SE 2141

Week 4

Data Models

`data modeling`
The process of creating a specific data model for a determined problem domain.

`data model`
A representation, usually graphic, of a complex "real-world" data structure.
Data models are used in the database design phase of the Database Life Cycle.

Note

An implementation-ready data model should contain at least the following components:

- A description of the data structure that will store the end-user data
- A set of enforceable rules to guarantee the integrity of the data
- A data manipulation methodology to support the real-world data transformations

The Importance of Data Models

- **Facilitates communication**
  Data models help bridge communication between designers, application programmers, and end users.
- **Improves organizational understanding**
  A well-structured data model can provide a clearer understanding of how different parts of an organization fit together.
- **Provides different perspectives**
  Different users (e.g., managers, clerks, programmers) view data from varying perspectives, and a data model accommodates these diverse views.
- **Offers a holistic view**
  Like a blueprint for a house, a data model gives an overall, cohesive view of the data, ensuring consistency across the organization.
- **Prevents conflicts**
  A well-designed data model avoids inconsistencies in data usage, which can otherwise lead to costly mistakes.
- **Guides database development**
  Just as a blueprint is essential to building a house, a data model is crucial for creating an efficient and accurate database.

Data Model Basic Building Blocks

- An **`entity`** is a person, place, thing, or event about which data will be collected and stored. An entity represents a particular type of object in the real world, which means an entity is "distinguishable" — that is, each entity occurrence is unique and distinct.
- An **`attribute`** is a characteristic of an entity
- A **`relationship`** describes an association among entities
- A **`constraint`** is a restriction placed on the data. Constraints are important because they help to ensure data integrity. Constraints are normally expressed in the form of rules:
    - An employees salary must have values that are between 6,000 and 350,000.
    - A student's GPA must be between 0.00 and 4.00.
    - Each class must have one and only one teacher.

Data Models use Three Types of Relationships

- **One-to-many (1:M or 1..*) relationship**

  Associations among two or more entities that are used by data models. In a 1:M relationship, one entity instance is associated with many instances of the related entity.

- **Many-to-many (M:N or *..*) relationship**

  Association among two or more entities in which one occurrence of an entity is associated with many occurrences of a related entity and one occurrence of the related entity is associated with many occurrences of the first entity.

- **One-to-one (1:1 or 1..1) relationship**

  Associations among two or more entities that are used by data models. In a 1:1 relationship, one entity instance is associated with only one instance of the related entity.

The Evolution of Data Models

- ➔ Hierarchical
- ➔ Network
- ➔ Relational
- ➔ Entity relationship
- ➔ Object oriented (00)

Hierarchical model

An early database model whose basic concepts and characteristics formed the basis for subsequent database development.
This model is based on an upside-down tree structure in which each record is called a **segment**.
The top record is the root segment. Each segment has a 1:M relationship to the segment directly below it.

The hierarchical model depicts a set of one-to-many (1:M) relationships between a parent and its children segments.

- ➔ each parent can have many children
- ➔ each child has only one parent

Benefits of Hierarchical Model

Efficiency: Allowed faster data retrieval compared to file-based systems.

Clear Relationships: Simple parent-child structure made it easy to see connections.

Example: Used in banking and telecommunication systems.

Limitations of Hierarchical Model

Rigidity: Data relationships are inflexible; one-to-one or one-to-many relationships only.

Difficult to Model Complex Data: Complex many-to-many relationships not easily represented.

Limited Query Flexibility: Difficult to search across branches.

Network Model

- The network model was created to represent complex data relationships more effectively than the hierarchical model, to improve database performance, and to impose a database standard.
- In the network model, the user perceives the network database as a collection of records in 1:M relationships.
- Network model allows a record to have more than one parent.

Example: CODASYL DBTG (Conference on Data Systems Languages Database Task Group)

Benefits of Network Model

Flexibility: Supports complex data relationships that the hierarchical model couldn’t handle.

Efficient Data Navigation: Users could efficiently navigate the graph structure to retrieve related records.

Use Cases: Ideal for managing supply chains, telecommunications, and airline reservation systems.

Limitations of Network Model

Complexity: More complex to design and implement compared to hierarchical models.

Steep Learning Curve: Required expert knowledge to manage and query data.

Difficult to Modify: Changes to the database structure could be cumbersome.

Standard database concepts

- The **schema** is the conceptual organization of the entire database as viewed by the database administrator.
- The **subschema** defines the portion of the database "seen" by the application programs that actually produce the desired information from the data within the database.
- A **data manipulation language (DML)** defines the environment in which data can be managed and is used to work with the data in the database.
- A **schema data definition language (DDL)** enables the database administrator to define the schema components.

Relational Model

- ➢ developed by Codd of IBM in 1970
- ➢ is based on mathematical set theory and represents data as independent relations
- ➢ data stored in tables (relations), with rows (tuples) and columns (attributes)
- ➢ produced an "automatic transmission" database to replace the "standard transmission" databases that preceded it
- ➢ is implemented through a very sophisticated relational database management system (RDBMS)

Key Features:

- Based on mathematical set theory
- Relations allow easy data manipulation and queries

Examples: SQL, Oracle, MySQL.

Benefits of Relational Model

Simplicity: Easier to understand and use compared to hierarchical and network models.

Powerful Querying: SQL enables complex queries with simple commands.

Flexibility: Tables can easily be joined to form new relationships.

Widespread Use: Relational databases like Oracle and MySQL are industry standards.

Limitations of Relational Model

Performance Issues with Big Data: As data grows, relational databases can struggle with performance and scalability.

Structured Data Only: Works best for structured data; less effective for unstructured data like multimedia.

Schema Changes: Modifying the schema can be time-consuming, especially in large databases.

Relational Database Management System (RDBMS)

- performs the same basic functions provided by the hierarchical and network DBMS systems, in addition to a host of other functions that make the relational data model easier to understand and implement
- most important advantage of the RDBMS is its ability to hide the complexities of the relational model from the user
- RDBMS manages all of the physical details, while the user sees the relational database as a collection of tables in which data is stored
- RDBMS software translates a user's logical requests (queries) into commands that physically locate and retrieve the requested data

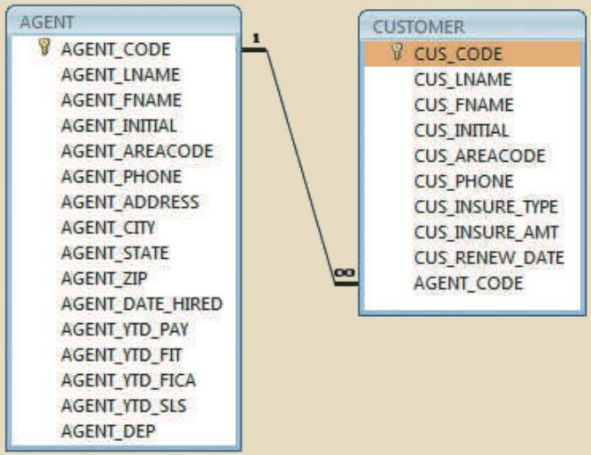
table (relation)
A logical construct perceived to be a two-dimensional structure composed of intersecting rows (entities) and columns (attributes) that represents an entity set in the relational model

tuple
In the relational model, a table row

relational diagram
A graphical representation of a relational database's entities, the attributes within those entities, and the relationships among the entities.

Relational Diagram

FIGURE 2.2 A RELATIONAL DIAGRAM



- A relational table stores a collection of related entities.
- Unlike a file, a relational table provides complete data and structural independence.
- The table is a logical structure; the physical storage of data is irrelevant to users or designers.
- This abstraction led to a significant breakthrough in database technology.
- The rise of the relational data model is due to its powerful and flexible query language, SQL.
- SQL allows users to define what data to retrieve without specifying how to retrieve it.
- SQL simplifies data retrieval compared to other database or file systems.

From an end-user perspective, an SQL-based relational database has three parts: the user interface, tables, and the SQL engine.

End-user interface

Allows users to interact with data by automatically generating SQL code. Interfaces vary based on the software vendor but can be customized with application generators.

A collection of tables stored in the database

All data is stored in independent tables that present information in an easy-to-understand format. Rows in different tables are linked by common attributes.

SQL engine:

Executes data queries behind the scenes. It processes user requests for creating tables, accessing data, and performing maintenance without users needing to know how it works, as SQL specifies what must be done, not how.

Entity Relationship (ER) Model (ERM)

- Widely accepted and adapted graphical tool for data modeling
- Introduced by Chen in 1976
- Graphical representation of entities and their relationships in a database structure
- A data model that describes relationships (1:1, 1:M, and M:N) among entities at the conceptual level with the help of ER diagrams.

The Entity-Relationship (ER) model is based on three main components:

Entity: An entity is any object or concept about which data is collected and stored.
In an Entity-Relationship Diagram (ERD), it is represented by a **rectangle** with the entity's name in capital letters and singular form.

Attributes: Each entity has attributes that describe particular characteristics. Attributes are detailed in ERDs to provide information about each entity.

Relationships: Relationships describe associations among datar. They typically describe associations between two entities and are categorized into one-to-many (1:M), many-to-many (M:N), and one-to-one (1:1) types.

entity relationship diagram (ERD)
A diagram that depicts an entity relationship model's entities, attributes, and relations.

entity instance (entity occurrence)
A row in a relational table.

entity set
A collection of like entities.

Connectivity
The type of relationship between entities.Classifications include 1:1, 1:M, and M:N.

Chen notation
A method for creating Entity-Relationship Diagrams (ERDs) where entities are represented by rectangles, attributes by ovals, and relationships by diamonds, with cardinality shown through lines and annotations.

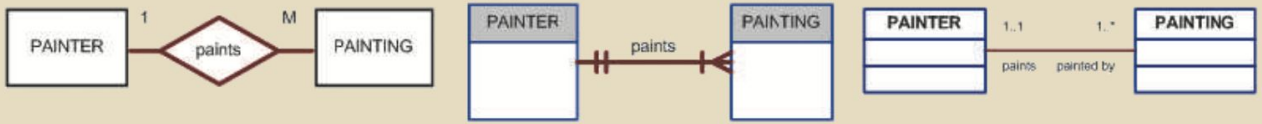
Crow's Foot notation
A representation of the entity relationship diagram that uses a three-pronged symbol to represent the "many" sides of the relationship.

class diagram notation
The set of symbols used in the creation of class diagrams.

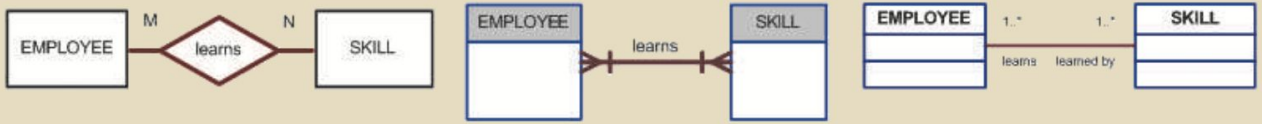
FIGURE 2.3 THE ER MODEL NOTATIONS

Chen Notation ***Crow's Foot Notation*** *UML Class Diagram Notation*

A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER.



A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs.



A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.

Object Oriented Model

Increasing complexity in real-world problems required a data model that better represents real-world entities.

Object-Oriented Data Model (OODM):

- Encapsulates both data and relationships in a single structure known as an object.
- Basis for Object-Oriented Database Management System (OODBMS).
- Is said to be a semantic data model

OODM Components

Objects: Abstractions of real-world entities, representing individual occurrences of an entity.

Attributes: Properties that describe an object's characteristics, such as Name or Social Security Number for a PERSON object.

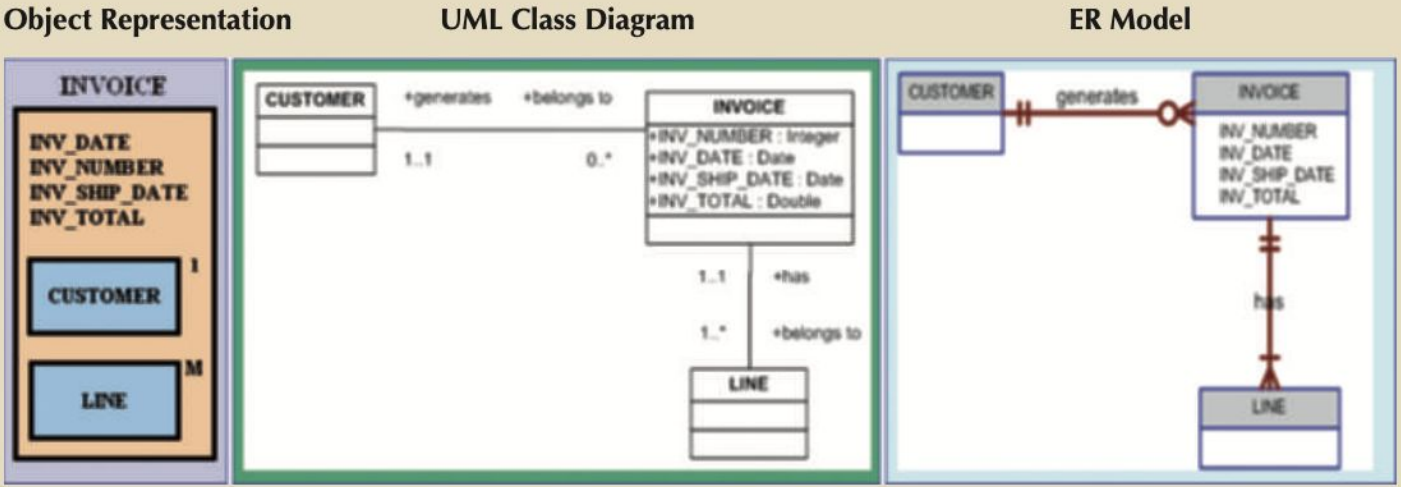
Classes: Collections of similar objects with shared attributes and methods, defining the structure and behavior of objects. Classes group objects and include methods that define their actions.

Class Hierarchy: An organizational structure resembling an upside-down tree, where each class has a single parent class, allowing for inheritance of attributes and methods.

Inheritance: The ability for subclasses to inherit attributes and methods from parent classes, facilitating reuse and extension of existing class definitions.

UML Class Diagrams: Used to visually represent the structure and relationships of objects within the system, based on object-oriented concepts.

FIGURE 2.4 A COMPARISON OF THE OO, UML, AND ER MODELS



object-oriented database management system (OODBMS)

Data management software used to manage data in an object-oriented database model.

semantic data model

The first of a series of data models that models both data and their relationships in a single structure known as an object.

class

A collection of similar objects with shared structure (attributes) and behavior (methods). A class encapsulates an object's data representation and a methods implementation.

method

In the object-oriented data model, a named set of instructions to perform an action. Methods represent real-world actions.

Object/Relational and XML

extended relational data model (ERDM)

A model that includes the object-oriented model's best features in an inherently simpler relational database structural environment.

object/relational database management system (O/R DBMS)

A DBMS based on the extended relational model (ERDM). The ERDM, championed by many relational database researchers, constitutes the relational model's response to the OODM. This model includes many of the object-oriented model's best features within an inherently simpler relational database structure.

Extensible Markup Language (XML)

A metalanguage used to represent and manipulate data elements. Unlike other markup languages, XML permits the manipulation of a document's data elements. XML facilitates the exchange of structured documents such as orders and invoices over the Internet.

Emerging Data Models: Big Data and NoSQL

Big Data refers to a movement to find new and better ways to manage large amounts of web- and sensor-generated data and derive business insight from it, while simultaneously providing high performance and scalability at a reasonable cost.

Basic characteristics of Big Data databases*: volume, velocity, and variety, or the 3 Vs.

Basic characteristics of Big Data databases: 3 Vs

Volume refers to the amounts of data being stored.

Furthermore, organizations are using multiple technologies to interact with end users and those technologies are generating mountains of data. This ever-growing volume of data quickly reached petabytes in size, and it's still growing.

Velocity refers not only to the speed with which data grows but also to the need to process this data quickly in order to generate information and insight.

The velocity of data growth is also due to the increase in the number of different data streams from which data is being piped to the organization (via the web, e-commerce, Tweets, Facebook posts, emails, sensors, GPS, and so on).

Variety refers to the fact that the data being collected comes in multiple different data formats.

A great portion of these data comes in formats not suitable to be handled by the typical operational databases based on the relational model.

Some of the most frequently used Big Data technologies are Hadoop, MapReduce, and NoSQL databases.

Hadoop

- ➔ is a Java-based, open-source, high-speed, fault-tolerant distributed storage and computational framework.
- ➔ uses low-cost hardware to create clusters of thousands of computer nodes to store and process data. Hadoop originated from Google's work on distributed file systems and parallel processing and is currently supported by the Apache Software Foundation.
- ➔ has several modules, but the two main components are Hadoop Distributed File System (HDFS) and MapReduce.

Hadoop Distributed File System (HDFS)

- ➔ is a highly distributed, fault-tolerant file storage system designed to manage large amounts of data at high speeds.
- ➔ In order to achieve high throughput, HDFS uses the write-once, read many model. This means that once the data is written, it cannot be modified.
- ➔ HDFS uses three types of nodes: a `name node` that stores all the metadata about the file system, a `data node` that stores fixed-size data blocks (that could be replicated to other data nodes), and a `client node` that acts as the interface between the user application and the HDFS.

MapReduce

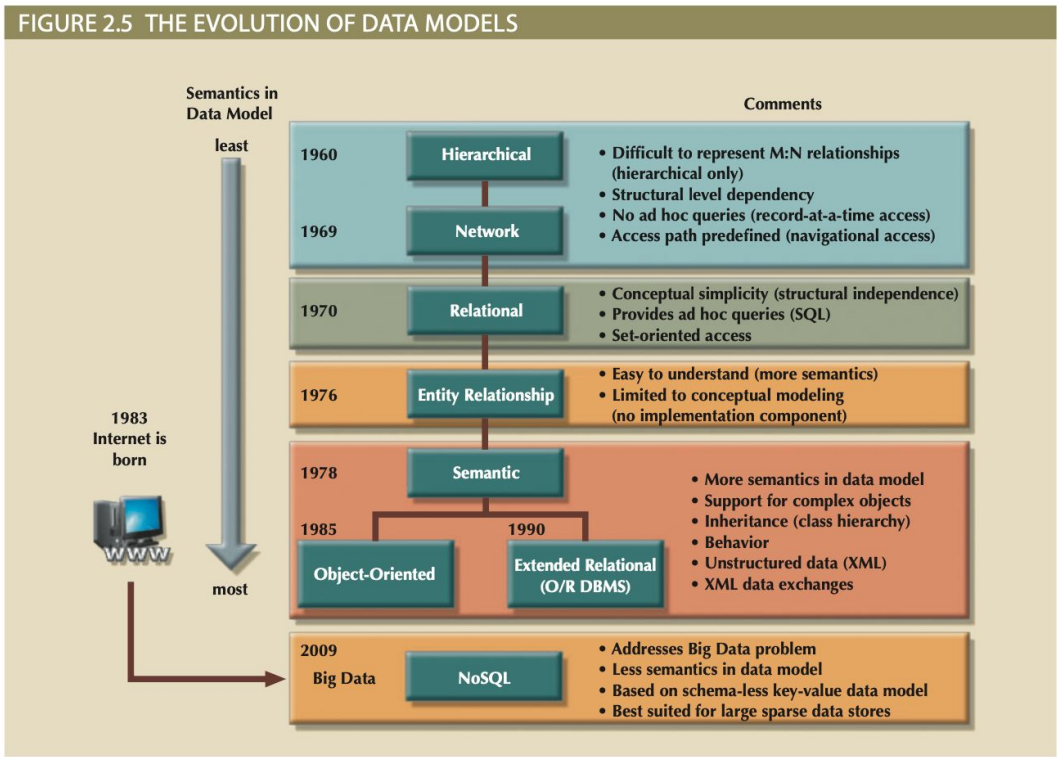
- ➔ is an open-source application programming interface (API) that provides fast data analytics services.
- ➔ distributes the processing of the data among thousands of nodes in parallel.
- ➔ works with structured and nonstructured data.
- ➔ The MapReduce framework provides two main functions:Map and Reduce.
- ➔ Map function takes a job and divides it into smaller units of work, and the Reduce function collects all the output results generated from the nodes and integrates them into a single result set.

NoSQL

- ➔ is a large-scale distributed database system that stores structured and unstructured data in efficient ways.

Data Models: A Summary

FIGURE 2.5 THE EVOLUTION OF DATA MODELS



Degrees of Data Abstraction

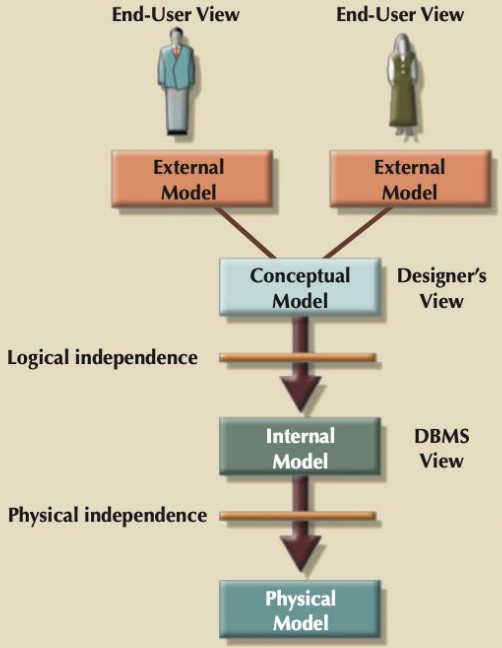
The American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC) defined a framework for data modeling based on degrees of data abstraction.

The resulting ANSI/SPARC architecture defines three levels of data abstraction: **external, conceptual, and internal**.

American National Standards Institute (ANSI)

The group that accepted the DBTG recommendations and augmented database standards in 1975 through its SPARC committee.

FIGURE 2.6 DATA ABSTRACTION LEVELS



Degree of Abstraction		Characteristics
High	ER Object-Oriented	Hardware-independent Software-independent
Medium	Relational	Hardware-independent Software-dependent
Low	Network Hierarchical	Hardware-dependent Software-dependent

The External Model

The external model is the end users' view of the data environment.

The term end users refers to people who use the application programs to manipulate the data and generate information. End users usually operate in an environment in which an application has a specific business unit focus.

Because data is being modeled, ER diagrams will be used to represent the external views. A specific representation of an external view is known as an `external schema`.

The use of external views that represent subsets of the database has some important advantages:

- It is easy to identify specific data required to support each business unit's operations.
- It makes the designer's job easy by providing feedback about the mode's adequacy.
- It helps to ensure security constraints in the database design.
    - Damaging an entire database is more difficult when each business unit works with only a subset of data.
- It makes application program development much simpler.

The Conceptual Model

- ➔ represents a global view of the entire database by the entire organization.
- ➔ integrates all external views (entities, relationships, constraints, and processes) into a single global view of the data in the enterprise.
- ➔ most widely used conceptual model is the ER model.

`Conceptual schema`

- it is the basis for the identification and high-level description of the main data objects (avoiding any database model-specific details)
- a representation of the conceptual model, usually expressed graphically

The conceptual model yields some important advantages.

1 It provides a bird's-eye (macro level) view of the data environment that is relatively easy to understand.

2 It is independent of both software and hardware.

`Software independence` means that the model does not depend on the DBMS software used to implement the model.

`Hardware independence` means that the model does not depend on the hardware used in the implementation of the model.

Therefore, changes in either the hardware or the DBMS software will have no effect on the database design at the conceptual level. Generally, the term `logical design` refers to the task of creating a conceptual data model that could be implemented in any DBMS.

The Internal Model

- once a specific DBMS has been selected, the internal model maps the conceptual model to the DBMS.
- is the representation of the database as "seen" by the DBMS.
- requires the designer to match the conceptual model's characteristics and constraints to those of the selected implementation model.
- an internal schema depicts a specific representation of an internal model, using the database constructs supported by the chosen database.
- because the internal model depends on specific database software, it is said to be *software dependent*.

The Physical Model

- operates at the lowest level of abstraction, describing the way data is saved on storage media such as magnetic, solid state, or optical media. T
- requires the definition of both the physical storage devices and the (physical) access methods required to reach the data within those storage devices, making it both software and hardware dependent.

Although the relational model does not require the designer to be concerned about the datas physical storage characteristics, the implementation of a relational model may require physical-level fine-tuning for increased performance.

`Fine-tuning` is especially important when very large databases are installed in a mainframe environment, yet even such performance fine-tuning at the physical level does not require knowledge of physical data storage characteristics.

logical independence

A condition in which the internal model can be changed without affecting the conceptual model.

The internal model is hardware-independent because it is unaffected by the computer on which the software is installed. Therefore, a change in storage devices or operating systems will not affect the internal model.

physical independence

A condition in which the physical model can be changed without affecting the internal model.

LEVELS OF DATA ABSTRACTION

MODEL	DEGREE OF ABSTRACTION	FOCUS	INDEPENDENT OF
External	High	End-user views	Hardware and software
Conceptual		Global view of data (database model independent)	Hardware and software
Internal		Specific database model	Hardware
Physical	Low	Storage and access methods	Neither hardware nor software