**Database Environment**
- Architectural and functional characteristics of DBMSs

**ANSI-SPARC Model**
- In 1975, at the American National Standards Institute (ANSI)
- Standards Planning and Requirements Committee (SPARC)
- Released model for standard terminology and general architecture for database systems

**Three-Level ANSI-SPARC Architecture**

1. **External Level**
   - users’ view of the database
   - describes that part of the database that is relevant to each user
   - may include different data formats, calculated fields, etc.

2. **Conceptual Level**
   - community view of the database
   - describes *what* data is stored in the database and the relationships
   - all entities, attributes, relationships, constraints, security & integrity information

3. **Internal Level**
   - physical representation of the database on the computer
   - describes *how* the data is stored in the database
   - storage space, record descriptions, record placement, data compression, encryption, etc.
Objectives of 3-Level Architecture
- each user should have access to the data, but with different customized views of it
- Users are immune to changes in other users’ views
- Users should not have to deal with physical storage details
- DBA should be able to change storage structures without affecting users’ views
- Internal structure of database unaffected by changes to physical aspects of storage
- Changes to conceptual structures should not affect users’ views

Database Schema
- Overall description of the database
- Each layer has its own type of schema information
  - External schemas – correspond to different views
  - Conceptual schema – describes all entities, attributes, relationships, constraints
  - Internal schema – complete description of internal model
- Mapping between schemas at each level

<table>
<thead>
<tr>
<th>External</th>
<th>Conceptual</th>
<th>Internal</th>
</tr>
</thead>
</table>
| Reg. Student | Student | struct Student {
| - Id | - Id | int Id; |
| - LName | - LName | char LName[20]; |
| - FName | - FName | char FName[20]; |
| - Year | - Room | struct Room *room; |
| - Program | - Year | int year; |
| - Major | - Program | struct Program *program; |
| Res. Student | - MealPlan | - Major | MajorType major; |
| - Id | - MealPlan | |
| - LName | | index Student::Id; |
| - FName | | |
| - Room | | |
| - MealPlan | | |

Data Independence
- Three level architecture designed so that upper levels are unaffected by change to lower

Logical Data Independence
- Immunity of the external schemas to changes in the conceptual schema
- Changes in entities, attributes, or relationships should not affect views not dealing directly with them

Physical Data Independence
- Immunity of the conceptual schema to changes in the internal schema
- Changes in storage structures, hardware, indexes, hashing, etc.
- Only effect might be a change in performance
Database Languages
- More accurately, sublanguages since do not typically include conditionals or iteration

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
- Result of compilation of DDL statements is a set of tables stored in special files collectively called the system catalog (metadata) or data dictionary

Data Manipulation Language (DML)
- A language that provides a set of operations to support the basic data manipulation operations on the data held in the database
- Insertion of new data
- Modification of existing data
- Retrieval of data – query language
- Deletion of data

Procedural DML
- A language that allows the user to tell the system what data is needed and exactly how to retrieve the data
- Typically used for hierarchical and network data models

Non-procedural DML
- A language that allows the user to state what data is needed rather than how it is to be retrieved
- DBMS translates DML statement into one or more procedures
- Typically used for relational models (SQL, QBE)

Fourth Generation Language (4GL)
- Essentially a shorthand programming language (one line translates to many in 3GL)
- Defines what is to be done, not how to do it

- Presentation languages
  - Query languages – procedural DMLs (SQL, QBE)
  - Report generators – fill in a form to show what to report
  - Forms generator – fill in a form to show what is to be included in special form
  - Graphics generator – permit user to create charts and diagrams
- Specialty languages
  - Spreadsheets
- Application generators
  - Pre-written modules provided as library of functions
- Very High Level Languages
  - Specialized languages to generate application code
Functions of a DBMS

Codd (1982) lists eight services that should be provided by any full-scale DBMS.

1. A DBMS must furnish users with the ability to store, retrieve, and update data in the database.

2. A DBMS must furnish a catalog in which description of data items are stored and which is accessible to users.

3. 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. A DBMS must furnish a mechanism to ensure that the database is updated correctly when multiple users are updating the database concurrently.

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

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

7. A DBMS must be capable of integrating with communication software.

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

In addition to these eight, Connolly & Begg list two additional services that should be provided.

9. A DBMS must include facilities to support the independence of programs from the actual structure of the database.

10. A DBMS should provide a set of utility services.

Summary

1. Data storage, retrieval, and update
2. A user-accessible catalog
3. Transaction support
4. Concurrency control services
5. Recovery services
6. Authorization services
7. Support for data communication
8. Integrity services
9. Services for data independence
10. Utility services
Functions of a DBMS
- Codd’s eight services plus two more

1. Data storage, retrieval, and update
   - fundamental function of a DBMS

2. User-accessible catalog
   - holds data about schemas, users, permissions, applications, etc.

3. Transaction Support
   - Transaction – series of actions carried out by single user or application that change data
   - All or none
   - Avoid inconsistency via rollback

4. Concurrency control services
   - No problem for multiple users to read data
   - Problem when multiple users performing updates
   - Must, in effect, serialize updates

5. Recovery services
   - Recover from system crash, media failure, or h/w or s/w error

6. Authorization services
   - Stop unauthorized access of database
   - Stop authorized users from accessing unauthorized data within database

7. Support for data communication
   - Permit users to access database from other machines or remote locations

8. Integrity services
   - Ensure correctness and consistency via constraints (rules)

9. Services to promote data independence
   - Both physical and logical data independence
   - Cannot change lower level if it affects upper level without first changing upper level

10. Utility services
    - Import facilities – import data into database
    - Monitoring facilities – monitor usage and operation
    - Statistical analysis programs – examine performance and usage statistics
    - Etc.
Components of a DBMS

Query processor
- Major DBMS component that transforms query into series of low-level instructions for database manager

Database Manager (DM)
- Interfaces with user-submitted application programs and queries
- Accepts queries and examines external and conceptual schemas to determine what conceptual records are required to satisfy the request
- Places call to File Manager to perform request

File Manager
- Manipulates underlying storage files and manages storage space
- Maintains list of structures and indexes defined in internal schema
- Uses access methods to read or write data to system buffer (cache)

DML preprocessor
- Converts DML statements embedded in application programs into standard function calls in the host language

DDL compiler - Converts DDL statements into set of tables containing metadata

Dictionary (Catalog) manager - Manages access to and maintains system catalog

Components of the Database Manager

Authorization control - Checks user authority

Command processor - Overall control module of database manager

Integrity checker - Checks if operation that changes database satisfies integrity constraints

Query optimizer - Determines optimal strategy for query execution

Transaction manager - Controls operation of an entire transaction

Scheduler
- Ensures concurrent operations do not conflict
- Controls relative order of transaction operations

Recovery manager
- Ensures database is in a consistent state
- Responsible for transaction commit & abort

Buffer manager - Responsible for data transfer between memory and secondary storage
Multi-User DBMS Architectures

Teleprocessing
- Traditional multi-user system architecture
- Single central computer (mainframe)
- Users working on dumb terminals
- All processing done on central computer (too much burden)

File-Server
- Processing is distributed on network (LAN)
- Each workstation runs its own DBMS
- Files for applications and DBMS stored on server
- Workstations request files from server (shared disk)
- Disadv
  - large amount of network traffic
  - full copy of DBMS running on each station
  - Concurrency, recovery, integrity control complex

Client-Server
- Server holds database and runs DBMS functions
- Clients manage their own user interface and applications
- Database requests transmitted to server
- Results transmitted back to client
- Adv:
  - Wider access to existing database
  - Increased performance – “parallel processing”
  - Possible reduced hardware costs
  - Reduced communication costs – results smaller than files
  - Increased consistency
  - Easier to manage concurrency, recovery, integrity