Databases and Database Management Systems

Types of file:

- Data file
- Text file
- Program file
- Directories (folders)
- Other

Flat file:

- Characteristics:
  - File that contains all the data
  - It is (therefore) not related to another file
- Also called “flat database” or “text database”

Flat-file system (traditional file system i.e. pre-database)

- System based on flat files and different departments having closed systems.
- Advantages:
  - All records are stored in one place (file, table).
  - Easy to set up.
  - Easy to understand.
  - Less hardware and software requirements
  - Best for small databases
- Disadvantages:
  - Potential duplication (redundancy) of data.
  - No primary keys.
  - Harder to update.
  - Potential inconsistencies.
  - Inefficient.
  - Less security.

Database System

- A database:
  - It is a set of related files.
  - It is created and managed by a database management system (DBMS).

    - Can be accessed by a number of authorised users and programs.
- Advantages:
    - Reduced data redundancy.
    - Increased consistency.
    - Greater data integrity and independence from applications programs.
    - Improved data access to users through use of query languages.
    - Improved data security.
- Disadvantages:
    - Database systems are complex, difficult, and time-consuming to design.
    - Substantial hardware and software start-up costs.
    - Damage to database affects virtually all applications programs.
    - Extensive conversion costs in moving form a file-based system to a database system.
    - Initial training required for all programmers and users.

The Database Management System (DBMS)

- The DBMS is software system that controls the organization, storage, retrieval, security and integrity of data in a database.
- It provides an interface between programs, users and the database.
- DBMSs may work with traditional programming languages (COBOL, C, etc.) or they may include their own programming language for application development.
- The major features of a DBMS are:
    - Data Security
        - Prevents unauthorized users (passwords, etc.)
    - Data Integrity
        - Data validation
    - Interactive Query
        - It provides a query language (e.g. SQL) and report writer.

    - Interactive Data Entry and Updating
    - Data Independence (from programs)
        - Data can be accessed by different languages.
        - A change in the formatting of data will not affect the programs accessing the data.

Database Administrator (DBA)

- Person responsible for the database's
    - Physical design
    - Management
    - Evaluation
    - Selection and implementation of the DBMS
- In more detail, it entails some or all of the following:
    - Establishing the needs of users.
    - Monitoring user access.
    - Security (passwords to users, encryption, firewalls, etc.)
    - Monitoring performance and managing parameters to provide fast query responses.
    - Mapping out the conceptual design.
    - Installing and testing new versions of the DBMS.
    - Maintaining adherence to the Data Protection Act.
    - Creating and modifying the database through the use of the DDL.
    - Controlling access privileges.
    - Developing, managing and testing back-up and recovery plans.
    - Ensuring that storage, archiving, back-up and recovery procedures are functioning correctly.
    - Working closely with IT project managers and programmers.
    - Communicating regularly with technical, applications and operational staff to ensure database integrity and security.
    - Commissioning and installing new applications and customising existing applications in order to make them fit for purpose.

Relational Databases

- Most frequently used today.
- It is a collection related tables.
- Relatively easy to create and access.
- Easy to extend.
- Terms:
    - Table, relation, entity
    - Record, tuple, instance, row
    - Field, attribute, column
- Remember that not every user has access to the whole database.
- Keys:
    - A key is a group of attributes for the purpose of working on a table in a RDBMS.
    - Types of keys (as an example consider the following table of primary school students:
        - Super Key: any combination of fields that uniquely identifies each record within the table.
        - Candidate Key: the least combination of fields that uniquely identifies each record in the table.
        - Primary Key: one (and only one) of the candidate keys chosen by the database designer.
        - Composite Key: a key made up of more than one attribute. A primary key can be composite.
        - Foreign Key: a key that forms a relationship.
        - Secondary Key (or Alternative Key): a candidate key that is not the primary key.
    - Example: STUDENTS (ID, index, name, address, class, teacher)
        - Super Key examples:
            - ID + name
            - index + address + class
            - others
        - Candidate Keys
            - ID
            - index

        - Primary Key:
            - index (let us say that this is the one chosen by the designer)
        - Composite Key example:
            - ID + class + teacher
        - Foreign Key:
            - In this example there are no foreign keys since there is only one table.
        - Secondary Key (or Alternative Key):
            - In this example it is ID.
- Relationships
    - Note diagrams 5 and 6. In diagram 6 the tables are related by means of the foreign key ('Course ID' in the table 'Students').
    - 3 kinds of relationships
        - one-to-one
        - one-to-many (or many-to-one)
        - many-to-many
    - Implementation relationships in RDBMS
        - one-to-one (foreign field)
        - one-to-many (foreign field)
        - many-to-many (junction table)
- Notation to represent Relational Databases:
    - Name of table in capital letters
    - Primary key is underlined
    - Foreign keys are written either in italic or with a bar over the fields' names.
- Types of values: e.g. integer, real, text, date, Boolean, memo
- Tools in Database Packages
    - Create a table
    - Create Relationships
    - Referential Integrity (program does not accept a value in a foreign field if it does not exist in the corresponding field of the linked table).
    - Validation (examples):
        - Type: DBMS would not let you enter a string where a number is expected.

        - Presence: DBMS would not let you leave a field empty if the field value is obligatory.
        - Uniqueness
        - Range
        - Format: e.g. field value should consist of three letters followed by three digits.
        - Referential integrity is a form of validation.
    - Queries
        - SQL is a query language
    - Forms (and sub-forms)
    - Reports
- Basic operations on tables:
    - Search
    - Delete
    - Modify
    - Sort
- Data dictionary
    - Defines the organization of the database
- SQL (Structured Query Language)
    - E.g. SELECT ALL WHERE age > 30 AND name = "Smith"
- DML (Data Manipulation Language) examples:
    - INSERT INTO personal_info
      values('bart','simpson',12345, $45000)
    - SELECT *
      FROM personal_info
    - SELECT last_name
      FROM personal_info
    - SELECT *
      FROM personal_info
      WHERE salary > $50000
    - UPDATE personal_info
      SET salary = salary * 1.03
    - UPDATE personal_info
      SET salary = salary + $5000
      WHERE employee_id = 12345
    - DELETE FROM personal_info
      WHERE employee_id = 12345
- DDL (Data Definition Language)

- CREATE DATABASE employees
- CREATE TABLE personal_info (first_name char(20) not null, last_name char(20) not null, employee_id int not null)
- USE employees
- ALTER TABLE personal_info
  ADD salary money null
- DROP TABLE personal_info
- DROP DATABASE employees

Database model

- Determines how the data is organized
- Types
  - Flat model i.e. a single table
    - Advantage: simple
    - Disadvantage: can only be used with small databases
  - Hierarchical model i.e. tree-like structure
    - Advantages
      - Very efficient to describe many relationships in the real world.
      - Simple and easy to design.
    - Disadvantages
      - To suit models that are not hierarchical part of data is duplicated.
      - Implementation is complex (though design is easy)
  - Network model i.e. no restrictions on which tables can be related.
    - Advantages
      - Conceptual simplicity – easy to design.
      - Ease of data access.
    - Disadvantage
      - Complex system
  - Relational model (see above)
  - Object-oriented model
    - Store objects rather than data.
    - Objects contain both executable code and data.

        - Advantages over RDBMS
            - Data model is based on the real world.
            - Less code required when applications are object oriented.
        - Disadvantages compared to RDBMS
            - Lower efficiency when data is simple and relationships are simple.
            - Relational tables are simpler.

Data flow diagram

- A graphical representation of the "flow" of data through an information system.
- It shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored.
- DFDs only involve four symbols. They are:
    - Process
    - Data Object
    - Data Store
    - External entity
- DFDs can be partitioned into levels. Each level has more information flow and data functional details than the previous level.
- Level 0 DFD's (context level DFD)
    - A context level DFD is the most basic form of DFD.
    - It aims to show how the entire system works at a glance.
    - There is only one process in the system and all the data flows either into or out of this process.
    - Context level DFD's demonstrates the interactions between the process and external entities.
    - They do not contain Data Stores.
- Level 1 DFD's
    - Give an overview of the full system.
    - They look at the system in more detail. Major processes are broken down into sub-processes.
    - Identifies data stores that are used by the major processes.

- Level 2 DFD's
  - Each module of a level 1 DFD is decomposed further.
- Strengths
  - DFDs have diagrams that are easy to understand, check and change data.
  - DFDs help tremendously in depicting information about how an organization operations.
  - They give a very clear and simple look at the organization of the interfaces between an application and the people or other applications that use it.

Entity–relationship diagram

- Shows entities (tables) and the relationships between them.

Database normalization:

- Also called data normalization
- A technique to organize the contents of the tables.
- Advantages
  - Reduces Data Duplication
  - Groups Data Logically
  - Relationships are Represented Better
- Disadvantages
  - Slows Database Performance
  - Requires Detailed Analysis and Design
- Terms
  - Functional dependency: when the value of a field 'depends' (can be derived) from the value of another field.
  - Transitive dependency: a non–key attribute is functionally dependent on another non–key attribute.
  - Prime attribute (key attribute)
  - Non–prime attribute (non–key attribute)
  - Full and partial dependencies
  - Denormalization (to quicken performance)
- First normal form

    - A relation is in 1NF if, and only if, it contains no repeating attributes or groups of attributes.
- Second normal form
    - A relation is in 2NF if, and only if, it is in 1NF and every non-key attribute is fully functionally dependent on the whole key.
    - If the primary key is not composite and the table is in the 1NF then the table is also in 2NF.
- Third normal form
    - A relation is in 3NF if, and only if, it is in 2NF and there are no transitive functional dependencies.