Learning Objectives

- The Three-Level ANSI-SPARC Architecture
- Database Languages
- Data Models and Conceptual Modeling
- Function & Components of DBMS

Chapter 2 Part One: Database Environment

The Three-Level ANSI-SPARC Architecture

- An early standard terminology and general architecture for database systems produced in 1971 by Data Base Task Group (DBTG) and Conference on Data Systems and Languages (CODASYL, 1971).
- The DBTG recognized the need for a two-level approach with a system view called the schema and user views called subschemas.

The Three-Level ANSI-SPARC Architecture

- In 1975, The American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC), ANSI/SPARC produced a similar terminology and recognized the need for a three-level approach with a system catalog:
  - External Level
  - Conceptual Level
  - Internal Level
The Three-Level ANSI-SPARC Architecture

Objectives

- All users should be able to access the same data.
- A user’s view is immune to changes made in other views.
- Users should not need to know physical database storage details.
- The Database Administrator (DBA) should be able to change the database storage structures without affecting the users’ views.
- The internal structure of the database should be unaffected by changes to the physical aspects of storage.
- The DBA should be able to change the conceptual structure of the database without affecting all users.

Internal Level

- Describes the physical representation of the database on the computer.
- Describes how the data is stored in the database.

Conceptual Level

- Describes what data is stored in the database and relationships among the data.

External Level

- The user’s view of the database.
- Describes that part of database that is relevant to each user.

External/Conceptual Mapping

- Describes how the conceptual structures are related to external views.

Conceptual/Internal Mapping

- Describes how the internal structures are related to the conceptual schema.
Data Independence

- A major objective for the three-level architecture is to provide **data independence**, which means that upper levels are unaffected by changes to lower levels.

- There are two kinds of data independence:
  - **logical independence**: immunity of the external schemas to changes in the conceptual schema;
  - **physical independence**: immunity of the conceptual schema to changes in the internal schema.

Logical Data Independence

changes to the conceptual schema, such as the addition or removal of new entities, attributes, or relationships, should be possible without having to change existing external schemas or having to rewrite application programs.

Physical Data Independence

changes to the internal schema, such as using different file organizations or storage structures, using different storage devices, modifying indexes, or hashing algorithms, should be possible without having to change the conceptual or external schemas.

Database Languages

- A **data sublanguage** consists of two parts: a **Data Definition Language** (DDL) and a **Data Manipulation Language** (DML).

- The DDL is used to **specify the database schema**.

- The DML is used to **both read and update the database**.

- Database languages are called **data sublanguages** because they do not include constructs for all computing needs.

- Many DBMSs have a facility for **embedding** the sublanguage in a high-level programming language such as COBOL, Fortran, Pascal, Ada, ‘C’, C++, Java, or Visual Basic.
Database Language – Data Definition Language (DDL)

- A language that allows the DBA or user to describe and name the entities, attributes, and relationships required for the application, together with any associated integrity and security constraints.
- The DDL is used to define a schema or to modify an existing one. It cannot be used to manipulate data.
- The result of the compilation of the DDL statements is a set of tables stored in special files collectively called the system catalog.

Database Language – Data Definition Language (DDL)

- The system catalog integrates the metadata, that is data that describes objects in the database and makes it easier for those objects to be accessed or manipulated.
- The metadata contains definitions of records, data items, and other objects that are of interest to users or are required by the DBMS.
- In practice, there is one comprehensive DDL that allows specification of at least the external and conceptual schemas.

Database Language – Data Definition Language (DDL)

- Used to set up, change and remove data structures from tables by using these instructions:
  - create
  - alter
  - drop
  - rename
  - truncate

Database Language – Data Man. Language (DML)

- A language that provides a set of operations to support the basic data manipulation operations on the data held in the database.
- Data manipulation operations usually include the following:
  - insertion of new data into the database;
  - modification of data stored in the database;
  - retrieval of data contained in the database;
  - deletion of data from the database.
- Data manipulation applies to the external, conceptual, and internal levels.
**Database Language – Data Man. Language (DML)**

- The part of a DML that involves data retrieval is called a **query language**.
- The term ‘query’ is therefore reserved to denote a retrieval statement expressed in a query language.
- There are two types of DML: **procedural** and **non-procedural**
  - **Procedural DMLs**: allows the user to tell the system what data is needed and exactly how to retrieve the data.
  - **Non-procedural DMLs**: allows the user to state what data is needed rather than how it is to be retrieved.

**Database Language – Data Man. Language (DML)**

- Allow to enters new rows, changes existing rows, and removes unwanted rows from tables in the database, respectively using these instructions:
  - `insert`
  - `update`
  - `delete`
  - `merge`
  - `select` ➞ data retrieval/query language

**Database Language – Data Man. Language (DML)**

- The DBMS translates a DML statement into one or more procedures that manipulate the required sets of records.
- Non-procedural languages are also called **declarative languages**.
- Relational DBMSs usually include some form of non-procedural language for data manipulation, typically **SQL (Structured Query Language)** or **QBE (Query-By-Example)**.
- Non-procedural DMLs are normally easier to learn and use than procedural DMLs.

**Database Language – 4th Generation Language (4GL)**

- 4GL is non-procedural language that allow user to defines what is to be done, not how.
- The user does not define the steps that a program needs to perform a task, but instead defines parameters for the tools that use them to generate an application program.
- 4GL encompass:
  - presentation languages, such as query languages and report generators;
  - speciality languages, such as spreadsheets and database languages;
Database Language – 4th Generation Language (4GL)

- application generators that define, insert, update, and retrieve data from the database to build applications;
- very high-level languages that are used to generate application code.
• 4GL is non-procedural language that allow user to defines what is to be done, not how.

Data Model

• Schema is written using a data definition language and is written in the data definition language of a particular DBMS.
• The type of this language is too low level to describe the data requirements of an organization in a way that is readily understandable by a variety of users.
• The require for a higher-level description of the schema: that is, a data model.

Data Model

• Data model: An integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization.
• A data model can be thought of as comprising three components:
  1. a structural part, consisting of a set of rules according to which databases can be constructed;
  2. a manipulative part, defining the types of operation that are allowed on the data;
  3. possibly a set of integrity constraints, which ensures that the data is accurate.

• Purpose: to represent data in an understandable way.
• Categories of data models include:
  - Object-based;
  - Record-based; and
  - Physical.
• 1st & 2nd categorie: used to describe data at the conceptual and external levels
• 3rd categorie: used to describe data at the internal level.
Physical Data Model

- Physical data models describe how data is stored in the computer, representing information such as record structures, record orderings, and access paths.
- There are not as many physical data models as logical data models, the most common ones being the unifying model and the frame memory.

Conceptual Modeling

- From an examination of the three-level architecture, we see that the conceptual schema is the ‘heart’ of the database.
- It supports all the external views and, in turn, supported by the internal schema.
- The internal schema is merely the physical implementation of the conceptual schema.
- The conceptual schema should be a complete and accurate representation of the data requirements of the enterprise.

Conceptual Modeling as Conceptual Data Model

- Conceptual modeling, or conceptual database design, is the process of constructing a model of the information use in an enterprise that is independent of implementation details. (conceptual data model).

DBMS Functions

- Codd (1982) lists eight services that should be provided by any full-scale DBMS + two more that might reasonably be expected to be available.
  1. Data storage, retrieval, and update
     - A DBMS must furnish users with the ability to store, retrieve, and update data in the database.
  2. A user-accessible catalog
     - A DBMS must furnish a catalog in which descriptions of data items are stored and which is accessible to users.
### DBMS Functions

3. **Transaction support**
   - A DBMS must furnish a mechanism which will ensure either that all the updates corresponding to a given transaction are made or that none of them is made.

4. **Concurrency control services**
   - A DBMS must furnish a mechanism to ensure that the database is updated correctly when multiple users are updating the database concurrently.

5. **Recovery services**
   - A DBMS must furnish a mechanism for recovering the database in the event that the database is damaged in any way.

6. **Authorization services**
   - A DBMS must furnish a mechanism to ensure that only authorized users can access the database.

7. **Support for data communication**
   - A DBMS must be capable of integrating with communication software.

8. **Integrity services**
   - A DBMS must furnish a means to ensure that both the data in the database and changes to the data follow certain rules.

+ 2 more

9. **Services to promote data independence**
   - A DBMS must include facilities to support the independence of programs from the actual structure of the database.

10. **Utility services**
    - A DBMS should provide a set of utility services.

### DBMS Components

Components of DBMS:
- **Query processor**, transforms queries into a series of low-level instructions directed to the database manager.
- **Database manager (DM)**, accepts queries and examines the external and conceptual schemas to determine what conceptual records are required to satisfy the request.
**DBMS Components**

- **File manager**, manipulates the underlying storage files and manages the allocation of storage space on disk.
- **DML preprocessor**, converts DML statements embedded in an application program into standard function calls in the host language.

**DBMS Components**

- **DDL compiler**, converts DDL statements into a set of tables containing metadata.
- **Catalog manager**, manages access to and maintains the system catalog.

**Database Manager**

Components of DM:

- **Authorization control**, checks that the user has the necessary authorization to carry out the required operation.
- **Command processor**, giving user the authority to carry out the operation once the user has the necessary authorization and then control is passed to the command processor.

**Database Manager**

- **Integrity checker**, checks that the requested operation satisfies all necessary integrity constraints.
- **Query optimizer**, determines an optimal strategy for the query execution.
- **Transaction manager**, performs the required processing of operations it receives from transactions.
- **Scheduler**, ensuring that concurrent operations on the database proceed without conflicting with one another.
- **Recovery manager**, ensures that the database remains in a consistent state in the presence of failures.
- **Buffer manager**, responsible for the transfer of data between main memory and secondary storage, such as disk and tape. The buffer manager is sometimes known as the cache manager.