Que : What are the different levels of Abstraction ?

- For the system to be usable,it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the data base.
- Since manydatabase-system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system.

Physical level –

- The lowest level of abstraction describes how the data are ac tually stored. The physical level describes complex low-level data structures in detail.

Logical level –

- The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data.
- The logical level thus describes the entire database in terms of a small number of relatively simple structures.
- Although implementation of the simple struc tures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity.
- This is referred to as physical data independence. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

View level –

- The highest level of abstraction describes only part of the entire database.
- Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database.
- Manyusers of the database system do not need all this information; instead, they need to access only a part of the database.
- The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

The three levels of data abstraction :

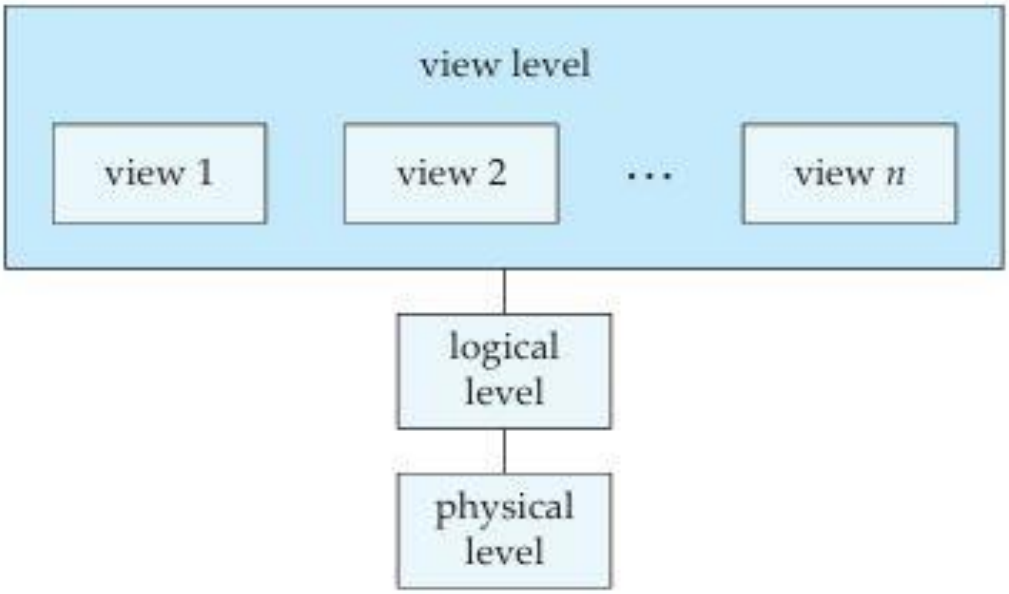

Figure 1.1 The three levels of data abstraction.

Explain the following terms

1. **Super Key :**
   - Asuperkeyis a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation.
   - For example, the ID attribute of the relation instructor is sufficient to distinguish one instructor tuple from another.
   - Thus, ID is a superkey. The name attribute of instructor, on the other hand, is not a superkey, because several instructors might have the same name.
   - Formally, let R denote the set of attributes in the schema of relation r. Ifwe say that a subset K of R is a superkey for r, we are restricting consideration to instances of relations r in which no two distinct tuples have the same values on all attributes in K.
   - That is,if t1 and t2 are in r and t1 = t2, then t1.K = t2.K. A super key may contain extraneous attributes. For example,the combination of ID and name is a superkey for the relation instructor.
   - If K is a superkey, then so is any superset of K. We are often interested in super keys for which no proper subset is a superkey. Such minimal superkeys are called candidate keys.
2. **Candidate key :**
   - Definition: A candidate key is a minimal superkey, meaning it is a superkey for which no proper subset is also a superkey. In other words, removing any attribute from the candidate key would result in a set that is not a superkey anymore.
   - Example: In the "instructor" relation, if the combination of name and dept name is sufficient to distinguish among members, both {ID} and {name, dept name} can be candidate keys. However, {ID, name} is not a candidate key because {ID} alone is
3. **Primary key :**
   - The primary key is a type of candidate key specifically chosen by the database designer to uniquely identify tuples within a relation.
   - It is a property of the entire relation, representing a constraint that ensures data integrity.
   - The values of the primary key attributes must be unique across all tuples in the relation.
   - Primary keys should be chosen with care to ensure uniqueness and stability.

- Attributes chosen as primary keys should rarely, if ever, change. For example, social-security numbers or unique identifiers generated by enterprises are often suitable as primary keys.
- Attributes that are likely to change, such as a person's address, should not be part of the primary key.

4. **Reference key :**
    - A foreign key in a relation, say r1, is an attribute that refers to the primary key of another relation, say r2. This establishes a link or dependency between the two relations.
    - Referencing Relation: The relation r1, which includes the foreign key, is called the referencing relation of the foreign key dependency.
    - Referenced Relation: The relation r2, whose primary key is being referenced, is called the referenced relation of the foreign key.
    - Example: In the "instructor" relation, the attribute "dept name" can be a foreign key referencing the primary key "dept name" in the "department" relation.
    - A referential integrity constraint ensures that values appearing in specified attributes of any tuple in the referencing relation must also appear in specified attributes of at least one tuple in the referenced relation.
    - Example Constraint: In the given passage, the example of the "section" and "teaches" relations is provided. A constraint could be enforced such that if a particular combination (course id, sec id, semester, year) appears in the "section" relation, then the same combination must appear in the "teaches" relation.
    - It is mentioned that in some cases, a set of values, even if it uniquely identifies a tuple in the referencing relation, may not form a primary key in the referenced relation.
    - Example Limitation: In the passage, it is noted that for the "teaches" relation, a combination of values (course id, sec id, semester, year) does not form a primary key, as more than one instructor may teach the same section. This prevents the declaration of a foreign key constraint from "section" to "teaches," though the constraint can be defined in the reverse direction.

Purpose of DBMS

The purpose of a Database Management System (DBMS) is to address and overcome the limitations and challenges posed by conventional file-processing systems. The passage highlights several drawbacks of file-processing systems, and the following outlines the key purposes of implementing a DBMS:

1. Elimination of Data Redundancy and Inconsistency:
    - Challenge: In file-processing systems, different programmers create files and application programs over time, leading to varying structures, programming languages, and data redundancy.
    - Purpose: A DBMS eliminates data redundancy by providing a centralized and standardized approach to data storage. It ensures data consistency by maintaining a single source of truth for each piece of information.
2. Improved Data Access:
    - Challenge: Difficulty in accessing data in conventional file-processing systems, as the absence of specific application programs for certain queries hinders efficient data retrieval.
    - Purpose: A DBMS facilitates efficient data retrieval through its query language and indexing mechanisms. Users can query the database without needing custom application programs, improving accessibility.
3. Data Isolation:
    - Challenge: In file-processing systems, data is scattered across various files, making it challenging to write new application programs to retrieve the necessary data.
    - Purpose: A DBMS centralizes data storage and provides a unified interface for data retrieval. This simplifies the process of writing new application programs and ensures data isolation is minimized.
4. Integrity and Consistency Enforcement:
    - Challenge: Ensuring data values meet consistency constraints in a file-processing system is difficult, and modifying programs to enforce new constraints is challenging.
    - Purpose: A DBMS enforces integrity constraints through its data model and allows for easy modification of constraints without altering

application programs, ensuring data consistency and adherence to defined rules.

5. Atomicity and Transaction Management:
    - Challenge: Ensuring atomicity (complete execution or none at all) of transactions is complex in file-processing systems.
    - Purpose: A DBMS ensures atomicity by providing transaction management mechanisms. If a failure occurs during a transaction, the system can roll back changes to maintain a consistent database state.
6. Concurrent Access Control:
    - Challenge: Simultaneous access by multiple users can lead to inconsistent data in file-processing systems.
    - Purpose: A DBMS provides mechanisms to control concurrent access, preventing anomalies and maintaining data consistency even with multiple users accessing and updating data simultaneously.
7. Enhanced Security:
    - Challenge: Enforcing security constraints, limiting access to specific data, is difficult in file-processing systems.
    - Purpose: A DBMS allows for the implementation of security measures, ensuring that only authorized users can access specific portions of the database, addressing security concerns.