# CHAPTER 1 Introduction

## 1.1 Database-System Applications

***Database-management system (DBMS):*** collection of interrelated data, called ***database***, and a set of programs to access those data.

***Main goal of DBMS***: provide a way to store and retrieve database information ***conveniently*** and ***efficiently***.

Database systems designed to manage large bodies of information.

***Some representative applications:***

|  |  |
| --- | --- |
| *Enterprise Information* | * *Sales*: customer, product, purchase info * *Accounting*: payments, receipts, account balances * *Human resources*: info about employees, salaries, * *Manufacturing*: management of supply chain, for tracking production of items in factories, inventories, orders. * *Online retailers*: sales data, order tracking |
| Banking and Finance | * *Banking****:*** customer info, accounts, loans * *Credit card transactions:* purchases, monthly stat. * *Finance:* storing info about holdings, sales |
| *Universities* | student info, course registrations, grades |
| *Airlines* | reservations, schedule info |
| *Telecommunication* | keeping records of calls, monthly bills |

## 1.2 Purpose of Database Systems

***Disadvantages*** ***of file-processing system***

|  |  |
| --- | --- |
| ***Data redundancy and inconsistency*** | diff programmers creates files they can have diff structures and in different programming languages + same information can be duplicated. redundancy leads to higher storage and access cost. It may lead to data inconsistency, various copies of same data may no longer agree. |
| ***Difficulty in accessing data*** | conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval are required for general use |
| ***Data isolation*** | data are scattered, files in diff formats, writing new programs to retrieve appropriate data is difficult |
| ***Integrity problems*** | data values stored in the DB must satisfy certain types of  **consistency constraints**. Developers enforce these constraints in the syst by adding appropriate code. When new constraints are added, it is difficult to change programs to enforce them. |
| ***Atomicity problems*** | A comp syst is subject to failure. It is crucial that, if failure, data to be restored to the consistent state that existed prior to the failure. Atomic: must happen entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing syst. |
| ***Concurrent-access anomalies*** | Many syst allow multiple users to update the data simultaneously. Interaction of concurrent updates is possible and may result in inconsistent data. |
| ***Security problems*** | Not every user of the DBS should be able to access all the data. |

## 1.3 View of Data

A major ***purpose*** ***of the DBS*** is to provide users with an abstract view of the data.

### 1.3.1 Data Abstraction

Developers hide complexity from users through several levels of abstraction, to simplify users’ interactions with the syst.

|  |  |
| --- | --- |
| ***Physical Level*** | * lowest level of abstraction describes how data are actually stored * It describes complex low-level data structures in detail. |
| ***Logical Level*** | * the next-higher level of abstraction describes what data are stored in the DB, and what relationships exist among those data. * it describes the entire DB in terms of small number of simple structures. * physical data independence * DB admin use logical level of abstraction |
| ***View Level*** | * highest level of abstraction describes only part of entire DB * complexity remains because of the variety of info stored in a large DB * It exists to simplify their interaction with syst. |

### 1.3.2 Instances and Schemas

***Instance***: collection of info stored in DB at a particular moment

***Schema***: overall design of DB

A DB schema = variable declarations in a program. Each variable has a particular value at a given instant.

Instance of DB schema = values of variables in a program at a point in time.

### 1.3.3 Data Models

A Data model provides a way to describe the design of a DB at the physical, logical, and view levels.

|  |  |
| --- | --- |
| ***Relational Model*** | * it uses collection of tables to represent data and relationships among data * each table has many columns, each column has a unique name. ***s*** * most widely used data model |
| ***Entity-Relationship Model*** | * it uses a collection of basic obj, called entities, and relationships among these obj. * entity ***= thing/obj in the real world that is distinguishable from other object*** * widely used in DB design |
| ***Object-Based Data Model*** | * extending E-R model with notions of encapsulation, methods/functions, obj identity * it combines festures of obj-oriented data model and relational data model |
| ***Semistructured Data Model*** | * it permits specification of data where individual data items of same type may have diff sets of attributes. * XML used to represent semistructure data |

Network data model and hierarchical data model preceded the relation data models. They were tied closely to the underlying implementation, and complicated the task of modelling data => little used now.

## 1.4 Database Languages

A DBS provides a ***data-definition language*** to specify the DB schema and a ***data-manipulation language*** to express DB queries and updates.

### 1.4.1 Data-manipulation language

it is a language that enables users to access/manipulate data as organized by appropriate data model.

|  |  |  |
| --- | --- | --- |
| **Types of access** | | * retrieval of info stored in DB * insertion of new info into DB * deletion of info from DB * modification of info store in DB |
|  | |  |
| **Two types** | * ***Procedural DMLs*** require user to specify *what* data are needed and *how* to get those data * ***Declarative DMLs/nonprocedural DMLs*** require user to specify *what* data are needed *without* specifying how to get | |

***Query***: statement requesting retrieval of info

***Query language***: portion of DML that involves info retrieval

### 1.4.1 Data-definition language

We specify a DB schemas by a set of definitions expressed by a special language, data-definition language. DDL is used to specify additional properties of data. We specify storage structure and access methods used by DBS by ***data storage and definition language.*** The data values store in DB must satisfy certain ***consistency constraints***.

|  |  |
| --- | --- |
| ***Domain constraints*** | * a domain of possible values must be associated with every attribute * declaring an attribute to be of a particular domain acts as a constraint on the values * they are most elementary form of integrity constraint |
| ***Referential Integrity*** | * DB modification can cause violations of referential integrity * when it is violated, normal procedure is to reject the action that caused it |
| ***Assertions*** | * an assertion is any condition that DB must always satisfy * domain constraints and referential-integrity constraints are special forms of assertions * if it’s valid, then any future modification to DB is allowed only if it doesn’t cause that assertion to be violated |
| ***Authorization*** | * differentiate users to type of access they are permitted on various data values in DB * read, insert, update, delete authorization. |

Output of the DDL is placed in the ***data dictionary***, which contains ***metadata*** (data about data). Data dictionary is considered to be a special type of table that can only be accessed and updated by the DBS itself. The DBS consults the data dictionary before reading or modifying the actual data.

## 1.5 Relational Databases

A ***relational DB*** is based on the relational model and uses a collection of tables to represent both data and relationships among those data.

### 1.5.1 Tables

Each table has multiple columns and each column has a unique name. The relational model is an example of a record-based model.

Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of table correspond to attributes of record type.

### 1.5.2 Data-Manipulation Language

SQL query language is nonprocedural. Query takes as input several tables and always returns a single table.

## 1.6 Database Design

### 1.6.1 Design Process

A high-level data model provides DB designer with conceptual framework that specifies data requirements of DB users, an dhow DB will be structured to fulfil these requirements.

**Phase 1**: characterize fully the data needs of prospective DB users. Designer needs to seek domain experts and users. Outcome= specification of user requirements.

**Phase 2**: choose the data model and translates requirements into conceptual schema of DB. It provides a detailed overview of the enterprise. Watch for redundant features. Focus on describing data and their relationships.

**Logical-design phase**: designer maps high-level conceptual schema onto implementation data model of DBS and uses resulting system-specific DB schema in the physical-design phase.

**Physical-design phase**: physical features are specified.

### 1.6.3 Entity-relationship Model

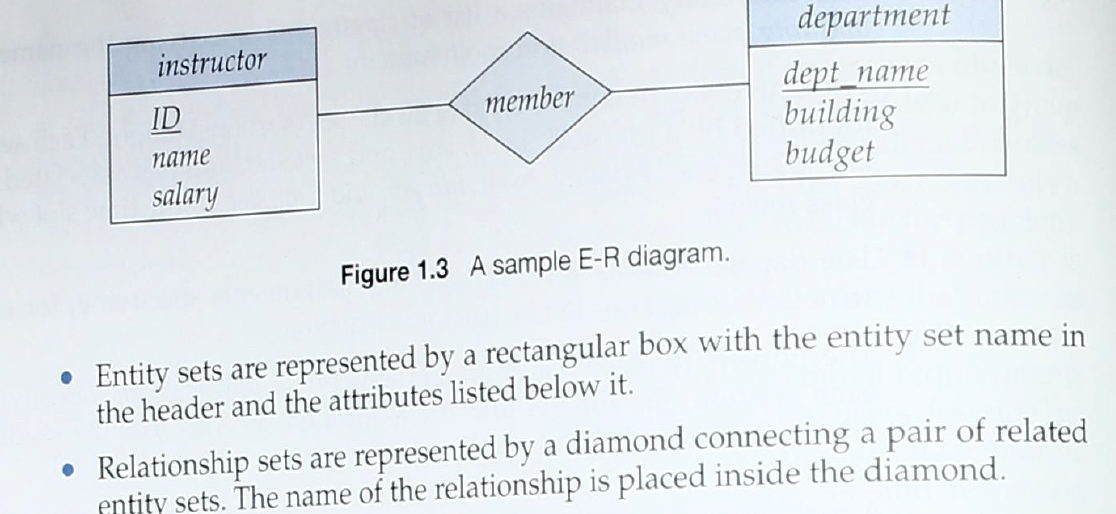
It uses a collection of basic obj, called ***entities and relationships*** among these objs. ***Entity***: set of attributes.

***Relationship***: association among several entities.

***Entity set***: set of all entities of same type

***Relationship set***: set of all relationships of same type

Overall logical structure (schema) of DB can be expressed graphically by entity-relationship (***E-R) diagram***.

Most popular is: UML

EX:

E-R model represents certain constraints to which contents of DB must conform. One important is ***mapping cardinalities***, which express nb of entities to which another entity is associated via a relationship set.

### 1.6.4 Normalization

Another method for designing relational DB is to use normalization.

***Goal***: generate a set of relation schemas that allows us to store info without unnecessary redundancy, yet allows us to retrieve info easily.

To determine whether a relation schema is one of desirable normal forms, need additional info about real-world enterprise we are modelling with DB => use ***functional dependencies.***

|  |  |
| --- | --- |
| What can go wrong in a bad DB design? | * repetition of info   waste of space; complicates updating Db; updates are costly; must ensure every tuple pertaining is updated   * inability to represent certain info   solution: introduce ***null values***. They indicates that value doesn’t exist or unknown. Unknown value may be either missing or not known. Hard to handle. |

## 1.7 Data Storage and Querying

DBS is partitioned into modules that deal with each of the responsibilities of overall syst Fctional components of DBS can be divided into storage manager and query processor components.

### 1.7.1 Storage Manager

***Storage manager:*** component of DBS that provides interface btw low-level data stored in DB and application programs and queries submitted to the syst.

It is ***responsible*** for interaction with ***file manager***. It ***translates*** various DML statements into low-level file-syst commands. ***It is responsible for storing, retrieving and updating data in the DB***.

|  |  |  |  |
| --- | --- | --- | --- |
| Storage manager components | | | |
| ***Authorization and integrity manager*** | ***Transaction manager*** | ***File Manager*** | ***Buffer Manager*** |
| * tests satisfaction of integrity constraints * checks authority of users | * ensures DB consistent state despite failures * concurrent transaction execution proceed without conflicting | * manages allocation of space on disk storage * manages data struct. used to represent info stored on disk | * responsible for fetching data from disk storage into main memory * deciding what data to cache in main memory |

|  |  |  |
| --- | --- | --- |
| Store manager implements | | |
| ***Data files*** | ***Data dictionary*** | ***Indices*** |
| * stores DB itself | * stores metadata about structure of DB (schema of DB) | * provides fast access to data items |

### 1.7.2 Query Processor

|  |  |
| --- | --- |
| Query processor components | |
| ***DDL interpreter*** | ***DML compiler*** |
| * interprets DDL statements and records def in data dictionary | * translates DML statements in query language into evaluation plan consisting of low-level instructions that ***query evaluation engine*** (executes low-level instructions ) understands. * ***query optimization***: picks lowest cost evaluation plan |

## 1.8 Transaction Manager

Often, several operations of DB form a single logical unit of work.

ex funds transfer, account A is debited account B is credited. The funds transfer must happen in its entirety or not at all. (***Atomicity***: all-or-none requirement). Also it is essential that execution of funds transfer preserve the consistency of DB. So the value of the sum of balances of A and B must be preserved. (***Consistency***) After successful execution, new values of balances of accounts A and B must persist, despite possibility of failure. (***Durability:*** persistence requirement)

***Transaction***: collection of operations that performs a single logical fct in DB application. Each transaction is unit of atomicity + consistency. Must not violate any DB constraints.

It’s programmer’s responsibility to define properly various transactions, so each preserves consistency of DB. Ensuring atomicity and durability properties is the responsibility of DBS itself (***recovery manager***).

If transaction failed, it must have no effect on state of DB. DB must be restored to state it was before the transaction = ***failure recovery.*** It is the responsibility of ***concurrency-control manager*** to control interaction among concurrent transaction to ensure consistency of DB. ***Transaction manager*** consist of concurrency-control manager and recovery manager.

## 1.9 Database Architecture

Architecture of DBS is influenced by computer syst on which DBS runs. DBS can be centralized or client-server (one server machine executes work on behalf of multiple client machines) or parallel.

DB applications partitioned into 2-3 parts. In a ***two-tier architecture***, application resides at client machine, where it invokes DBS fctionality at server machine through query language statements. In a ***three-tier architecture***, client machine acts as a front end and doesn’t contain any direct DB calls. Client end communicates with application server, through forms interface.

## 1.10 Data Mining and Information Retrieval

***Data mining***: process of semiautomatically analysing large DB to find useful patterns. It attempts to discover rules and patterns from data.

***Machine learning***: knowledge discovery in artificial intelligence

## 1.11 Specialty Databases

### 1.11.1 Object-Based Data Models

Dominant software-dev methodology; can be seen as extending E-R model with notions of encapsulations, methods/fcts, obj entity.

***Object-relation data model***: combines features of obj-oriented data model and relational data model. It extends traditional relation model with variety of features.

### 1.11.2 Semistructured Data Models

It permits specification of data where individual data items of same type may have diff set of attributes. In contrast with data models where every data item of particular type must have same set of attributes.

## 1.12 Database Users and Administrators

### 1.12.1 DB Users and User Interfaces

|  |  |
| --- | --- |
| Naïve users | Unsophisticated users who interact with syst invoking one of application programs that have been written previously. |
| Application programmers | * computer professionals who write application programs. * can choose from many tools to dev. user interfaces * RAD tools |
| Sophisticated users | * interact with syst without writing programs * they form their requests either using a DB query languages or by using tools |
| Specialized users | * sophisticated users who write specialized DB apps that don’t fit into traditional data-processing framework. |

### 1.12.2 DB Admin

***DB Admin (DBA):*** person to have central control of both data and programs that can access those data.

|  |  |
| --- | --- |
| DBA functions | |
| ***Schema definition*** | DBA creates original DB schema by executing set of data def statements in DDL |
| ***Storage structure and access-method definition*** | |
| ***Schema and physical-organization modification*** | DBA carries out changes to schema and physical organization to reflect changing needs to organization or improve performance. |
| ***Granting authorization for data access*** | DBA can regulate which parts of DB various users can access. |
| ***Routine maintenance*** | Backing up DB; ensuring enough free disk space; monitoring jobs running on DB |

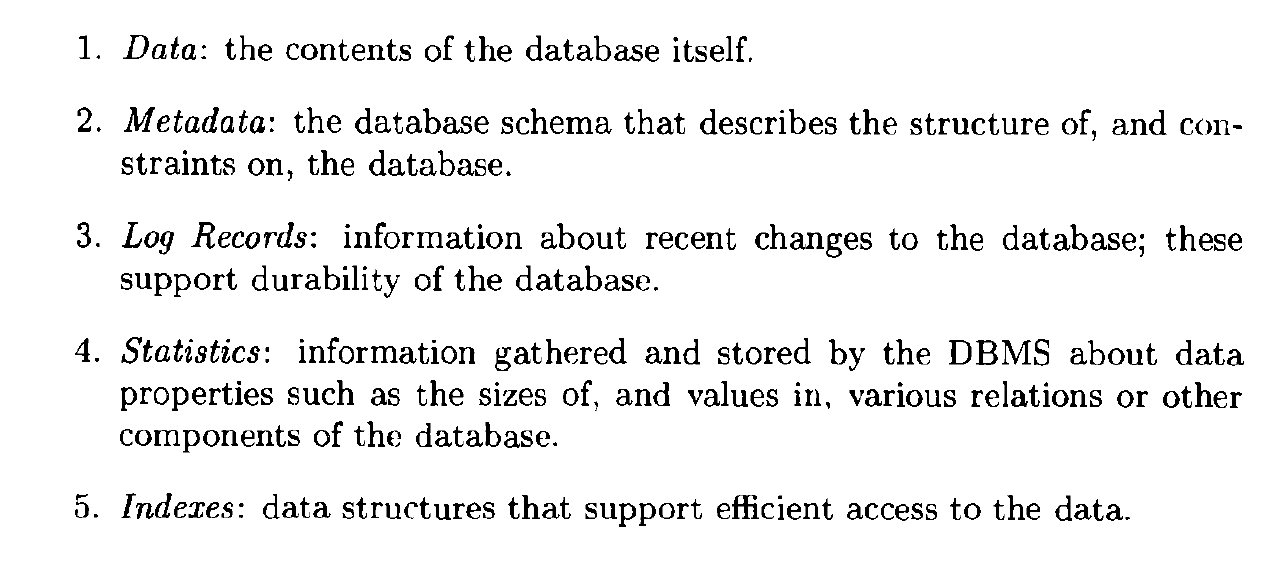
# Book:

|  |
| --- |
| **Answer a query**  query is parsed and optimized by a query compiler -> resulting query plan/sequence of actions the DBMS will perform to answer the query, is passed to the execution engine -> execution engine issues a sequence of requests for small pieces of data to a resource manager that knows about data -> requests for data are passed to the buffer manager -> buffer manager's task is to bring appropriate portions of the data from secondary storage (disk) where it is kept permanently, to the main-memory buffers. -> buffer manager communicates with a storage manager to get data from disk. |

|  |
| --- |
| **Transaction Processing**  Queries and other DML actions are grouped into transactions.  Transaction: units that must be executed atomically and in isolation from one another.  Any query/modification action can be a transaction by itself.  The execution of transactions must be durable  durable: effect of any completed transaction must be preserved even if system fails.  transaction processor into 2 parts:  1. concurrency manager control/scheduler: responsible for assuring atomicity and isolation of transaction  2. logging and recovery manager: responsible for durability of transactions. |

To perform any useful operation on data, that data must be in main memory. It is the job of the storage manager to control the placement of data on disk and its movement between disk and main memory.

For efficiency purposes, DBMS's normally control storage on the disk directly, at least under some circumstances. The storage manager keeps track of the location of files on the disk and obtains the block or blocks containing a file on request from the buffer manager. The buffer manager is responsible for partitioning the available main memory into buffers, which are page-sized regions into which disk blocks can be transferred.



The transaction manager therefore accepts transaction commands from an application, which tell the transaction manager when transactions begin and end, as well as information. The transaction processor performs the following tasks:

|  |
| --- |
| 1. ***Logging***: In order to assure durability, every change in the database is logged separately on disk. The log manager follows one of several policies designed to assure that no matter when a system failure or "crash" occurs, a recovery manager will be able to examine the log of changes and restore the database to some consistent state.  2. ***Concurrency control***: Transactions must appear to execute in isolation. Thus, the scheduler (concurrency-control manager) must assure that the individual actions of multiple transactions are executed in such an order that the net effect is the same as if the transactions had in fact executed in their entirety, one-at-a-time. A typical scheduler does its work by maintaining locks on certain pieces of the database. These locks prevent two transactions from accessing the same piece of data in ways that interact badly. Locks are generally stored in a main-memory lock table.  3. ***Deadlock resolution***: As transactions compete for resources through the locks that the scheduler grants, they can get into a situation where none can proceed because each needs something another transaction has. The transaction manager has the responsibility to intervene and cancel ("rollback or "abort") one or more transactions to let the others proceed. |
| The portion of the DBMS that most affects the performance that the user sees is the query processor.  Query processor is in 2 components:  1) ***query compiler***, which translates the query into an internal form called a query plan. The latter is a sequence of operations to be performed on the data. The query compiler consists of three major units:  (a) ***query*** ***parser***, which builds a tree structure from the textual form of the query.  (b) ***query preprocessor***, which performs semantic checks on the query (e.g., making sure all relations mentioned by the query actually exist), and performing some tree transformations to turn the parse tree into a tree of algebraic operators representing the initial query plan.  (c) ***query optimizer***, which transforms the initial query plan into the best available sequence of operations on the actual data.  The query compiler uses metadata and statistics about the data to decide which sequence of operations is likely to be the fastest. |

2) ***execution engine***, which has the responsibility for executing each of the steps in the chosen query plan. The execution engine interacts with most of the other components of the DBMS.