Chapter 1: Introduction

Contents

1 Purpose of Database Systems

Conventional operating systems store records in files, forming a *file-processing system*. The advantages of a database system over a file system are:

1) **Data redundancy and inconsistency:**
   a) Various files created over time have different structures.
   b) Application programs may be written in several programming languages.
   c) Same information may be duplicated in several files, records may not match. This is called *data inconsistency*.
2) **Difficulty in Accessing Data:**
   a) With a file system, a query must either be *manually* processed, or an application program must be written.
   b) File systems do not allow data retrieval and querying in an efficient manner.
3) **Data isolation:**
   a) Data scattered in various files and formats.
   b) Difficult to write new applications to retireve data.
4) **Integrity problems:**
   a) Data in a database must satisfy some *consistency constraints*.
   b) Adding new constraints to application programs is difficult.
5) **Atomicity problems:**
   a) For database consistency, an operation must either occur entirely or not at all.
   b) Difficult to ensure atomicity in a file system.
6) **Concurrent-access anomalies:**
   a) For performance and faster response, systems allow multiple users to update the data simultaneously.
   b) Concurrent updates may interact and result in inconsistent data.
   c) To guard against this, the system must maintain some form of supervision. Difficult in file systems.
7) **Security problems:**
   a) Users should have restricted access to prevent tampering of data.
   b) Difficult to enforce restrictions in application programs.

2 View of Data

2.1 Data Models

A **data model** is a collection of conceptual tools for describing data, data relationships/semantics and consistency constraints. They are classified as:

1) **Relational Model:**
   a) Collection of tables (called *relations*) used.
   b) Each table has multiple columns (each with their unique name), correspinding to the attributes of the records.
   c) Each record has a fixed number of attributes.

   d) Most widely used data model.
2) **Entity-Relationship (E-R) Model:**
   a) Uses a collection of basic objects (called *entities*) and *relationships* among those objects.
   b) Most widely used in database design.
3) **Semi-structured Data Model:**
   a) Permits the specification of data where individual items can have different sets of attributes.
   b) Examples are *JSON* and *XML* (eXtensible Markup Language).
4) **Object-Based Data Model:**
   a) Well integrated into relational databases.
   b) Can store objects and procedures to execute in the database system.
   c) Can be seen as extending the relational model with notions of encapsulation, methods, and object identity.

2.2 Data Abstraction

The complexity of the database system is hidden from users through several levels of *data abstraction* as follows:

1) **Physical Level:**
   a) Describes *how* the data are actually stored.
   b) Descirbes complex low-level data structures in detail.
   c) Records can be described as a block of consecutive bytes. Hidden by compilers and database systems from programmers.
   d) *Index* data structure is used to support efficient retrieval of records, part of the physical level.
2) **Logical Level:**
   a) Describes *what* data are stored in the database, and their relationships.
   b) Describes the database in terms of small number of relatively simple structures.
   c) The user of the logical level does not need to be aware of the complexity of underlying physical-level structures. This is called *physical data independence*.
   d) Each record is described by a type definition. The relationships between these types is also defined at this level.
   e) Used by database administrators and programmers.
3) **View Level:**
   a) Highest level of abstraction. Describes only part of the entire database.
   b) Simplifies user interaction with the database system by showing part of the database required. Hides undelying complexities from end users.
   c) Also provides a security mechanism to prevent unrestricted access.

2.3 Instances and Schemas

1) **Instance:** Collection of information stored in the databse at a particular moment.
2) **Schema:** The overall design of the database.
   a) *Physical schema* describes database design at a physical level.
   b) *Logical schema* describes database design at a logical level. Applications are constructed using the logical schema.
   c) *Subschemas* at the view level describe different views of the database.
3) Schemas if poorly created can have problems such as duplication and redundancy.

3 Database Languages

3.1 Data-Definition Language (DDL)

1) Used to specify the database schema and additional properties of the data.
2) *Data storage and definition* language can be used to specify the storage structure and access methods of the data.
3) Database systems implement constraints that can be tested with minimal overhead:
   a) **Domain Constraints:** Constrains the values an attribute can take, by associating a domain with every atribute.
   b) **Referential integrity:** Ensure that a value of a certain attribute(s) appearing in one relation also appears in another relation.
   c) **Authorization:** To differentiate the users based on the type of access. Database systems support *read, insert, update, delete* authorizations for this purpose.
4) DDL outputs are placed in the **data dictionary**, which contains **metadata** (data about data). The data dictionary is a special table that can only be accessed by the database system and not by any user.
5) Format in SQL

```
create table t
(
    <attr1> <domain1> <constraint1>,
    <attr2> <domain2> <constraint2>,
    ... ,
    <addn−constraint1>,
    <addn−constraint2>,
    ...,
);
```

3.2 Data Manipulation Language (DML)

Enables users to access or manipulate data organized by the data model. Various types of accesses are:

1) Retrieval of information
2) Insertion of new information
3) Deletion of information
4) Modification of information

There are two types of DMLs:

Procedural DMLs	**Declarative or Non-procedural DMLs**
Specify *what* data is needed and *how* to get those data.	Specify *what* data is needed *without* specifying how to get those data.
Tougher to learn.	Easier to learn.

Query: Statement requesting the retrieval of information. The part of a DML that deals with information retrieval is called a *query language*.

In SQL,

1) The SQL query language is nonprocedural.
2) A query takes inputs of several (possibly one) table(s) and outputs a single table.

3.3 Database Access from Application Programs

1) Nonprocedural query languages are not Turing complete.
2) Two ways to integrate them using a general-purpose programming language:
   a) *Embedded SQL:* SQL statements embedded in a *host* programming language.
   b) *Application Program Interface (API):* Set of procedures used to send queries to the database from the *host* programming language.
   c) Examples are Open Database Connectivity (ODBC) standard for languages like C, and Java Database Connectivity (JDBC) standard for Java.

4 Database Design

1) Mainly involves design of the database schema.
2) Initial phase: characterize data needs of the users; interact with domain experts and users for this. Leads to a specification of user requirements.
3) *Conceptual-design phase:* Requirements translated into a conceptual database schema. Review to prevent conflicts between requirements. Focus on describing data and their relationships, using the E-R model and various algorithms to quantify attributes.
4) *Specification of functional requirements:* Users describe the kinds of operations that should be supported. Design reviewed for such support.
5) *Logical-design phase:* Map high-level conceptual schema onto the implementation data model of the database system.
6) *Physical-design phase:* Specify physical features of the database.

5 Database Engine

A database system contains various components, described ahead.

5.1 Storage Manager

Provides interface between the low-level data and the application programs/queries submitted to the system. Converts DML statements into low-level file-system commands. Responsible for storing, retrieving and updating data in the database.

Components of the storage manager are as follows:

1) **Authorization and integrity manager:** Checks whether the integrity constraints are satisfied and checks user authority.
2) **Transaction manager:** Ensures the database remains in a consistent state despite system failures, and that concurrent transactions occured without conflicts.
3) **File manager:** Manages allocation of space on disk and data structures used to represent information stored on disk.

4) **Buffer manager:** Fetches data from disk storage, and decides what data to cache in main memory. Critical since it enables databases to handle data sizes much larger than the size of main memory.

Data structures implemented by the storage manager are:

1) **Data files:** Store the database itself
2) **Data dictionary:** Stores metadata of the database.
3) **Indices:** Provide fast access to data items.

5.2 Query Processor

Components of the query processor are as follows:

1) **DDL interpreter:** Interprets DDL statements and records the definitions in the data dictionary.
2) **DML compiler:** Translates DML statements into an evalutaion plan of low-level instructions that the query-evaluation engine understands. Also performs *query optimization i.e.,* picks the lowest cost evaluation plan.
3) **Query evaluation engine:** Executes low-level instructions generated by the DML compiler.

5.3 Transaction Management

Atomicity: Single transaction on a database must happen entirely or not at all. **Consistency:** Database must be correct at any point of time. **Durability:** After successful transaction, the new records must persist, despite system failure. **Transaction:** A collection of operations that performs a single logical function in a database application.

1) Each transaction is a unit of atomicity and consistency.
2) State of consistency must be maintained before and after the transaction.

Transaction manager consists of:

1) **Recovery manager:** Ensures atomicity and durability of the database. Must perform *failure recovery* to detect failures and restore the database to a consistent state.
2) **Concurrency-control manager:** Control interaction among concurrent transactions to maintain the consistency of the database.

6 Database and Application Architecture

To scale up to larger data sizes and higher processing speeds, we have *parallel databases* (run on a cluster consisting of multiple machines) and *distributed databases* (data stored/queries processed across multiple geographically seaparated machines).

Database applications can be partitioned into two or three parts:

1) *Two-tier architecture:* Application resides at the client machine and queries the server machine.
2) *Three-tier architecture:* Used by modern applications. Frontend communicates with the *application server*, which has *business logic* embedded in it (as opposed to deploying it on various clients). Provides better security and performance.

7 Database Users and Administrators

There are four types of database users based on the way they interact with the system:

1) **Naive users:**
   a) Unsophisticated users, interact using the predefined interfaces.
   b) Typically fill forms and read reports generated by the database.
2) **Application programmers:** Develop user interfaces by writing application programs using various tools/frameworks.
3) **Sophisticated users:**
   a) Interact with the system without writing programs.
   b) Form requests using query languages or data analysis software, mainly to explore the data in the database.
4) **Database administrator (DBA):** Has central control over the system. Functions are:
   a) *Schema definition:* Execute DDL to create the original database schema.
   b) *Storage structure and access-method definition:* Specify parameters regarding physical organization of the data and generated indices.
   c) *Schema and physical-organization modification:* Change the schema and physical organization to suit changing needs of the organization, or to improve performance.

d) *Granting authorization for data access:* Regulate user access to appropriate parts of the database.
e) *Routine maintenance:* Periodic backup of data onto remote servers, keeping a check on free disk space, monitoring jobs running on the database to maintain performance.