CHAPTER - 1

INTRODUCTION

1.1 Concept and Applications

1.1.1 Data

Data are the raw facts and figure. In the context of databases, data refers to all the single items that are stored in a database, either individually or as a set.

Example:

In a banking application, data is the whole collection of bank account numbers, bank customer's names, addresses, ages, bank transactions and so on.

1.1.2 Information

Information is data that has been processed, organized, or interpreted to make it meaningful, relevant, and useful for decision-making or understanding.

Example:

Bank statement

Differences between Data and Information

Data refers to raw facts or figures that are collected and stored in the system, whereas information refers to the processed and analyzed data that provides meaning and context to the raw data.

For example, in a banking system, the data might include:

- Account numbers
- Transaction dates
- Amounts debited or credited
- Account holders' names
- Account types

These data can be stored in a database, but on its own, they don't provide any meaningful information. However, when these

data are processed and analyzed, they can be transformed into useful information.

For example:

- An account statement that shows a customer's transaction history and current balance.
- A monthly report that shows the total deposits and withdrawals for each account type.
- A fraud detection system that uses transaction data to flag suspicious activity.

In each of these cases, the raw data is processed and analyzed to provide meaningful information that can be used to make decisions or take action.

1.1.3 Database

A database is a collection of related data necessary to manage an organization. A database is usually controlled by a database management system (DBMS).

Example:

A company can have various details of employees, such as name, empID, email, blood group, salary, and so on. All these details can be stored in a database with the name "Employee" in a structured format such as tables, hierarchy, etc.

Types of databases are:

1. Relational database
2. Cloud database
3. NoSQL database
4. Graph database

1.1.4 Database Management System (DBMS)

Database management system (DBMS) is a system software that is used to access, create, and manage databases. With the help of DBMS, we can easily create, retrieve and update data in databases.

Type of DBMS are:

1. Object Oriented DBMS
2. Network DBMS
3. Relational DBMS
4. Hierarchical DBMS

Example of DBMS are:

MySQL, PostgreSQL, Microsoft Access, SQL Server, FileMaker, Oracle, RDBMS, dBASE, Clipper, and FoxPro.

Relation between Database and DBMS

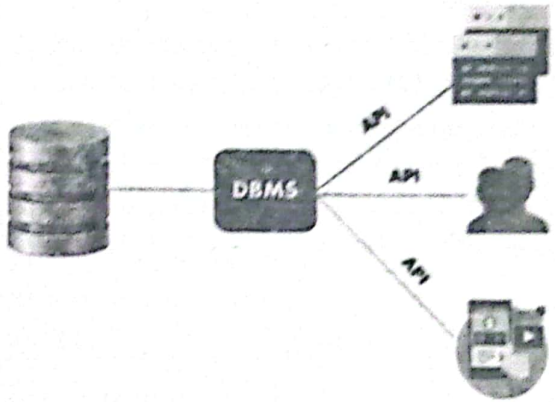

Figure: Relation between database and DBMS

In figure, API means *Application Programming Interface*; it is a software intermediary that allows two applications to talk to each other.

1.1.5 Database Application

Following are the applications of database:

- **Banking:** Transactions
- **Airlines:** Reservations, schedules
- **Universities:** Registration, grades
- **Sales:** Customers, products, purchases
- **Online retailers:** Order tracking, customized recommendations

- **Human resources:** Employee records, salaries, tax deductions
- **Telecommunication:** Records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards

1.2 Objectives and Evolution

1.2.1 Objectives of DBMS

Following are the objectives of DBMS:

- A database should provide efficient storage, update, and retrieval of data.
- A database should be reliable – the stored data should have high integrity and promote user trust in that data.
- A database should be adaptable and scalable to new and unforeseen requirements and applications.
- A database should identify the existence of common data and avoid duplicate recording.

1.2.2 Evolution of DBMS

The history and evolution of DBMS is given below:

- **1960:** Charles Bachman designed first DBMS system
- **1970:** Codd introduced IBM'S Information Management System (IMS)
- **1976:** Peter Chen coined and defined the entity-relationship model also known as the ER model
- **1980:** Relational model becomes a widely accepted database component
- **1985:** Object-oriented DBMS develops.
- **1990s:** Incorporation of object-orientation in relational DBMS.
- **1991:** Microsoft ships MS access, a personal DBMS and that displaces all other personal DBMS products.
- **1995:** First Internet database applications
- **1997:** XML applied to database processing. Many vendors begin to integrate XML into DBMS products.

1.3 Needs of DBMS

A file system is a process that manages how and where data on a storage disk, typically a hard disk drive (HDD), is stored, accessed and managed. It is inbuilt in OS.

Example:

NTFS (New Technology File System), EXT (Extended File System).

DBMS is needed because of the following drawbacks of file system over DBMS:

1. **Data redundancy and inconsistency problem**

   Data redundancy refers to the duplication of data. Data redundancy often leads to data inconsistency, higher storage costs and poor access time.

   Data inconsistency means different and conflicting versions of same data occur at different places.

2. **Difficult in accessing data**

   We need to write a new program to carry out each new task in file system.

   Example:

   To obtain few KB of data from a huge TB data, we need to process all the huge data first which makes difficulty in accessing data in file system.

3. **Data isolation problems**

   File system consists of unstructured data i.e., multiple files and formats which leads to data isolation problem.

4. **Integrity problems**

   It is hard to add new constraints or change existing ones in file system. The term data integrity refers to the accuracy and consistency of data. For example, a user could accidentally try to enter a phone number into a date field. If the system enforces data integrity, it will prevent the user from making these mistakes.

5. **Atomicity problems**

   Failures may leave database in an inconsistent state with partial updates carried out. For example: Transfer of funds from one account to another should either complete or not happen at all. It is difficult to achieve atomicity in file processing systems.

6. **Concurrent access anomalies**

   Uncontrolled concurrent accesses in file processing system can lead to inconsistencies.

7. **Security problems**

   Data should be secured from unauthorized access. For example a student in a college should not be able to see the payroll details of the teachers, such kind of security constraints are difficult to apply in file processing systems.

Differences between file system and DBMS:

1. A file system is a software that manages and organizes the files in a storage medium, whereas DBMS is a software application that is used for accessing, creating, and managing databases.
2. The file system doesn't have a crash recovery mechanism on the other hand, DBMS provides a crash recovery mechanism.
3. Data inconsistency is higher in the file system. On the contrary data inconsistency is low in a database management system.
4. File system does not provide support for complicated transactions, while in the DBMS system, it is easy to implement complicated transactions using SQL.
5. File system does not offer concurrency, whereas DBMS provides a concurrency facility.
6. File System are often single user oriented, whereas DBMS are often multiuser oriented.

1.4 Data Abstraction

Database systems are made up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.

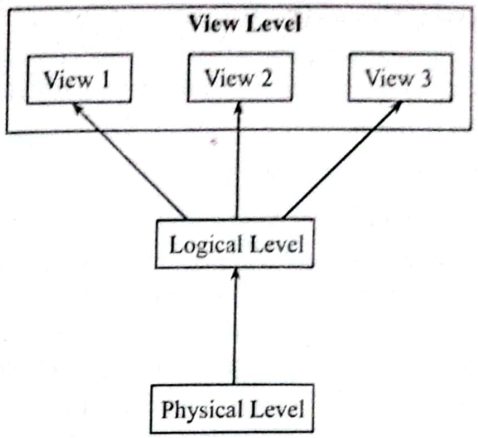

Figure: Levels of data abstraction

There are three levels of abstraction:

1. **Physical Level (Internal Level)**

   This is the lowest level of data abstraction. It describes how data is actually stored in database. We can get the complex data structure details at this level. The access methods like sequential or random access and file organization methods like B + trees, hashing used for the same. Suppose we need to store the details of an employee, blocks of storage and the amount of memory used for these purposes is kept hidden from the user

2. **Logical Level (Conceptual Level)**

   This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database and the relationship among the data.

   Example:

```
type customer = record
    name : string;
    street : string;
    city : integer;
end;
```

3. **View Level (External Level)**

This is the highest level of data abstraction. This level describes the user interaction with database system.

Example:

Interaction of user with system using GUI.

Example of Data abstraction

Let's say we are storing customer information in a customer table. At physical level, these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.

At logical level, these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

At view level, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

1.5 Data Independence

Data independence can be explained using the three-schema architecture. Data independence refers to characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level. The interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Importance of data modeling

- Data independence helps us to improve the quality of the data
- Database system maintenance becomes affordable
- Enforcement of standards and improvement in database security
- We 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 us to improve state which is undamaged or undivided
- Database incompatibility is vastly reduced.

There are two types of data independence:

1. **Physical Data Independence**

It indicates that the physical storages structures or devices could be changed without affecting the conceptual schema. It is the ability to modify the physical schema without changing the logical schema.

Benefits of physical data independence

a. **Performance Optimization**

To increase performance without affecting the application programs, DBMS administrators might rearrange or restructure the physical storage of data. For instance, they can alter partition tables, indexes, or storage formats to improve the speed of data processing and retrieval.

b. **Platform Independence**

It is possible to move data between various hardware or operating systems thanks to physical data independence. The DBMS can migrate to new platforms or adapt to new storage technologies without requiring changes to the programs that use the database.

c. **Data Security**

Without interfering with the functioning of the current applications, changing the physical storage structure can help impose security features like encryption, access control, or data segregation.

2. **Logical Data Independence**

It is the capacity to change the conceptual schema without having to change the external schemas and their application programs. We may change the conceptual schema to expand

the database (by adding a record type or data item), or to reduce the database (by removing a record type or data item).

Benefits of logical data independence

a. Evolution of Data Models

Data models may need to be improved, expanded, or modified over time to meet shifting business needs. Modifications to the database's structure are possible with logical data independence without having an influence on the programs that use it. This adaptability makes it simpler to meet changing corporate needs.

b. Database Restructuring

The database schema can be reorganized to improve data integrity, normalization, or data modeling with the help of logical data independence. These changes can be implemented without impairing current applications or resulting in inconsistent data.

c. Data Integration

The seamless integration of additional data sources or databases into an existing system is supported by logical data independence. It permits the creation of new tables, relationships, or attributes without interfering with the operation of any already-running applications that are using the database.

Differences between physical and logical data independence

Logical Data Independence	Physical Data Independence
Ability to change the logical (conceptual) schema without changing the external schema (user view) is called logical data independence.	Ability to change the physical schema without changing the logical schema is called physical data independence.
Modification is necessary whenever the logical structure of the database is altered.	Modification is rarely done, as it is done to improve the performance of the database.
It is more difficult to achieve.	It is relatively easy to achieve.
Example: Consider two users A & B, both are selecting the fields "employee_number". If user B adds new column "salary" to the same table, it will not affect the user view level.	**Example:** Changing the storage drive of the data base from "C" drive to "D" drive will not affect the conceptual and view level as the new changes are absorbed by the mapping technique.

1.6 Schema and Instances

1.6.1 Schema

Design of a database including tables, fields, relationships, views and other elements is called the schema. Database schema is analogous to the variable declarations (along with associated type).

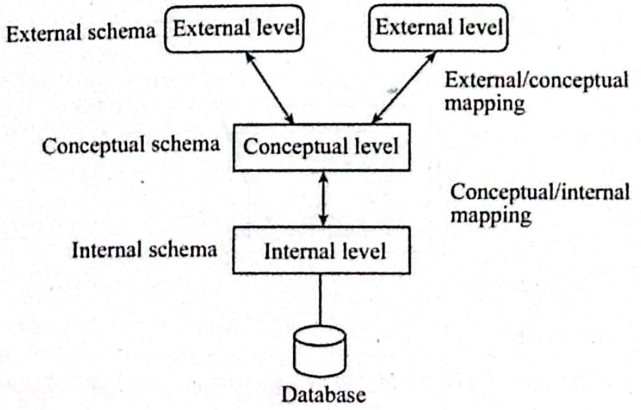

Figure: Various levels of schema

Types of schema:

1. Physical Schema

The design of a database at physical level is called physical schema; how the data stored in blocks of storage is described at this level.

2. **Logical Schema**

Design of database at logical level is called logical schema; programmers and database administrators work at this level.

3. **View Schema**

Design of database at view level is called view schema (also called external schema); this generally describes end user interaction with database systems.

Example:

In the following diagram, we have a schema that shows the relationship between three tables: Course, Student and Section. The diagram only shows the design of the database, it doesn't show the data present in those tables. Schema is only a structural view (design) of a database.

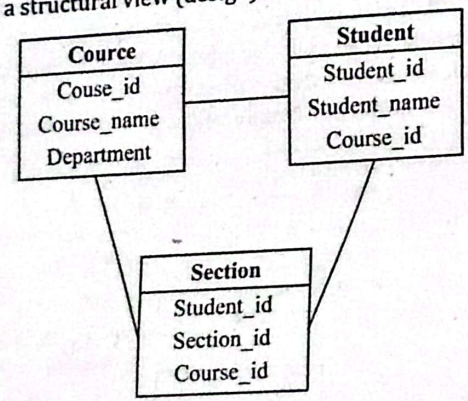

Figure: Schema of 3 tables: course, student and section

1.6.2 Instance

The data stored in database at a particular moment of time is called instance of database. Database instance is analogous to the value of variables in a program at a point in time.

Example:

Let's say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. Let's say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table. In short, at a particular moment the data stored in database is called the instance, that changes over time when we add or delete data from the database.

Differences between schema and instance:

Schema	Instance
• It is the overall description of the database.	• It is the collection of information stored in a database at a particular moment.
• Schema is same for whole database.	• Data in instances can be changed using addition, deletion, updating.
• Defines the basic structure of the database i.e., how the data will be stored in the database.	• It is the set of Information stored at a particular time.

Consider a database for a social media platform. The schema might include tables for users, posts, comments, and likes, as well as columns for the username, email address, and password of each user. The schema would also define the relationships between the tables, such as the fact that each user can make many posts and receive many comments and likes. The instance, in this case, would be the actual data stored in the tables at a given moment, such as a list of all the posts made by a particular user and the associated comments and likes.

1.7 Concept of DDL, DML and DCL

Database languages are the language that allow users to communicate with database. The main purpose of these language is to define the schema and then store, access and manipulate the data. This can be implemented with SQL (Structured Query Language). A statement requesting the retrieval of information is called query statement.

1.7.1 Data Definition Language (DDL)

DDL deals with the description of the database schema where it creates and modifies the structure of the database objects on the database.

DDL compiler generates a set of table templates stored in a data dictionary. Data dictionary contains metadata (i.e., data about data). DDL includes:

i. Database schema
ii. Integrity constraints
iii. Authorization (who can access what)

Example:
CREATE, ALTER, DROP, TRUNCATE, COMMENT, RENAME

1.7.2 Data Manipulation Language (DML)

A data manipulation language (DML) is a language for accessing and manipulating the data organized by the appropriate data model. DML is also known as query language. It is used to retrieve, store, modify, delete, insert and update data in database.

Example:
SELECT, INSERT, UPDATE, DELETE

There are two classes of DML:

1. **Procedural:** Here user specifies what data is required and how to get those data.

   Example:
   C, COBOL, BASIC, ALGOL, Pascal, etc.

2. **Nonprocedural (declarative):** Here user specifies what data is required without specifying how to get those data. Declarative DMLs are usually easier to learn and use than procedural DMLs.

   Example:
   SQL, Relational Calculus, LISP, ProLog.

1.7.3 Transaction Control Language (TCL)

Transaction control language (TCL) is used to manage different transactions occurring within a database.

Example:
COMMIT, ROLLBACK, SAVE TRANSACTION

1.8 Database Manager and Users

1.8.1 Database Manager (Administrator)

A person who has central control over the database management system is called database manager. Database manager is responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources and controlling their use and monitoring efficiency of operations.

Role of Database administrator

In order to manage and maintain the effectiveness, security, and integrity of a database system, a Database Administrator (DBA) is essential. DBAs are essential to many aspects of database management and administration during the course of a database's lifecycle. The following are some significant duties and tasks of a database administrator:

i. **Database Design**
DBAs collaborate closely with developers and other stakeholders to establish the database structure, schema, and relationships throughout the design phase of a database system. They make sure that the design complies with the needs of the company and industry standards for performance and scalability.

ii. **Database Installation and Configuration**
DBAs are in charge of setting up the database management system (DBMS) software and customizing it to the requirements of the enterprise. To maximize performance and resource usage, they configure the appropriate parameters, memory allocation, storage configuration, and other system settings.

iii. **Performance Tuning and Optimization**
DBAs monitor and assess the performance of databases, finding bottlenecks and improving queries and database configurations for increased effectiveness. To improve system responsiveness and throughput, they carry out

performance tuning procedures such index optimization, query optimization, database parameter adjustments, and workload management.

iv. Backup and Recovery

To safeguard data against possible loss or corruption, DBAs create and maintain backup and recovery strategies. To ensure that data can be recovered in the event of system failures, natural disasters, or human mistake, they design backup schedules, carry out routine backups, and test restoration procedures.

v. Security and Access Control

To protect the database and its sensitive data, DBAs put security procedures in place and make sure they are followed. They oversee user access privileges, roles, and permissions to the database, making sure that only authorized users have the proper access. To safeguard the integrity and confidentiality of data, DBAs also use security measures such as encryption and auditing tools.

vi. Database Monitoring and Maintenance

DBAs keep constant watch on the database system for problems with performance, availability, and security. To monitor resource usage, spot anomalies, and take preventative measures against possible issues, they employ monitoring tools. To maintain optimal database performance, DBAs also carry out standard maintenance procedures such database rearrangement, statistics updating, and space management.

vii. Capacity Planning and Scalability

DBAs evaluate the database system's present and potential needs and make plans for its expansion and scalability. In order to meet rising data volumes and user needs, they keep an eye on patterns of data growth, resource use trends, and prepare for capacity improvements, storage expansion, and system innovations.

viii. Database Upgrades and Patch Management

ix. Disaster Recovery Planning

1.8.2 Database User

A person who uses or accesses the data from a database is called database user.

Types of user:

1. Application Programmer

Application programmer implements the specification generated by system analyst in some high level programming language

2. Naive End User

Naïve end user is typically unaware of DBMS and access the database through specially written application programs. Naïve end user uses form based interface for querying and updating database.

3. Sophisticated End User

Sophisticated end user may use high level query language to perform any required operations. Engineers, scientists, business analyst who thoroughly familiarize themselves with the facilities of DBMS so as to implement their application to meet their complex requirement are sophisticated end user.

SOLUTION TO EXAMS' AND OTHER IMPORTANT QUESTIONS

1. List the major steps that you would take in setting up a database for a particular enterprise. [2018 Fall]

ANS: Major steps involved in setting up a database for an enterprise:

i. Identify the purpose and scope of the database

Before setting up a database, it's important to understand why the enterprise needs a database and what information it needs to store and manage. This will help determine the scope of the database and guide the design process.

ii. Choose a DBMS

The next step is to choose a suitable DBMS based on the enterprise's requirements and budget. The DBMS should be able to handle the scale and complexity of the enterprise's data and provide the necessary features for data management, security, and analysis.

iii. Define the data model

The data model defines the structure of the database and how data is organized and related. This involves identifying the entities (objects or concepts) that need to be stored in the database, as well as their attributes and relationships.

iv. Design the database schema

Based on the data model, the next step is to design the database schema, which specifies the tables, columns, and constraints that make up the database. This includes defining primary and foreign keys, setting up indexes, and enforcing data integrity rules.

v. Populate the database

Once the schema is defined, the database can be populated with data. This involves entering data manually, importing data from external sources, or a combination of both.

vi. Test the database

vii. Deploy the database

viii. Maintain the database

The process may involve additional steps depending on the enterprise's specific needs and requirements.

2. What are the major responsibilities of Database Management System? For each responsibility, explain the problems that would arise if the responsibility were not discharged. [2014 Spring]

ANS: The main responsibilities of a database management system(DBMS) are listed below, along with the problems that may occur if each responsibilities are not discharged:

i. Data Definition

- **Responsibility:** Defining the structure and schema of the database.
- **Problem:** Data inconsistency and ambiguity can result from improperly defined data. Users could interpret the data differently in the absence of explicit definitions, leading to inaccurate analysis, reporting, and decision-making.

ii. Data Manipulation

- **Responsibility:** Inserting, updating, deleting, and retrieving data from the database.
- **Problem:** Data integrity may be at risk if data manipulation is not done properly. Data that is inaccurate or lacking certain information may be stored, producing inaccurate results and untrustworthy information. Decision-making and business operations may be impacted by inconsistent data.

iii. Data Storage and Retrieval

- **Responsibility:** Managing the physical storage and efficient retrieval of data.
- **Problem:** Inefficient data storage can lead to poor performance and higher storage needs. Data retrieval can become time-consuming without effective indexing

and storage optimization, which has an adverse effect on system responsiveness and user experience.

iv. Data Security

- **Responsibility:** Enforcing security measures to protect data from unauthorized access.
- **Problem:** Sensitive data can be compromised without sufficient security, resulting in data breaches, privacy violations, and legal repercussions. Unauthorized users have the ability to access, change, or destroy data, resulting in financial losses and reputational harm to a business.

v. Data Integrity and Consistency

- **Responsibility:** Enforcing data integrity constraints and maintaining consistency.
- **Problem:** Data that is inconsistent or wrong might produce unreliable analyses, reports, and decisions. Problems with data integrity can compromise the system's credibility and dependability as a whole.

vi. Concurrency Control

- **Responsibility:** Managing concurrent access to the database by multiple users.
- **Problem:** Conflicts may occur if numerous users attempt to change the same piece of data at the same time without appropriate concurrency management. Inconsistent data can result in inaccurate findings and corrupted data, such as missing updates or dirty reads.

vii. Backup and Recovery

- **Responsibility:** Creating backups and restoring data in case of failures or errors.
- **Problem:** Data loss that is permanent might be caused by insufficient or poor backups. Without proper recovery measures, system malfunctions or human mistake might result in data being lost forever, resulting in operational halts and possibly large losses.

viii. Performance Optimization

- ***Responsibility:*** Optimizing the performance of the database system.

- **Problem:** Without enough performance optimization, queries can be executed inefficiently, responses take a long time to process, and resources can be used up quickly. This may lead to poor system performance, disgruntled users, and reduced productivity.

ix. Data Independence

- **Responsibility:** Providing an abstraction layer between the physical and logical data representation.
- **Problem:** Applications and the database structure may get entangled as a result of a lack of data independence. All dependent applications may need to be modified if the database schema is changed, which can complicate maintenance and limit system adaptability.

x. Data Administration

- **Responsibility:** Managing the overall administration and maintenance of the database system.
- **Problem:** Inadequate data management can lead to underwhelming system performance, security flaws, and ineffective resource management. The database system may become unreliable and challenging to administer without sufficient monitoring, upkeep, and user control.

To guarantee data reliability, security, and optimum system performance, these responsibilities must be fulfilled. Data inconsistencies, security breaches, poor performance, and operational difficulties inside the database system can result from ignoring these responsibilities.

3. Differentiate between Data abstraction and Data independence. **[2014 Fall]**

ANS: The difference between data abstraction and Data independence is given below:

- **Data abstraction**

  Data abstraction refers to the process of hiding the complexity of the underlying data model from the users and applications that access the database. This is typically achieved through the use of abstraction layers or interfaces that provide a simplified view of the database and shield

users and applications from the underlying details. The purpose of data abstraction is to provide a simple and consistent interface for users and applications to interact with the database, regardless of the underlying complexity.

Example:

In a database for an e-commerce website, the abstraction layer might provide a simple interface for users to search for products, add them to their cart, and check out. The users don't need to know about the complex data model used to store information about products, inventory levels, and customer orders. The abstraction layer hides these details and presents a simplified view of the data.

- **Data independence**

  Data independence refers to the ability to modify the schema or data storage mechanisms without affecting the applications or users that depend on the database. There are two types of data independence: physical data independence and logical data independence.

  Physical data independence allows the physical storage structure of the database to be changed without affecting the applications or users.

  Example:

  The underlying storage medium (such as hard disk or solid-state drive) could be changed, or the data could be partitioned across multiple disks, without affecting the applications or users.

  Logical data independence allows the logical structure of the database to be changed without affecting the applications or users.

  Example:

  The schema could be modified to add or remove tables, columns, or constraints, without affecting the applications or users.

  The purpose of data independence is to make the database more flexible and scalable, allowing it to adapt to changing requirements or technologies without requiring major changes to applications or users.

◆◆◆