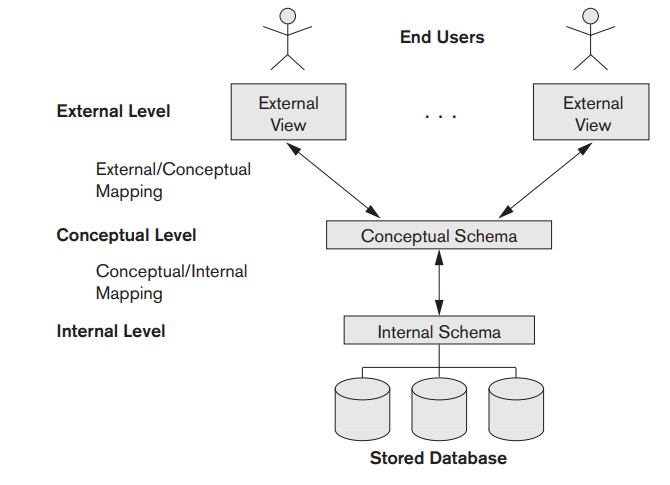
**2.2)Three-Schema Architecture and Data Independence**

* Three-schema architecture help to achieve and visualize these characteristics.

1. Use of a catalog to store the database description (schema) so as to make it self-describing.
2. Insulating to programs and data (program-data and program-operation independence)
3. Support of multiple user views.

* The goal of the three-schema architecture is to separate the user applications from the physical database.
* The three-schema architecture is a convenient tool with which the user can visualize the schema levels in a database system.
* In this architecture, schemas can be defined at the following three levels:

1. The **internal level** has an **internal schema**, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
2. The **conceptual level** has a **conceptual schema,** which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented.****
3. The **external or view level** includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

**Mapping**

* + - Notice that the three schemas are only descriptions of data; the actual data is stored at the physical level only. In the three-schema architecture, each user group refers to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user’s external view. The processes of transforming requests and results between levels are called mappings.

Why do we need mappings between the different schema levels?

* Mapping is used to transform the request and response between various database levels of architecture.
* Mapping is not good for small DBMS because it takes more time.
* In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.

Mapping between Views

The three levels of DBMS architecture don't exist independently of each other. There must be correspondence between the three levels i.e. how they actually correspond with each other. DBMS is responsible for correspondence between the three types of schema. This correspondence is called Mapping.

**There are basically two types of mapping in the database architecture:**

* Conceptual/ Internal Mapping
* External / Conceptual Mapping

**Conceptual/ Internal Mapping**

The Conceptual/ Internal Mapping lies between the conceptual level and the internal level. Its role is to define the correspondence between the records and fields of the conceptual level and files and data structures of the internal level.

**External/ Conceptual Mapping**

The external/Conceptual Mapping lies between the external level and the Conceptual level. Its role is to define the correspondence between a particular external and the conceptual view.

**Data Independence**

* + The data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.
  + There are Two Types of Data Independence:
    - **Logical data independence**
    - **Physical data independence**

**Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs.

* + We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).
  + Only the view definition and the mappings need to be changed in a DBMS that supports logical data independence. After the conceptual schema undergoes a logical reorganization, application programs that reference the external schema constructs must work as before.
  + Changes to constraints can be applied to the conceptual schema without affecting the external schemas or application programs.

**Physical data independence**  is the capacity to change the internal schema without having to change the conceptual schema.

* Changes to the internal schema may be needed because some physical files were reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update.
* If the same data as before remains in the database, we should not have to change the conceptual schema.
* providing an access path to improve retrieval speed of the query would be executed more efficiently by the DBMS by utilizing the new access path

**Note:-** Generally, physical data independence exists in most databases and file environments where physical details, such as the exact location of data on disk, and hardware details of storage encoding, placement, compression, splitting, merging of records, and so on are hidden from the user. Applications remain unaware of these details. On the other hand, **logical data independence is harder to achieve** because it allows structural and constraint changes without affecting application programs—a much stricter requirement.

**Note:-**  Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.

**2.3 Database Languages and Interfaces**

* The DBMS must provide appropriate languages and interfaces for each category of users.
* In this section we discuss the types of languages and interfaces provided by a DBMS and the user categories targeted by each interface.

**2.3.1 DBMS Languages**

* **Data Definition Language (DDL)**
* In many DBMSs where no strict separation of levels is maintained, one language, called the data definition language (DDL), is used by the DBA and by database designers to define both conceptul and internal schemas for the data base
* The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.
* In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the **conceptual schema** only.
* **Storage Definition Language(SDL)**
  + The storage definition language (SDL), is used to specify the **internal schema.**
  + In most relational DBMSs today, there is no specific language that performs the role of SDL.
  + Instead, the internal schema is specified by a combination of functions, parameters, and specifications related to storage of files. These permit the DBA staff to control indexing choices and mapping of data to storage.
* **View Definition Language(VDL)**
  + The view definition language (VDL), to specify **user views** and their mappings to the conceptual schema.
  + Note:- But in most DBMSs the DDL is used to define both conceptual and external schemas.
* **Data Manipulation Language(DML)**
  + Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database. Typical manipulations include retrieval, insertion, deletion, and modification of the data. The DBMS provides a set of operations or a language called the **data manipulation language (DML)** for these purposes.
    - * + There are two main types of DMLs:-

1. **High-level or nonprocedural DML**
2. **Low-level or procedural DML**
   * **High-level or nonprocedural DML**:- A high-level or nonprocedural DML can be used on its own to specify complex database operations concisely.
     + - High-level DMLs, such as SQL, can specify and retrieve many records in a single DML statement; therefore, they are called set-at-a-time or set-oriented DMLs.
       - A query in a high-level DML often specifies which data to retrieve rather than how to retrieve it; therefore, such languages are also called declarative.
   * **Low-level or procedural DML:-** This type of DML typically retrieves individual records or objects from the database and processes each separately.
     + - Therefore, it needs to use programming language constructs, such as looping, to retrieve and process each record from a set of records. Low-level DMLs are also called record-at-a-time DMLs because of this property.

**2.3.2 DBMS Interfaces**

User-friendly interfaces provided by a DBMS may include the following:

1. Menu-based Interfaces for Web Clients or Browsing.
2. Apps for Mobile Devices.
3. Forms-based Interfaces.
4. Graphical User Interfaces.
5. Natural Language Interfaces.
6. Keyword-based Database Search.
7. Speech Input and Output.
8. **Interfaces for Parametric Users:** Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly. For example, a teller is able to use single function keys to invoke routine and repetitive transactions such as account deposits or withdrawals, or balance inquiries. Systems analysts and programmers design and implement a special interface for each known class of naive users. Usually a small set of abbreviated commands is included, with the goal of minimizing the number of keystrokes required for each request.
9. **Interfaces for the DBA:** Most database systems contain privileged commands that can be used only by the DBA staff. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

7.1 Using High-Level Conceptual Data Models for Database Design

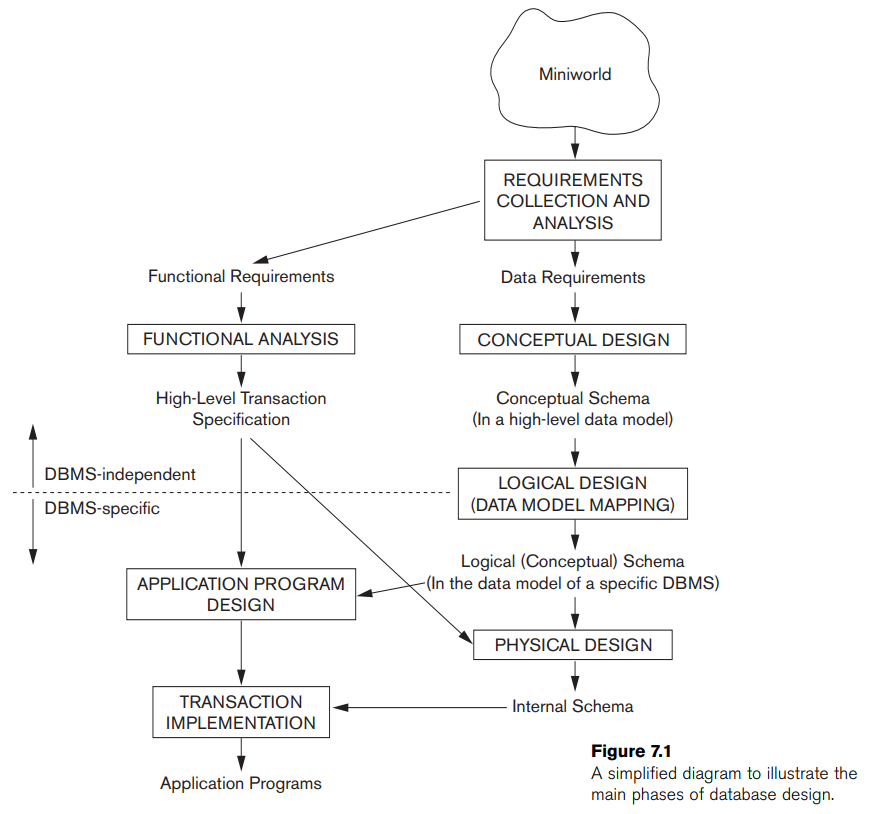
* The first step shown is requirements collection and analysis. During this step, the database designers interview prospective database users to understand and document their data requirements. The result of this step is a concisely written set of

users’ requirements.

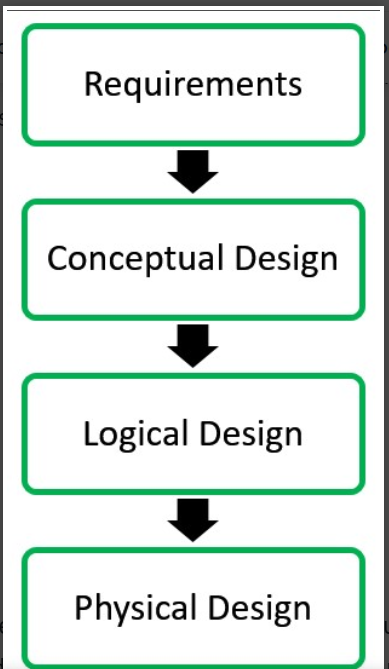
* In parallel with specifying the data requirements, it is useful to specify the known functional requirements of the application. These consist of the userdefined operations (or transactions) that will be applied to the database, including both retrievals and updates
* Once the requirements have been collected and analyzed, the next step is to create a conceptual schema for the database, using a high-level conceptual data model. This step is called conceptual design. The conceptual schema is a concise description of the data requirements of the users and includes detailed descriptions of the entity types, relationships, and constraints; these are expressed using the concepts provided by the high-level data model
* During or after the conceptual schema design, the basic data model operations can be used to specify the high-level user queries and operations identified during functional analysis. This also serves to confirm that the conceptual schema meets all the identified functional requirements. Modifications to the conceptual schema can be introduced if some functional requirements cannot be specified using the initial schema
* The next step in database design is the actual implementation of the database, using a commercial DBMS. Most current commercial DBMSs use an implementation data model—such as the relational or the object-relational database model—so the conceptual schema is transformed from the high-level data model into the implementation data model. This step is called logical design or data model mapping.
* The last step is the physical design phase, during which the internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files are specified.

**Phases of database design**

* Database designing for a real-world application starts from capturing the requirements to physical implementation using DBMS software which consists of following steps shown



-----------------------------------------------------------------------------------------------------------



**Conceptual Design:**The requirements of database are captured using high level conceptual data model. For Example, the ER model is used for the conceptual design of the database.

**Logical Design:**Logical Design represents data in the form of relational model. ER diagram produced in the conceptual design phase is used to convert the data into the Relational Model.

**Physical Design:** In physical design, data in relational model is implemented using commercial DBMS like Oracle, DB2.

**Advantages of DBMS**

DBMS helps in efficient organization of data in database which has following advantages over typical file system:

* **Minimized redundancy and data inconsistency:** Data is normalized in DBMS to minimize the redundancy which helps in keeping data consistent. For Example, student information can be kept at one place in DBMS and accessed by different users.This minimized redundancy is due to primary key and foreign keys
* **Simplified Data Access:** A user need only name of the relation not exact location to access data, so the process is very simple.
* **Multiple data views:** Different views of same data can be created to cater the needs of different users. For Example, faculty salary information can be hidden from student view of data but shown in admin view.
* **Data Security:** Only authorized users are allowed to access the data in DBMS. Also, data can be encrypted by DBMS which makes it secure.
* **Concurrent access to data:**Data can be accessed concurrently by different users at same time in DBMS.
* **Backup and Recovery mechanism:** DBMS backup and recovery mechanism helps to avoid data loss and data inconsistency in case of catastrophic failures.