Chapter 1. Introduction

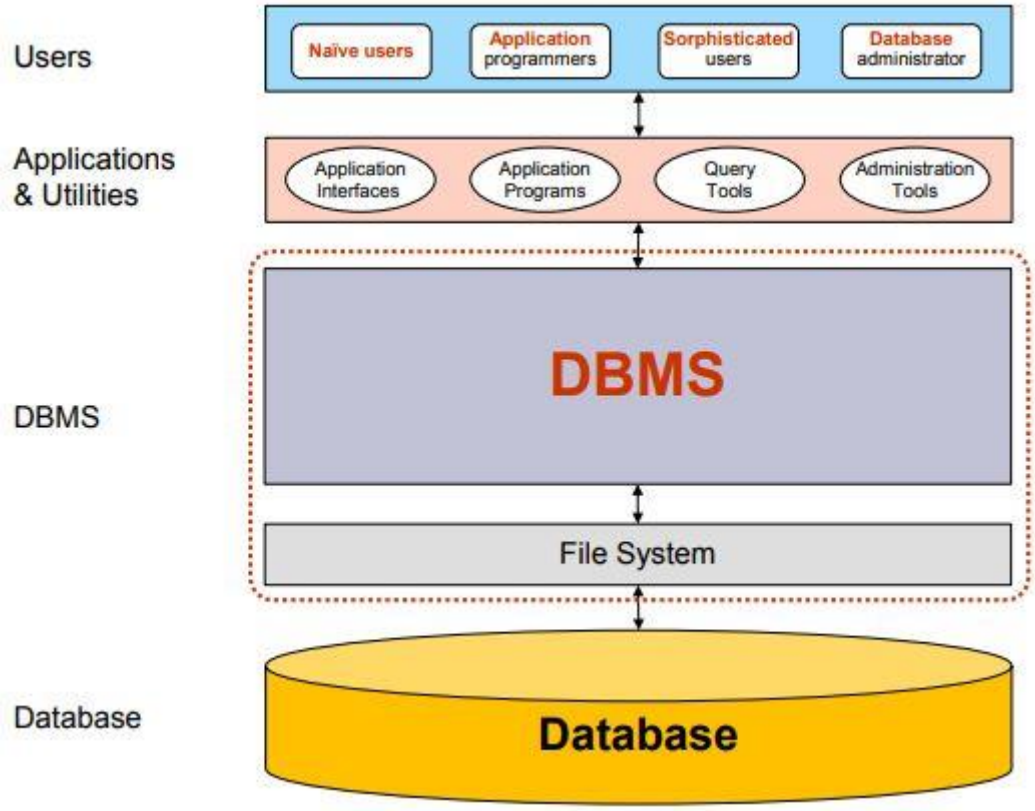
1.1 Database-System Applications

Databases, which refer to a collection of persistent, interrelated, and integrated data that is used by the application systems of some given enterprise, are used everywhere. Although most people might not care so much, databases touch all aspects of our daily life. Applications are:

- Universities: student information, course lists, grades, department information…
- Bank: customer information, loans, accounts, stocks, transactions…

1.2 Purpose of Database Systems

Database System consists of the users, database applications, DBMS(Database-management System), and databases.



Before the invention of DBMS, data were managed by using file system. The system stores permanent records in various files, and it needs different application programs to extract records from, add records to the appropriate files. This system is known to have a number of major drawbacks below. To cope with those problems, DBMS was designed.

- **Data redundancy and inconsistency**

  Same information may exist in several files. For instance, I have a double major (Political Science and Computer Science), my student info will be stored in Political Science department file and Computer Science department file. It naturally brings inconsistency problem among duplicated information stored in different files. If my phone number has been changed, it may be reflected in the CS dept, but not in the PS dept.

- **Difficulty in accessing data**

  Conventional file-processing environments do not allow requested data to be retrieved in a convenient and efficient manner. Data retrieval in specific condition is difficult if there is no application executing that condition check. More responsive data-retrieval systems are required for general use.

- **Data isolation**

  Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

- **Integrity problems**

  The data values must satisfy certain types of consistency constraints. For example, balance in an account should be greater than or equal to 0.

- **Atomicity of updates**

  If any failure occurs while transaction, the data should be restored to the state before that transaction. It must happen in its entirety or not at all. That's why we call it atomicity. It is hard to ensure atomicity in a file system.

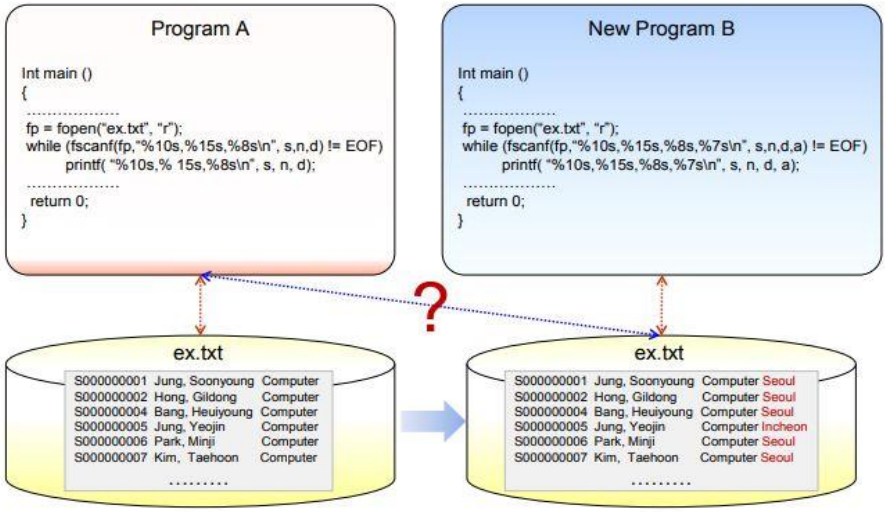
- **Concurrent-access by multiple users**

  Many people are trying to update the data simultaneously. Sujin has 500 won in her account. Credit card company A and B confirms that Sujin has 500 won in her account, so tries to transfer 500 won each at the same time, the amount she used in the previous month. Database must control this situation to prevent anomalies in the data. But it is difficult for a file system to offer such control.

- **Security problems**

Not every user of the database system should be able to access all data. Since application programs are added to the file system in an ad hoc manner, enforcing such security constraints is difficult.

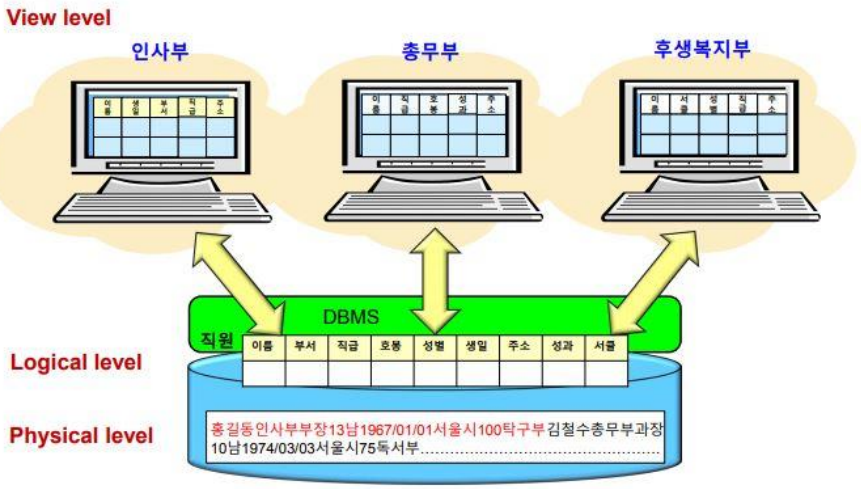
- **Data dependence problem**



1.3 View of Data

1.3.1 Data Abstraction

Users in Database System are different kind of people and have different purposes accessing database. To simplify users' interaction with the system, there are 3 levels of data abstraction.



- **Physical level**: how the data are actually stored.
- **Logical level**: what data are stored in the database, what relationships exist among those data. Database administrators, who must decide what information to keep, use this level.
- **View level**: Many users of the database system do not need all the information stored. Only part of the entire information is described.

1.3.2 Instances and Schemas

Schema – the logical structure of the database(analogy: type of a variable)

- Physical schema: the database design at the physical level
- **Logical schema**: database design at the logical level. Programmers construct applications by using this schema.
- Subschemas: database schemas at the view level

Instance – the actual content of the database at a certain point in time(analogy: value of a variable)

Physical Data Independence – the ability to modify the physical schema without changing the logical schema.

1.3.3 Data Models

A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to properties of the real world entities.

A collection of conceptual tools for describing:

- **Data(structure)**
- **Data relationships**
- **Data semantics**
- **Consistency constraints**

cf. Operators

There are 4 categories where the data models can be classified:

- **Relational Model**

  uses a collection of tables(relations) to represent both data and the relationships among those data.

- **Entity-Relationship Model(E-R Model)**

  uses a collection of basic objects(entities, relationships) among these objects, mainly for database design.

- **Object-Based Data Model**

  seen as extending E-R model with notions of encapsulation, methods, and object identity.

- **Semistructured Data Model**

  permits the specification of data where individual data items of the same type may have different sets of attributes. XML(Extensible Markup Language) is widely used.

1.4 Database Languages

1.4.1 Data-Manipulation Language(DML)

Data-Manipulation Language is a language for accessing and manipulating the data organized by the appropriate data model.

- Procedural DML: user specifies ***what*** data are needed and ***how*** to get those data.
- Nonprocedural DML(Declarative): user only specifies ***what*** data are needed.

The portion of a DML that involves information retrieval is called a query language.

1.4.2 Data-Definition Language(DDL)

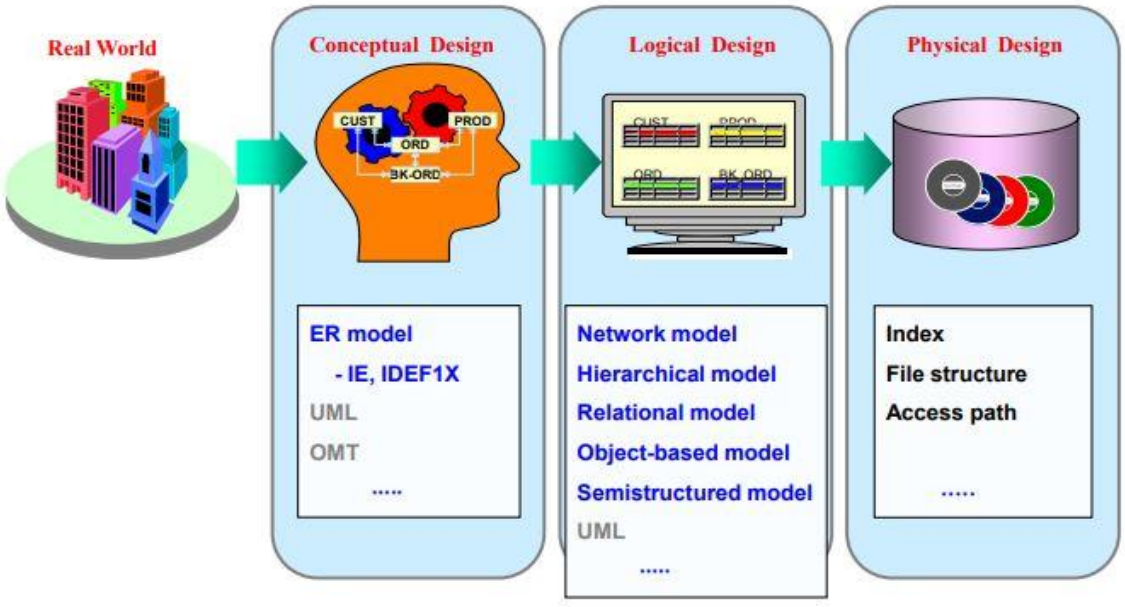
Language to specify a database schema by a set of definitions. The data values stored in the database must satisfy certain consistency constraints. We can set the implementation of a number of integrity constraints below.

- Domain Constraints: A domain of possible values must satisfy every attribute(for example, integer types, character types, date/time types).

- Referential Integrity: There are cases where we wish to ensure that a value that appears in one relation for a given set of attributes also appears in a certain set of attributes in another relation. The value we are looking for must appear in the referenced relation.
- Assertions: Any condition that the database must always satisfy. Above two are also special forms of assertions.
- Authorization: read/insert/update/delete authorization is given differently to the users.

DDL compiler generates a set of table templates stored in a data dictionary, which contains metadata.

1.6 Database Design



1.7 Data Storage and Querying

A database system can be broadly divided into two parts, the storage manager and the query processor components.

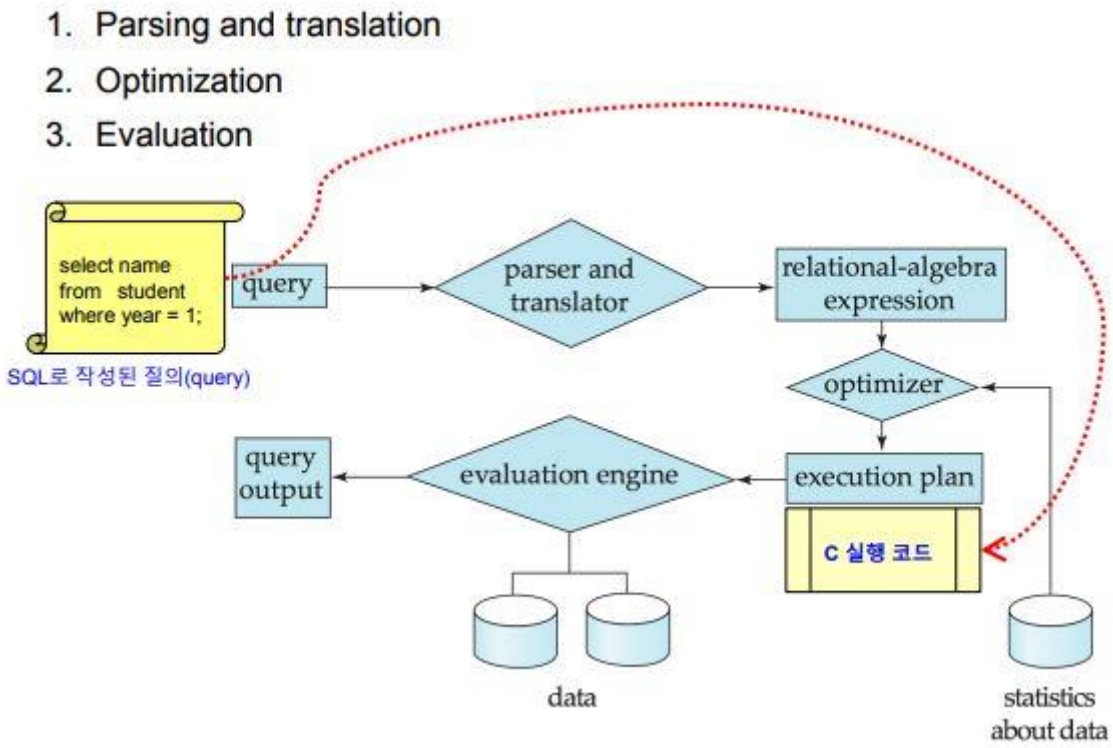
1.7.1 Storage Manager

The storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system. As raw data are stored on the disk using the file system provided by the operating system, the storage

manager translates DML statements into low-level file-system commands.

1.7.2 The Query Processor

- DDL interpreter: interprets DDL statements and records the definitions in the data dictionary.
- DML complier: translates DML statements(query) into optimized low-level instructions.
- Query evaluation engine: executes low-level instructions generated by DML compiler.

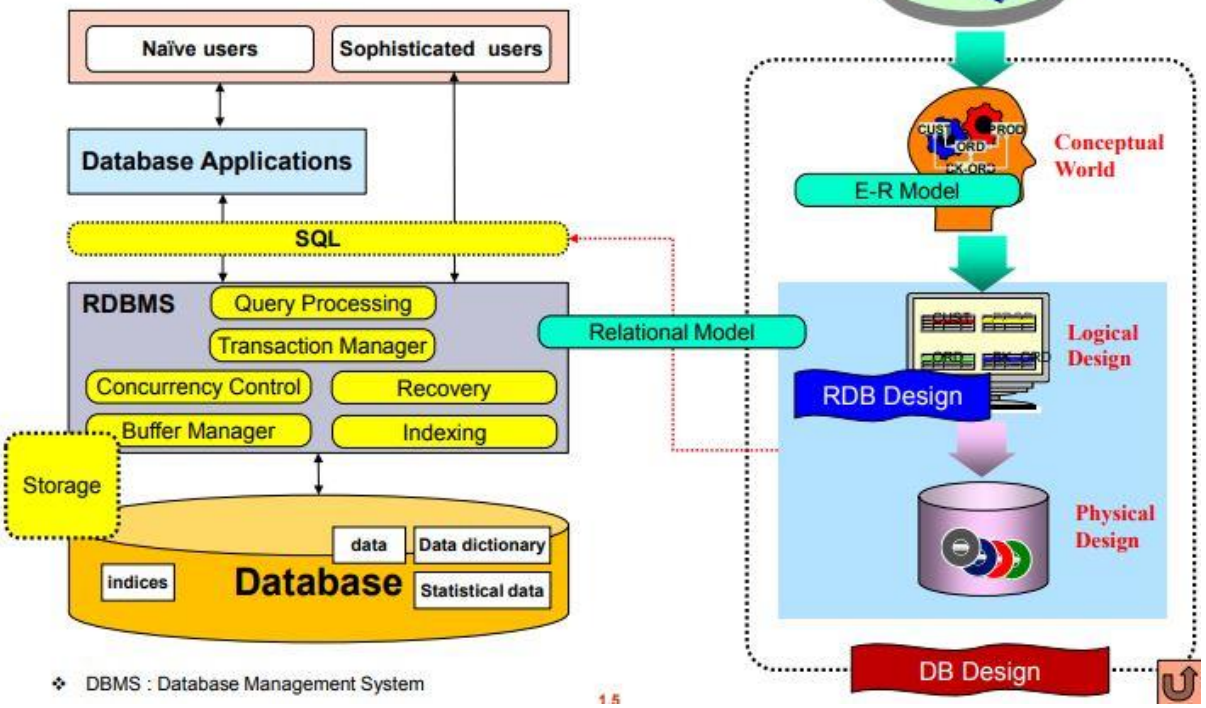


1.8 Transaction Management

A transaction is a collection of operations that performs a single logical function in a database application. Each transaction is a unit of both atomicity and consistency.

- **Programmer's job**: define transactions properly
- **Database system(transaction manager)'s job**: ensure the atomicity and durability(recovery manager), and concurrency(concurrency-control manager)

Database Systems & DB Design



❖ DBMS : Database Management System