCL**.

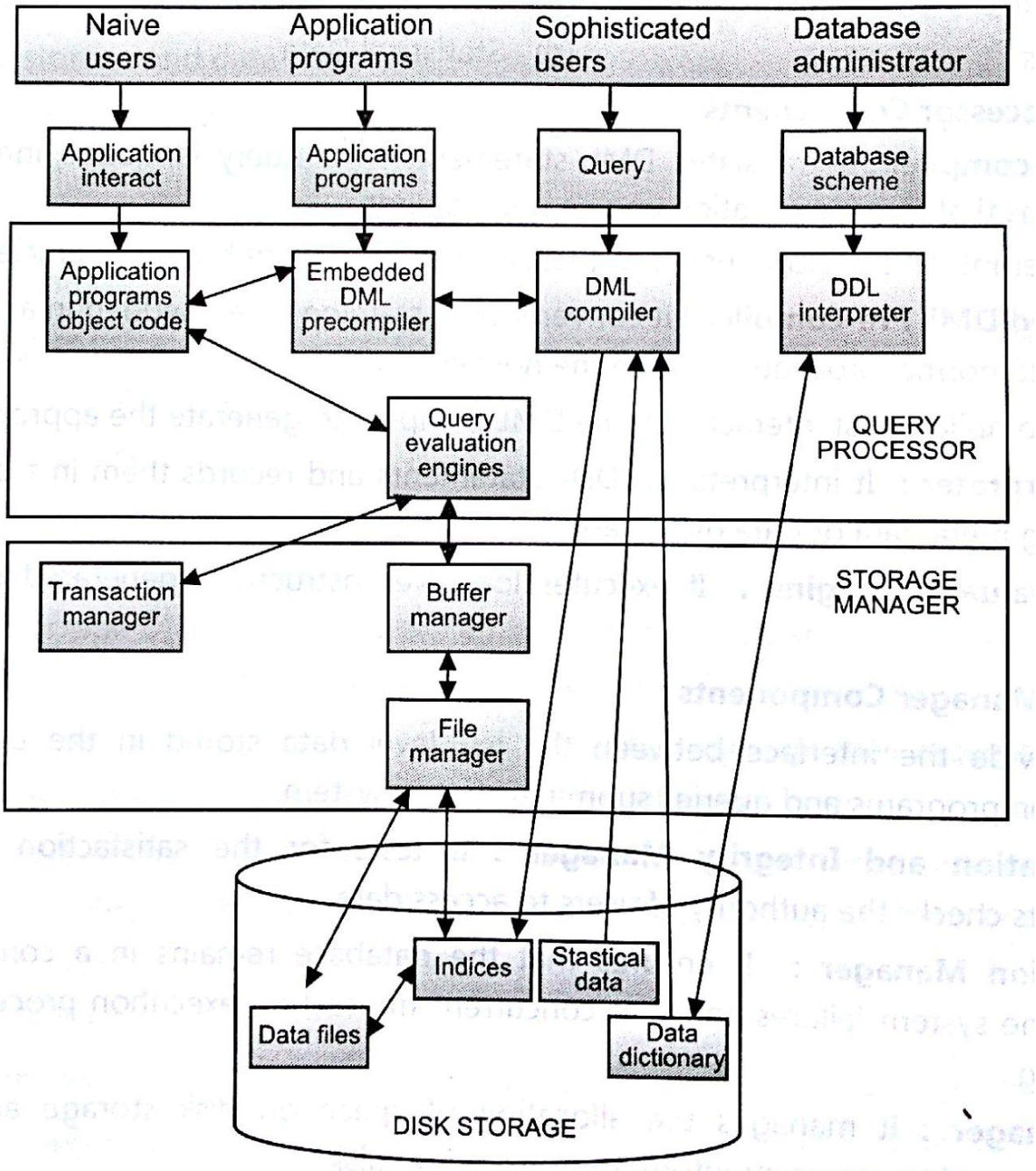
The drawback of **DCL** is that one cannot rollback the command as **DCL** allows only implicit commit. This mainly occurs in Oracle Database. Some of the **commands of DCL** are:

a. **Grant:** It allows only specific user to perform specific tasks.
b. **Revoke:** It allows the cancellation of the previously granted permissions or the denied one.

4) Transaction Control Language

Transaction Control Language is used to manage the transactions in the database. The functions performed by DML are further managed by this language. It allows the statements to be grouped together into logical transactions. TCL performs the following tasks:

a. **Commit:** It is used to save the transaction in the database.
b. **Rollback:** It is used to restore the database to the last committed state.
c. **Savepoint:** It is used to store the transaction temporarily such that one can easily rollback to the transaction when needed.



System structure

DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations such as insert, delete, update and retrieval on the database. The components of DBMS perform these requested operations on the database and provide necessary data to the users.

Components of DBMS are broadly classified as follows :

1. Query Processor :

(a) DML Compiler
(b) Embedded DML pre-compiler
(c) DDL Interpreter
(d) Query Evaluation Engine

2. Storage Manager :

(a) Authorization and Integrity Manager
(b) Transaction Manager
(c) File Manager
(d) Buffer Manager

3. Data Structure :

(a) Data Files
(b) Data Dictionary
(c) Indices
(d) Statistical Data

1. Query Processor Components :

• **DML compiler :** It translates DML statements in a query language into low level instructions that query evaluation engine understands. It also attempts to transform user's request into an equivalent but more efficient form.
• **Embedded DML compiler :** It converts DML statements embedded in an application program to normal procedure calls in the host language. The compiler must interact with the DML compiler to generate the appropriate code.
• **DDL Interpreter :** It interprets the DDL statements and records them in a set of tables containing meta data or data dictionary.
• **Query Evaluation Engine :** It executes low-level instructions generated by the DML compiler.

2. Storage Manager Components :

They provide the interface between the low-level data stored in the database and application programs and queries submitted to the system.
• **Authorization and Integrity Manager :** It tests for the satisfaction of integrity constraints checks the authority of users to access data.
• **Transaction Manager :** It ensures that the database remains in a consistent state despite the system failures and that concurrent transaction execution proceeds without conflicting.
• **File Manager :** It 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 fetching data from disk storage into main memory and deciding what data to cache in memory.

3. Data Structures :

Following data structures are required as a part of the physical system implementation.
• **Data Files :** It stores the database.
• **Data Dictionary :**

- Data Dictionary, which stores metadata about the database, in particular the schema of the database.
- names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.
- Detailed information on physical database design such as storage structure, access paths, files and record sizes.
- Usage statistics such as frequency of query and transactions.
- Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as a important part of the DBMS

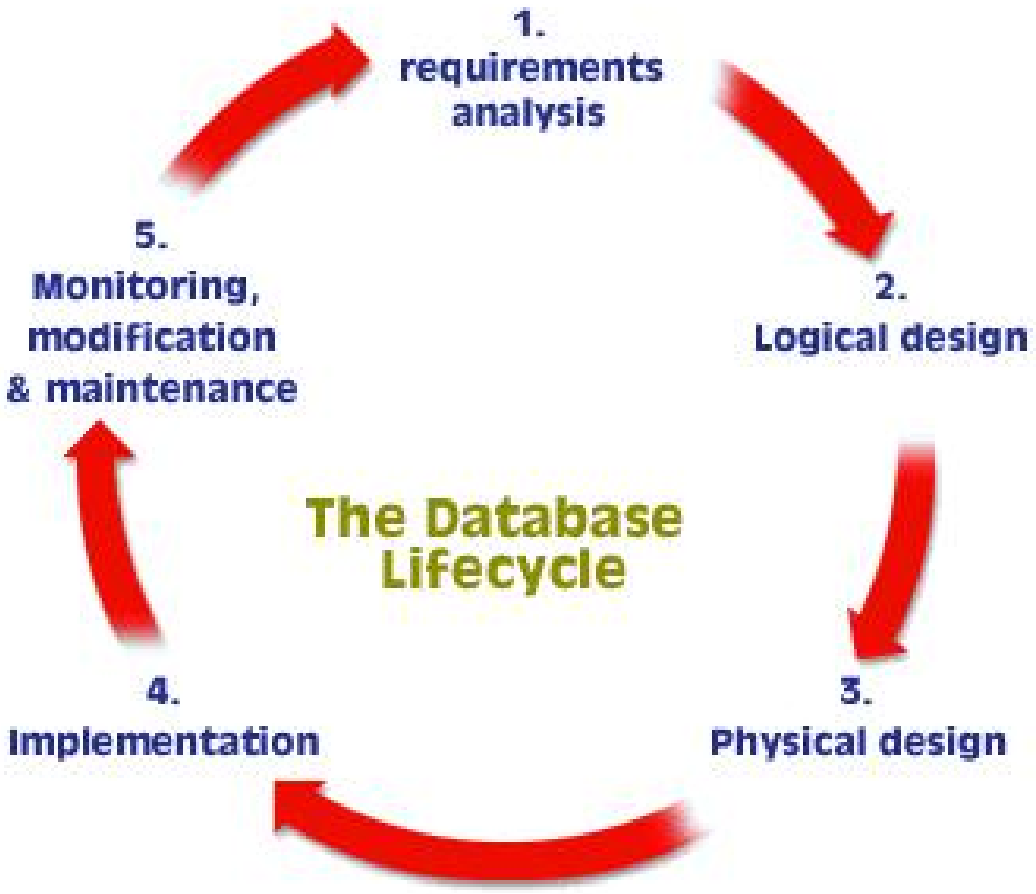
• **Indices :** Provide fast access to data items that hold particular values.
• **Statistical Data :** It stores statistical information about the data in the database. This information is used by query processor to select efficient ways to execute query.

(DBLC) Database Life Cycle

The database life cycle (DBLC) defines the stages involved for implementing a database, starting with requirements analysis and ending with monitoring and modification. Furthermore, the DBLC never ends because database monitoring, modification, and maintenance are part of the life cycle, and these activities continue long after a database has been implemented. Put simply, the DBLC encompasses the lifetime of the database.

The five stages in the database life cycle are:

1. Requirements analysis
2. Logical design
3. Physical design
4. Implementation
5. Monitoring, modification, and maintenance



1.

The first three stages (1. Requirements analysis 2. Logical Design 3. Physical Design) are database-design

stages, which are listed below the diagram shown above.

I. **Requirements analysis**
Requirements Analysis is the first and most important stage in the *Database Life Cycle*.
It is the most labor-intensive for the database designer.
This stage involves assessing the informational needs of an organization so that a database can be designed to meet those needs.

II. Logical design
During the first part of Logical Design, a *conceptual model* is created based on the needs assessment performed in stage one. A conceptual model is typically an *entity-relationship (ER) diagram* that shows the tables, fields, and primary keys of the database, and how tables are related (linked) to one another.
The tables sketched in the ER diagram are then *normalized*. The normalization process resolves any problems associated with the database design, so that data can be accessed quickly and efficiently.

Database Design
conceptual model: A description of the structure of a database.

1. *entity-relationship (ER) diagram:* A diagram used during the design phase of database development to illustrate the organization of and relationships between data during database design.
2. *normalization:* The process of applying increasingly stringent rules to a relational database to correct any problems associated with poor design.

III. Physical Design
The Physical Design stage has only one purpose: to maximize database efficiency.
This means finding ways to speed up the performance of the RDBMS. Manipulating certain database design elements can speed up the two slowest operations in an RDBMS: retrieving data from and writing data to a database.

Developing a conceptual model:
In this course, you will complete the tasks associated with Requirements Analysis and the first part of Logical Design: developing a conceptual model.

Fourth Stage - Implementation
During the implementation stage of the DBLC, the tables developed in the ER diagram (and subsequently normalized) are converted into SQL statements. These SQL statements are then executed in the RDBMS to create a database. By this stage in the database life cycle, the *System Administrator* has installed and configured an RDBMS.

System administrator:
In this case the System administrator is the person responsible for administering a multi-user computer system. His duties range from setting up and configuring system components (i.e. an RDBMS) to performing maintenance procedures (for example, database backups) on the system.

Certain database design books consider converting an ER diagram into SQL statements to be the final task in the logical-design stage. According to such books, implementation is just a matter of feeding SQL statements into an RDBMS and populating the database with data. The difference is not especially important.

Fifth Stage - Monitoring, modification, and maintenance

A successfully implemented database must be carefully *monitored* to ensure that it is functioning properly and that it is secure from unauthorized access. The RDBMS usually provides utilities to help monitor database functionality and security.

Database *modification* involves adding and deleting records, importing data from other systems (as needed), and creating additional tables, user views, and other objects and tools. As an organization grows, its *information system* [Interrelated components (i.e., people, hardware, software, databases, telecommunications, policies, and procedures) that input, process, output, and store data to provide an organization with useful information.]must grow to remain useful.

In presenting this database design methodology, the design process is divided into three main phases: conceptual, logical, and physical database design.

Conceptual database design

The process of constructing a model of the data used in an enterprise, independent of *all* physical considerations.

The conceptual database design phase begins with the creation of a conceptual data model of the enterprise that is entirely independent of implementation details such as the target DBMS, application programs, programming languages, hardware platform, performance issues, or any other physical considerations.

Logical database design

The process of constructing a model of the data used in an enterprise based on a specific data model, but independent of a particular DBMS and other physical considerations.

The logical database design phase maps the conceptual data model on to a logical model, which is influenced by the data model for the target database (for example, the relational model). The logical data model is a source of information for the physical design phase, providing the physical database designer with a vehicle for making trade-offs that are very important to the design of an efficient database.

Physical database design

The process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes used to achieve efficient access to the data, and any associated integrity constraints and security measures.

The physical database design phase allows the designer to make decisions on how the database is to be implemented. Therefore, physical design is tailored to a specific DBMS. There is feedback between physical and logical design, because decisions taken during physical design for improving performance may affect the logical data model.