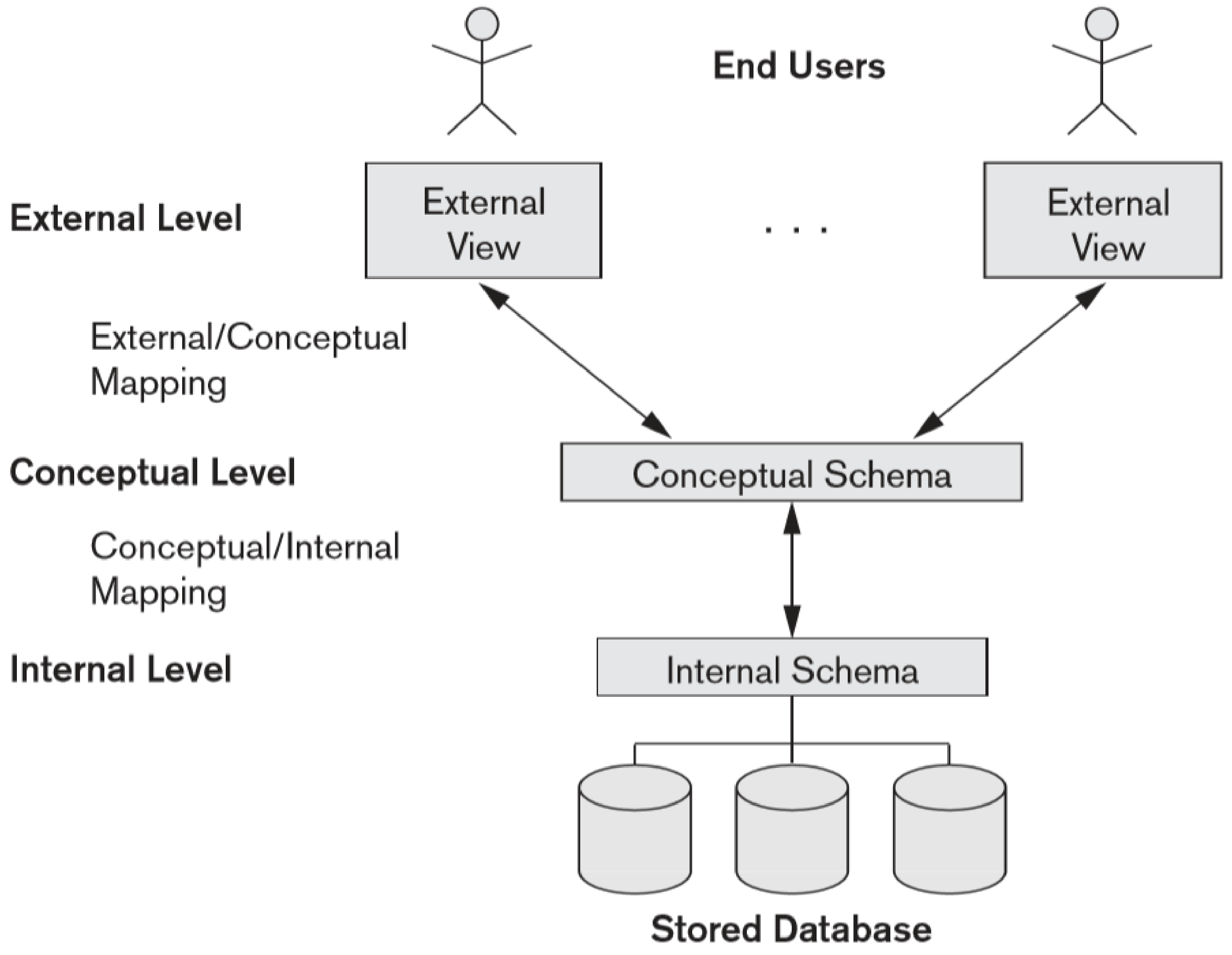
Database System Concepts and Architecture

Part 2

Three-Schema Architecture and Data Independence

The Three-Schema Architecture



- 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.

- The **conceptual level** has a **conceptual schema**, which describes the structure of the whole database for a community of users.
- The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
- Usually, a representational data model is used to describe the conceptual schema when a database system is implemented.
- This *implementation conceptual schema* is often based on a *conceptual schema design* in a high-level data model.

- 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.
- As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.

Data Independence

- The three-schema architecture can be used to further explain the concept of **data independence**, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

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

- **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.

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

Database Languages and Interfaces

DBMS Languages

- In many DBMSs where no strict separation of levels is maintained, one language, called the **data definition language** (**DDL**), is used by the DBA and by database designers to define both conceptual and internal schemas.

- In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only.
- Another language, the **storage definition language** (**SDL**), is used to specify the internal schema.
- In most relational DBMSs today, there is no specific language that performs the role of SDL.
- Instead, the internal schema is specified by a combination of functions, parameters, and specifications related to storage.

- For a true three-schema architecture, we would need a third language, the **view definition language** (**VDL**), to specify user views and their mappings to the conceptual schema, but in most DBMSs *the DDL is used to define both conceptual and external schemas*.
- In relational DBMSs, SQL is used in the role of VDL to define user or application views as results of predefined queries

- Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database.
- Typical manipulations include retrieval, insertion, deletion, and modification of the data.
- The DBMS provides a set of operations or a language called the **data manipulation language** (**DML**) for these purposes.

- In current DBMSs, the preceding types of languages are usually *not considered distinct languages*; rather, a comprehensive integrated language is used that includes constructs for conceptual schema definition, view definition, and data manipulation.
- Storage definition is typically kept separate, since it is used for defining physical storage structures to fine-tune the performance of the database system, which is usually done by the DBA staff.
- A typical example of a comprehensive database language is the SQL relational database language, which represents a combination of DDL, VDL, and DML, as well as statements for constraint specification, schema evolution, and other features.
- The SDL was a component in early versions of SQL but has been removed from the language to keep it at the conceptual and external levels only.

- There are two main types of DMLs.

- A **high-level** or **nonprocedural** DML can be used on its own to specify complex database operations concisely.
- Many DBMSs allow high-level DML statements either to be entered interactively from a display monitor or terminal or to be embedded in a general-purpose programming language.
- In the latter case, DML statements must be identified within the program so that they can be extracted by a precompiler and processed by the DBMS.

- A **low-level** or **procedural** DML must be embedded in a general-purpose programming language.
- This type of DML typically retrieves individual records or objects from the database and processes each separately.
- Therefore, it needs to use programming language constructs, such as looping, to retrieve and process each record from a set of records.

- Low-level DMLs are also called **record-at-a-time** DMLs because of this property.
- DL/1, a DML designed for the hierarchical model, is a low-level DML that uses commands such as GET UNIQUE, GET NEXT, or GET NEXT WITHIN PARENT to navigate from record to record within a hierarchy of records in the database.
- High-level DMLs, such as SQL, can specify and retrieve many records in a single DML statement; therefore, they are called **set-at-a-time** or **set-oriented** DMLs.
- A query in a high-level DML often specifies *which* data to retrieve rather than *how* to retrieve it; therefore, such languages are also called **declarative**.

- 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**.
- On the other hand, a high-level DML used in a standalone interactive manner is called a **query language**.

DBMS Interfaces

- Menu-Based Interfaces for Web Clients or Browsing
- Forms-Based Interfaces
- Graphical User Interfaces
- Natural Language Interfaces
- Speech Input and Output
- Interfaces for Parametric Users
- Interfaces for the DBA