

INTRODUCTION

Note: *This is a personal note only intended for the students of 1st Semester, MCA(2021), CIME. This note should not be published/distributed anywhere else except for the above said students.*

Data: known facts that can be recorded/ stored and have implicit meaning.

For example, consider the *names, phone numbers, and addresses* of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a hard drive, using a personal computer and software such as Microsoft Access or Excel.

Information: The processed Data is known as Information.

Database : The collection of related data with an implicit meaning is a database.

Ex: University database in which info is kept on students - such as *name, Course, Semester, birthdate*, etc.

AADHAR DataBase : *Name, Fathers name, Address, Phone Number, Biometric Information, Profile Pictures etc. are stored*

A database has the following implicit properties:

- A database represents some aspect of the real world, sometimes called the mini world or the universe of discourse (UoD).
- A database is a logically coherent collection of data with some inherent meaning. A random collection of data cannot correctly be referred to as a database.
- A database is designed, built, and populated with data for a specific purpose.

For example, a large commercial database is [Amazon.com](https://www.amazon.com). It contains huge data including millions of books, CDs, videos, DVDs, games, electronics, apparel, and other items. The database occupies over some terabytes (a terabyte is 10^{12} bytes) and is stored on different computers (called servers). Millions of visitors access [Amazon.com](https://www.amazon.com) each day and use the database to make purchases. The database is continually updated as new books and other items are added to the inventory and stock quantities are updated as purchases are transacted. Hundreds of people are responsible for keeping the Amazon database up-to-date.

Database management system (DBMS):

A DBMS is a collection of programs that enables users to create and maintain a database.

The DBMS is a general-purpose software system that facilitates the processes of **defining, constructing, manipulating, and sharing** databases among various users and applications.

Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database.

The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called meta-data.

Constructing the database is the process of storing the data on some storage medium that is controlled by the DBMS.

Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the database, and generating reports from the data.

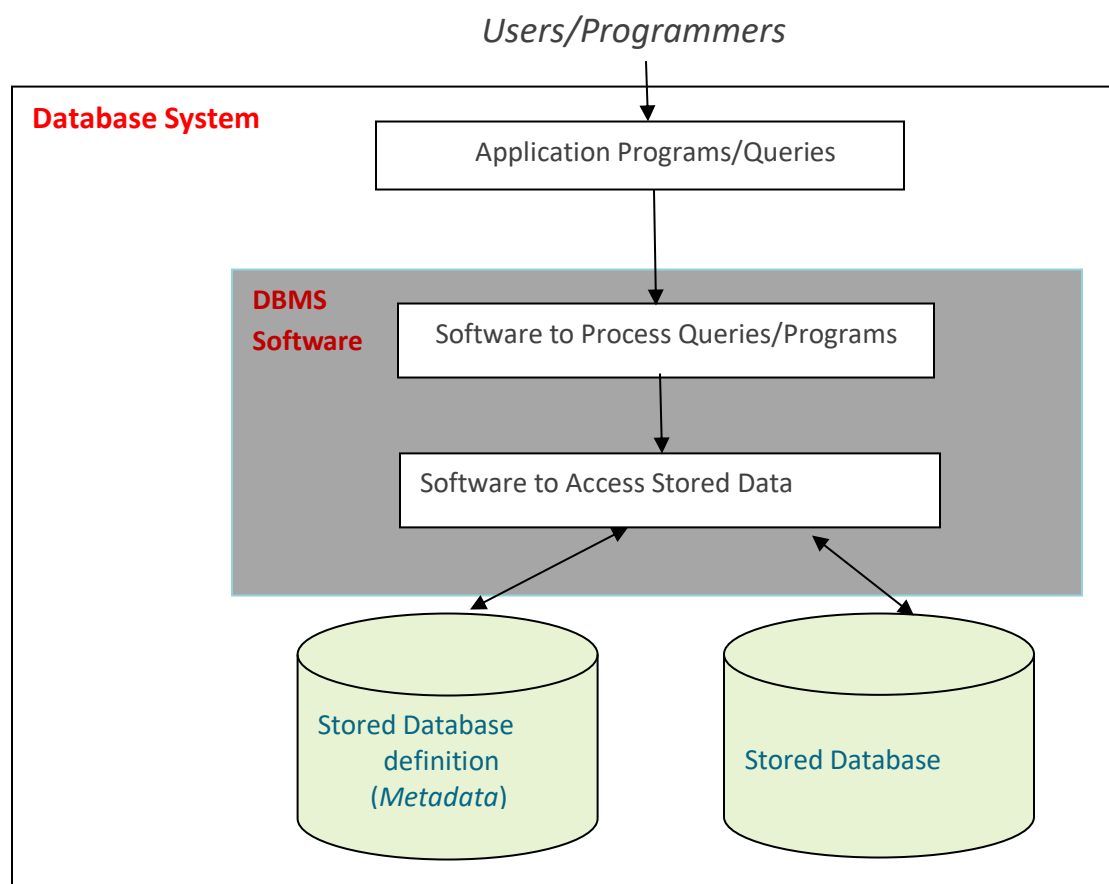
Sharing a database allows multiple users and programs to access the database simultaneously.

Other important functions provided by the DBMS **include protecting the database and maintaining it over a long period of time.**

Protection includes **system protection** against hardware or software malfunction (or crashes) and **security protection** against unauthorized or malicious access.

A typical large database may have a life cycle of many years, so the DBMS must be able to **maintain** the database system by allowing the system to evolve as requirements change over time.

Representation of a Simplified Database System Environment is as follows.



Note: *Database System = DBMS + DATA*


Example: Let us consider a Student database which contains records of each student's RollNo, Name, Gender, date of birth, the course in which he/she admits

and the secured Grade. So, each student is characterized by his/her Rollno, Name, Gender, dob, course and the Grade.

i.e. STUDENT (*RollNo, Name, dob, Gender, Course, Grade*)

To input values or records for students we have to define it. We have to define different kinds of data elements to be stored in each record i.e. what kind of value a particular field of a record can store. That means we have to define the data type for each data element of a record as follows.

STUDENT (*RollNo* : string or integer
 Name : string,
 Dob : date
 Gender : char,
 Course : string)



Defining the Structure of a database

(**Note:** *For database programming the data types can be chosen based on the Query language syntax*)

The following figure represents some sample data in the STUDENT data base

STUDENT

RollNo	Name	Dob	Gender	Course	Grade
20201	Ramesh	12-APR-1995	M	MCA	A
20202	Suresh	13-MAY-1996	M	MBA	E
20203	Rakhi	09-JAN-1996	F	MCA	B
20204	Sima	13-OCT-1995	F	MBA	B

Database manipulation involves querying and updating.

Examples of queries are as follows:

- List the students of MCA
- List all the students who secured grade 'A ' or above.

Example of updates:

- change the grade of the student having Roll number-20203 to 'A '

These queries and updates must be specified precisely in the query language of the DBMS before they can be processed.

Characteristics of Database Systems/Advantages of Database Systems over older File Processing Systems/ Purpose of :

(i) Self-describing nature of Database Systems:

The database system not only contains the database itself but also contains a complete definition or description of the database structure and constraints. This definition is stored in the **DBMS catalog** which contains information such as *the structure of each database file, the type and storage format of each data item, and various constraints on the data*. The information stored in the catalog is called **Meta Data** or **Data Dictionary**.

A general-purpose DBMS software package is not written for a specific database application. Therefore, it must refer to the catalog to know the structure of the files in a specific database, such as the type and format of data it will access.

In traditional file processing, data definition is typically part of the application programs themselves. Hence, these programs are confined to work with only one specific file (database application), whose structure is declared in the application programs. On the other hand, DBMS software can access diverse databases by extracting the database definitions from the catalog itself.

(ii) Insulation between Programs and Data/Program Data Independence :

In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file.

On the other hand, DBMS access programs do not require such changes in most cases. The structure of data files is stored in the DBMS catalog separately from the access programs. Whenever changes are made to the structure of a data base (*e.g. by adding or removing some fields*), DBMS program refers to the catalog and the new structure will be accessed and used. This property is called as **program-data independence**.

(iii) Support of Multiple Views of the Data:

A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.

A database typically has many users, each of whom may require a different perspective or view of the database. A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

(iv) Sharing of Data and Multiuser Transaction Processing:

A multiuser DBMS allows multiple users to access the database at the same time. The main role of multiuser DBMS software is to ensure that concurrent transactions are operating correctly and efficiently. The DBMS includes a **concurrency control sub module** to ensure that several users trying to update the same data in a controlled manner so that the result of the updates is always correct.

(v) Controlling Redundancy :

Data Redundancy is very prevalent in traditional file processing system. Because, each user group may keep the copy of same file to handle data processing.

Redundancy means storing the same data multiple times in different places. Redundancy leads to several problems which are as follows

- **Duplication of Effort** *i.e.*, entering data for the same file multiple times.
- **Wastage of Storage space** *i.e.*, storage space is wasted when the same data is stored repeatedly, and this problem may be serious for large databases.
- **Inconsistent Data** *i.e.*, The files that represent the same data may become inconsistent. This is because an update is applied to some of the files but not to others. Even if an update is applied to all the appropriate files, the data concerning the file still be inconsistent because the updates are applied independently by each user group.

DBMS provides data normalization mechanisms which controls redundancy and ensures data consistency.

Some other features provided by Database Systems are

(vi) Restricting Unauthorized Access

(vii) Providing Storage Structures and Search Techniques for Efficient Query Processing

(viii) Providing Backup and Recovery

(ix) Providing Multiple User Interfaces

(x) Enforcing Integrity Constraints (Data Integrity)

Database Architecture

Data model explicitly determines the structure of data.

*A **data model** defines the logical design and structure of a database and in which format the data are stored, accessed and updated in a database. Data model supports the concept data abstraction in a database system.*

Types of data model:

- **High-level or conceptual data models**
- **Representational or implementation data models (Logical data model)**
- **Low-level or physical data models**

High-level or conceptual data models:

A Conceptual data model is an abstract data model. It uses concepts such as entities, attributes, and relationships.

An entity represents a real-world object or concept, such as an employee or a project from the real-world concepts that is described in the database.

An attribute represents some property that further describes an entity, such as the employee's id, name, job, salary etc.

A relationship among two or more entities represents an association among the entities, for example, a works-on relationship between an EMPLOYEE and a PROJECT.

e.g., Entity-Relationship model

Representational or implementation data models (Logical data model):

It adds detail to the conceptual model. It basically represents record types (datatypes of each attributes), constraints and relationship among data in a database.

e.g., Relational Model.

Low-level or physical data models:

It basically adds **storage structure and access path** to logical data model. It describes how data is stored by representing information such as record formats, record orderings, and access paths.

An **access path** is a structure that makes the search for particular database records efficient. An **index** is an example of an access path that allows direct access to data using an index term or a keyword.

Database Schema:

The description of a database is called the **database schema**. A schema represents the names of data items, their formats or data types, and some types of constraints.

A data base schema can be represented by any of the following ways.

- EMPLOYEE(eid, ename, dob, degn, salary, dno)
- EMPLOYEE(eid : number/String ,
 ename : string,
 dob : date ,
 degn: string,
 salary : number,
 deptno : number)

- EMPLOYEE

<u>eid</u>	ename	dob	degn	salary	deptno
------------	-------	-----	------	--------	--------

Key field are underlined

Data base state:

The data in the database at a particular moment in time is called a *database state* or *snapshot*. It is also called the *database instance* or *instance of a data base*.

Valid State— It is a state that satisfies the structure and constraints specified in the schema.

Note: The **schema** is sometimes called the *intension*, and a **database state** is called an *extension* of the schema.

Three-Schema Architecture /ANSI Data base System Architecture

The goal of the three-schema architecture is to achieve

- use of a catalog to store the database description (schema) so as to make it self-describing
- insulation of programs and data (program-data independence)
- support of multiple user views.

In this architecture, schemas can be defined at the following three levels:

- (i) **The internal schema**
- (ii) **The conceptual schema**
- (iii) **The external schema**

Internal Schema:

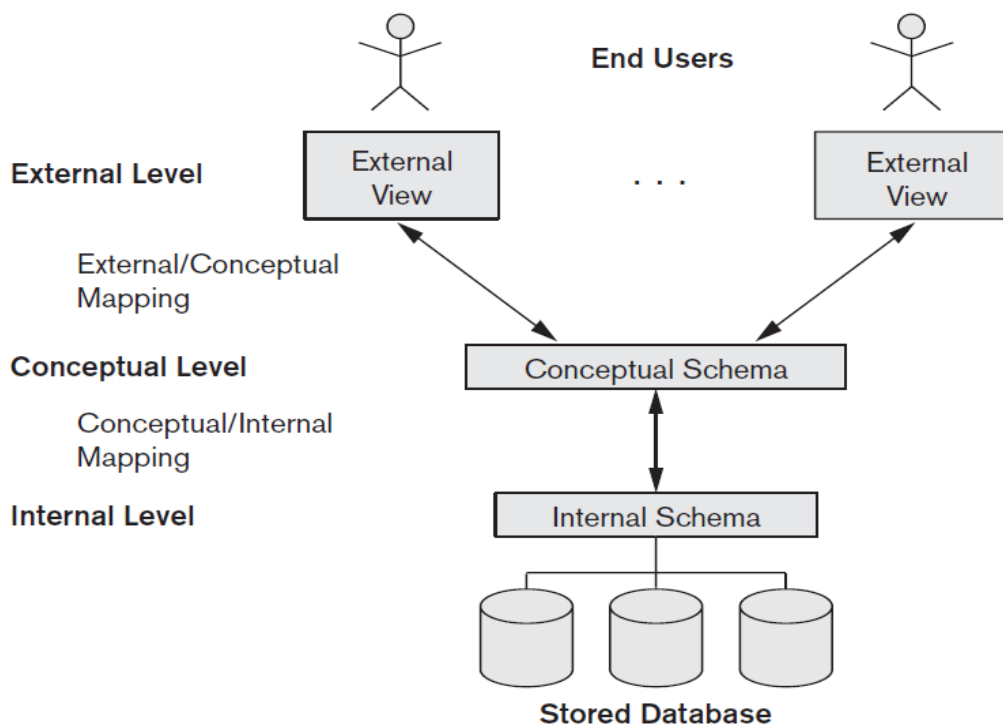
- The internal level has an internal schema, which describes the physical storage structure of the database.
- The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

Conceptual Schema:

- The conceptual level has a conceptual schema, which describes the structure of the whole database.
- It describes entities, data types, relationships, user operations, and constraints.
- It hides details of the physical data storage structures.
- Usually, a representational data model is used to describe the conceptual schema.

External Schema:

- 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.



Mappings:

In a DBMS based on 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.

Data Independence:

Data Independence 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.

We can define two types of data independence:

i) Logical data independence:

- It 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). In these cases, the external schemas should not be affected.

ii) Physical Data Independence:

- Physical data independence is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well.

- 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.

Database Users:

Database Users are the people who are involved in the design, use and maintenance of a large database with hundreds of users.

The database users are of the following categories

i) Database Administer:

DBA is a person or a group of people in an organization whose main task is to administrator, oversee and manage the database itself, DBMS and the related software.

The responsibilities of DBA are as follows

- ***Schema Definition:***
 - The DBA definition the logical Schema of the database. A Schema refers to the overall logical structure of the database.
 - According to this schema, database will be developed to store required data for an organization.
- ***Storage Structure and Access Method Definition:***
 - The DBA decides how the data is to be represented in the stored database and decide the access path for efficient data access.

- ***Ensuring Data Security (i.e. Approving Data Access):***
 - The DBA determines which user needs access to which part of the database.
 - According to this, various types of authorizations are granted to different users.
- ***Schema modification and performance monitoring:***
 - The DBA can change the conceptual schema and also the physical schema according to the requirement of the organization.
 - The DBA monitors performance of the system. The DBA ensures that better performance is maintained by making changes in physical or logical schema if required.
- ***Assisting Application Programmers:***
 - The DBA provides assistance to application programmers to develop application programs where DBMS is used for backend applications.
- ***Backup and Recovery:***
 - Database should not be lost or damaged.
 - The DBA ensures this periodically backing up the database on magnetic tapes or remote servers.
 - In case of failure, such as virus attack database is recovered from this backup.

ii) **Database Designers:**

- Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. These tasks are

mostly undertaken before the database is actually implemented and populated with data.

- It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these users' requirements.
- Database designers typically interact with each potential group of users and develop views of the database that meet the data and processing requirements of these groups. Each view is then analyzed and integrated with the views of other user groups. The final database design must be capable of supporting the requirements of all user groups.

iii) End Users:

End users are the people whose jobs require access to the database for querying, updating, and generating reports from the database.

The end users can be categorized as **Casual end users, Naive or parametric end users, Sophisticated end users, and Standalone users.**

- **Casual end users** occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.
- **Naive or parametric end users** make up a sizable portion of database end-users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates.

The tasks that such **Naive users** perform are:

- *Bank tellers* check account balances and post withdrawals and deposits.
- *Reservation agents* for airlines, hotels, and car rental companies check availability for a given request and make reservations.
- *Employees at receiving stations for shipping companies* enter package identifications via bar codes and descriptive information through buttons to update a central database of received and in-transit packages.
- **Sophisticated end users** include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements.
- **Standalone users** maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces. An example is the user of a tax package that stores a variety of personal financial data for tax purposes.

iv) **System Analysts and Application Programmers (Software Engineers):**

- **System analysts:** They determine the requirements of end users, especially naive and parametric end users, and develop requirement specifications for standard canned transactions that meet these requirements.
- **Application programmers:** They implement these specifications as programs; then they test, debug, document, and maintain these canned transactions.

Applications of Data Base:

- **Scientific applications** that store large amounts of data resulting from scientific experiments in areas such as high-energy physics, the mapping of the human genome, and the discovery of protein structures.
- **Storage and retrieval of images**, including scanned news or personal photographs, satellite photographic images, and images from medical procedure such as x-rays and MRIs (magnetic resonance imaging).
- **Storage and retrieval of videos**, such as movies, and video clips from news or personal digital cameras.
- **Data mining applications** that analyze large amounts of data searching for the occurrences of specific patterns or relationships, and for identifying unusual patterns in areas such as credit card usage.
- **Spatial applications** that store spatial locations of data, such as weather information, maps used in geographical information systems, and in automobile navigational systems.
- **Time series applications** that store information such as economic data at regular points in time, such as daily sales and monthly gross national product figures.

Database Languages

DBMS provide appropriate languages which is known as Database Languages for each category of users to create and maintain the data base

Types of Database Languages

- **Data Definition Language (DDL)**
- **Storage Definition Languages (SDL)**
- **View Definition Languages (VDL)**
- **Data Manipulation Language (DML)**
- **Data Control Language (DCL)**
- **Transaction Control Language (TCL)**

Data Definition Language (DDL):

- DDL is used to define the conceptual Schema.
- Data Definition Language (DDL) statements are used to specify the database structure or schema.
- It specifies entities, attributes, constraints and relationships

After the compilation of DDL statements, System Catalog (or Data Dictionary) is created which stores information of metadata like the number of tables and schemas, their names, indexes, columns in each table and their types, constraints, etc.

DDL Statements are as follows

- **CREATE** - used to create objects in the database
- **ALTER** - used to alter the structure of the database
- **DROP** - used to delete objects from the database
- **TRUNCATE** - used to remove all records from a table, including all spaces allocated for the records are removed
- **COMMENT** - used to add comments to the data dictionary

- **RENAME** - used to rename an object

Storage Definition Language (SDL):

- SDL is used to define Internal Schema. It is used for defining physical storage structures and access methods to fine-tune the performance of the database system.

View Definition Language (VDL):

- Used to define external schema i.e., it is used to specify user views and their mappings to the conceptual schema

Data Manipulation Language (DML):

- DML provides a set of operations to support the basic data manipulation operations on the data held in the database.
- Typical manipulations include retrieval, insertion, deletion, and modification of the data present in a database.

DML Statements:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used execute a **routine** (a procedure or function, or a procedure or function defined within a type or package) from within SQL.
- **Explain Plan:** It determines the execution plan that Oracle Database follows to execute a specified SQL statement.
- **Lock Table:** It controls concurrency.

There are two main types of DMLs

- **High level or Non-Procedural DML**
- **Low level or Procedural DML**

High level or Non-Procedural DML:

- Non-Procedural DML can be used on its own to specify database operations
- A query in a high-level DML often specifies *which* data to retrieve without specifying *how* to retrieve it; therefore, such languages are also called **declarative languages**.
- These DML statements either to be entered through terminal or to be embedded in a general-purpose programming language.
- Non-Procedural DMLs can specify and retrieve multiple records in a single DML statement; therefore, they are called **set-at-a-time** or **set-oriented** DMLs.
- Example of **Non-Procedural DML** is **SQL**

Low level or Procedural DML

- A **low level** or **procedural DML** specifies *what* data is needed and *how* to get it.
- A **low level** or **procedural DML** must be embedded in a general-purpose programming language.
- This type of DML typically retrieves single records from the database and processes each record separately. Thus, Low-level DMLs are also called **record-at-a-time** DMLs.
- Example of **Procedural DML** is **PL/SQL**

Note: Whenever DML commands, whether high level or low level, are embedded in a general-purpose programming language, that language is called the **host language** and the DML is called the **data sublanguage**.

Data Control Language (DCL)

- DCL is used for granting and revoking user access on a database
- DCL Statements include GRANT and REVOKE commands
 - **GRANT**- gives user's access privileges to database.
 - **REVOKE**- withdraw user's access privileges given by using the GRANT command.

Transaction Control Language (TCL)

- Transaction Control Language commands are used to manage transactions in the database.
- These are used to manage the changes made by DML-statements.
- It also allows statements to be grouped together into logical transactions.
 - **COMMIT**: Commit command is used to permanently save any transaction into the database.
 - **ROLLBACK**: This command restores the database to last committed state. It is also used with savepoint command to jump to a savepoint a transaction.
 - **SAVEPOINT**: Savepoint command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

EF Codd Rules

Dr. Edgar Frank Codd was a computer scientist while working for IBM, he invented the relational model for database management known as Relational Database Management System (RDBMS).

Codd proposed thirteen rules (numbered zero to twelve) and said that if a Database Management System meets these rules, it can be called as a Relational Database Management System.

These rules are called as Codd's 12 rules.

Rule Zero: The Foundation Rule

- For a system to qualify as a relational database management system (RDBMS), that system must be able to manage the database entirely through its relational capabilities

The other 12 rules derive from this rule. These rules are as follows:

Rule 1 : The information rule:

- All information in a relational database is represented explicitly at the logical level and in exactly one way – by values in tables.

Rule 2 : The guaranteed access rule:

- Each and every data (atomic value) in a relational database is guaranteed to be accessible by specifying the combination of table name, primary key value and column name.

Rule 3 : Systematic treatment of NULL values:

- NULL values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type.

Rule 4 : Active online catalog based on the relational model:

- The database description is represented at the logical level in the same way as ordinary data, so that authorized users must be able to access the database description (catalog) using the same relational language that they use to access the regular data.

Rule 5 : The comprehensive data sub language rule:

- The system must support at least one relational language that
 1. Has a linear syntax
 2. Can be used both interactively and within application programs,
 3. Supports data definition operations (including view definitions), data manipulation operations (insert, update, delete as well as retrieval), security and integrity constraints, and transaction management operations (commit, and rollback).

Rule 6 : The view updating rule:

- All views those can be updated theoretically, must be updated by the system.

Rule 7 : High-level insert, update, and delete:

- The system must support set-at-a-time insert, update, and delete operators.
- This means that data can be retrieved from a relational database in sets of data from multiple rows and/or multiple tables.
- This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

Rule 8 : Physical data independence:

- Changes to the physical level (how the data is stored and access methods) must not require a change to logical structure and application programs.

Rule 9 : Logical data independence:

- Changes to the logical level (structures, constraints etc) must not require a change to the application program or user views.

Rule 10 : Integrity independence:

- Integrity constraints specific to a particular relational database must be defined by the relational database language and stored in the catalog but not in the application program.

Rule 11 : Distribution independence:

- The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only.

Rule 12: The non-subversion rule:

- If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher-level relational language (multiple-records-at-a-time).

Practice Questions

Objective Questions:

- Data Redundancy leads to data inconsistency, justify.
- Define DBMS.
- Define database schema and database state.
- What is data dictionary.
- What do you mean by program data independence
- Write the responsibilities of DBA.
- What is access path.

Descriptive Questions:

Q: Discuss the advantages of Database System over Traditional File Systems

Q: Discuss different types of database users and their roles.

Q: What is ANSI Database architecture? Discuss Data Independence.

Q: What are different database languages. Explain each of these with examples.

Q: What is Significance of Codd's rule. Write all thirteen proposed Codd's rule.

References:

1. RamezElmasri and Shamkant B. Navathe, "Fundamentals of Database Systems", Pearson Education Inc., New Delhi.
2. Raghu Ramakrishnan, Johannes Gehrke, "Database Management Systems", McGraw-Hill Education (India), New Delhi.
3. www.nptel.ac.in