

Introduction

Database

- **Data:** Facts.
- **Information:** Processed facts. Reveals the meaning of data.
- **Metadata:** Data about data.
Example: The database was created on the 5th of September 1980.
- **Database:**
 - Collection of data central to some enterprise
 - Centralized on one computer or distributed over several, perhaps widely separated geographically
 - Essential to operation of enterprise (Contains the only record of enterprise activity)
 - An asset (Historical data, of interest to other enterprises)
 - State of database mirrors state of enterprise

File-processing system

- Ancestor of Database Management System.
- File systems are supported by conventional operating systems.
- FS store records in various files and need different application programs to extract from and add records to the appropriate files.

Database management system

- A DBMS is a set of interrelated data and a set of programs to access the data. It is an environment that is convenient and efficient (space and time) to use.
- DBMS support high-level access language (e.g. SQL).
DBMS describe database accesses using data manipulation language (e.g. SQL).
DBMS interpret statements of language to perform requested database accesses.
- Examples of DBMS: Oracle, Sybase, SQL Server, Access...

Systems: Then and now

- Relational model \Rightarrow high level view of data.
 - Older systems had a low level view of data.
- Might contain multimedia data.
 - Older systems contain only alphanumeric data.
- On-line databases accessed at time of event.
- Concurrent systems (Multiple transactions execute simultaneously).
 - Older systems were only sequential.
- Distributed data.
 - Older systems centralized data.
- Heterogeneous data.
 - Older systems dealt with homogeneous data.
- Accessed by everyone.
 - Older systems required trained users.

Systems Requirements

- High availability (On-line)
- High reliability (Correctly tracks state, does not lose data, controlled concurrency)
- High throughput (many users \Rightarrow many transactions/sec)
- Low response time (On-line \Rightarrow users are waiting)
- Long life (Complex systems are not easily replaced)
- Fault tolerance (Data integrity -later- despite hardware and software malfunctions)
- Security (Sensitive data must be carefully protected since system is accessible to many users. Users must be authenticated)
- Attractive user interface

Transaction

- Action (program) on the database that changes the state of the database.
- A transaction has some particular properties
 - **A**tomicity
 - **C**onsistency
 - **I**solation
 - **D**urability

ACID

- Database/State 1 $\xrightarrow{\text{Transaction}}$ Database/State 2
- Examples:
 - Your check-in at an hotel involve a transaction with the hotel reservation database.
 - Your flight arrangements involves a transaction with the airline's reservation database.
- The transaction processing system:
Transaction \rightarrow DBMS \rightarrow Database

Integrity constraints

- The database state must satisfy certain properties.
- The occurrence of certain real world events are limited.
- Examples:
 - The balance of a bank account may never fall below \$25.
 - The age of a person is not smaller than 0 and greater than 130 years.
 - Students cannot register for a course if current number of registrants = maximum allowed.

Consistency / Correctness

- Property of a database.
- The database reflects the real world.
- Assuming the database is in a state that satisfies the integrity constraints, after the processing of a transaction:
 - All integrity constraints are satisfied in the new database.
 - The transaction has been executed.

Atomicity

- The system must ensure that the transaction either runs to completion or if it does not complete has no effect at all (as if it has never started).

We say respectively that the transaction is either committed or aborted.

- Note: Ordinary programs do not have the property of atomicity. A crash could leave files partially updated on recovery.
- Example: A student either registers or does not register.

Durability

- The system must ensure that once a transaction commits its effect on the database state is not lost in spite of subsequent failures.
- Note: Not true for ordinary programs.

Isolation

- Deals with execution of multiple transaction concurrently.
- Sequential/Serial execution of transactions:
One transaction is executed to completion before another is started.
Advantage: Preserves consistency/correctness.
Disadvantage: delays.
- Concurrent execution of transactions:
Different transactions are interleaved in time.
Advantage: Performance benefits.
Problem: It may not be correct/consistent (See the next example).

Example

- Bank account. Consider a balance of \$500.
Transaction T1: withdraw \$50 from the account.
Transaction T2: withdraw \$100 from the account.
T1 and T2 are executed concurrently.
- One schedule may be:
T1 reads the balance.
T2 reads the balance.
T1 withdraws \$50 from \$500.
T2 withdraws \$100 from \$500.
The final balance is: \$400.
The database state no longer corresponds to the real-world state.
- Another schedule may be:
T1 reads the balance.
T2 reads the balance.
T2 withdraws \$100 from \$500.
T1 withdraws \$50 from \$500.
The final balance is: \$450.
The database state no longer corresponds to the real-world state.

- Another schedule may be:

T1 reads the balance.

T1 withdraws \$50 from \$500.

T2 reads the balance.

T2 withdraws \$100 from \$450.

The final balance is: \$350.

The database state corresponds to the real-world state.

Isolation

- Even though transactions are executed concurrently the overall effect of the schedule must be the same as if the transaction had been executed serially in some order. We say that the execution is serializable.
- Serializable is better than serial from a performance point of view but often cannot be used.

FS versus DBMS

- Drawbacks of using file systems:
 - Data redundancy
 - Data inconsistency
 - Data isolation
 - Integrity problems
 - Atomicity of updates
 - Concurrent access by multiple users
 - Security problems

Data Redundancy

- Duplication of data.

Storage problems.

Example: If a customer of a bank has 2 accounts, his address and phone number do not need to be stored twice.

- Data that can easily be deduced from other data.

Example: *age* can be easily deduced from the *data of birth*.

Data inconsistency

- Various copies of the same data may no longer agree.

Example: If a customer of a bank changes his address, this update must be registered in all places.

Data isolation

- Data are in different files and in different formats.
- There are difficulties and inefficiency of accessing data
- New applications have to be written for unanticipated problem. This requires maintaining permanent staff of application programmers.

Data dependence

- A change in a data implies the need to make lots of changes in data and access programs.
- Various files may be in different formats which must be made known to applications programs.

Integrity problems

- Integrity constraints are part of program code.
- Hard to add new constraints or change existing one.
- Example: The number of students of cs387 must not exceed 40.

Atomicity of updates

- Failures may leave database in an inconsistent state with partial updates carried out.
- Example: Transfer of funds from one account to another should either complete or not happen at all.

Concurrent anomalies

- Concurrent access needed for performance.
- Uncontrolled concurrent accesses can lead to inconsistencies.
- Example: Two people reading a balance and updating it at the same time.

Security

- Many transactions processing contain data about the private concerns of individuals. So security is important.
- Examples: credit card, health and financial records, purchase history.
- Users must be authenticated (Are they the one they claim to be?)
- Users must be allowed to execute only the transaction they are allowed to execute.
Examples: Only a bank teller can execute a transaction to generate a certified check. Bank tellers should not have access to all employees data.
- The data in the DB must not be corrupted or read by an attacker. The data transmitted between the user and the system must not be altered or overheard by an eavesdropper.

SQL

- Structured Query Language
- 4GL (fourth generation language)
- Most widely used language in DBMS.
- Language of description of data (DDL).
- Language of manipulation of data (DML).
- Declarative language (Statements specify goals not how it is to be achieved).
- Simplifies application programs.
- However programmers should have an idea of strategies used in the DBMS because of performance.

Levels of abstraction

- Data actually stored as bits, but it is difficult to work at this level.
- It is convenient to view data at different levels of abstraction.
- Circa, 1975
- A **schema** is the description of data at some level - structure. Each level has its own schema.
- We will be concerned with 3 levels:
 - Physical/Implementation
 - Conceptual/Logical
 - External/View
- Similarities with programming languages.

Physical level

- Describe details of how data are stored: tracks, cylinders, indices...
- Physical schema.
- Early applications worked at this level - explicitly worked with details.
- Coding problems.
 - Change of data structure difficult to make
 - Application code becomes complex since it must deal with details
 - Rapid implementation of new features impossible
- Example: What is a customer?
Bits, bytes or words.

Logical/Conceptual level

- Hide details, closer to the way human sees data.
- Conceptual schema presents data and relationships between data.
- Use of Relational or Entity-Relationship model.
- DBMS maps from conceptual to physical level automatically.
- Physical schema can be changed without changing application.
 - DBMS must change mapping.
- Referred to as **physical data independence**.
- Example: What is a customer?

Customer(*ID* : *String*, *Name* : *String*, *Street* : *String*, *City* : *String*).

A customer is described by 4 fields.

account(*Ida* : *number*, *ID* : *String*)

An account is described by 2 fields.

External/View level

- An external schema provides a view of the conceptual level tailored to the needs of particular users. View of the data as seen by individual user.

Example: Payroll, Accounting, Advising.

- Portions of stored data should not be seen by some users (security mechanisms).

Examples:

- Students do not see data about other students.
- Faculty do not see billing data of the university.

- Information that can be derived from stored data might be viewed as if it were stored.

Example: GPA is not stored but computed when needed.

- Example: The representation of *Customer* is used. Details of the data type are hidden. Suppress the *Street* field.

External/View level

- View are computed when accessed not stored.
- Different external schemes can be provided to different groups of users.
- DBMS maps from external to conceptual level automatically.
- Conceptual schema can be changed without changing application.
 - Mapping from conceptual to external level must be changed.
- Referred to as **conceptual data independence**.

Summary

Reality



View/External level



Conceptual/Logical level



Physical level



Disk

Instance of a schema

- The contents of the database.
- Example:

For *Customer*(*Idc* : *String*, *Name* : *String*, *Street* : *String*, *City* : *String*), we instantiate *Idc* with "0", *Name* with "Scharff", *Street* with "Maple" and *City* with "Stony Brook".

Components of a DBMS

- Data Definition Language (DDL)
 - Specification notation for defining the database schema.
Analogy: definition of a type in a programming language.
 - DDL generate a set of tables stored in a **data dictionary** (or directory or catalog).
 - Example: SQL
- Data Manipulation Language (DML)
 - It supports loading data in the DB, updating it and generating reports.
 - Also known as query language.
 - Two classes of languages:
 - procedural: user specifies what data is required and how to get those data.
 - non procedural: user specifies what data is required without specifying how to get those data
 - Example: SQL - non procedural language

- Data dictionary

A small DB in its own right. It contains metadata and is consulted before actual data is accessed.

It contains: database schema, consistency constraints, access privileges of users...

- Host language Interface

Allows high level language to access DB.

- Graphics generator (optional)

What is a data model?

- Schema: description of data at some level.
- Model: tools and language for describing:
 - Conceptual and external schema
 - DDL
 - Integrity constraints
 - DDL
 - Operation on data
 - DML
 - Directives that influence the physical schema (affects performance, not semantics)
 - SDL (Storage Description Language)

The storage manager is a program module that provide the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

It is responsible of interaction with the file manager and efficient storing, retrieving and updating data.
- Examples: Relational model, Entity-Relationship model, object-oriented model, network model, hierarchical model.

What is database design?

- Goal: specification of database schema.
- Why database design is important?

A good database does not just happen. The structure of its contents must be designed carefully. Even a good DBMS will perform badly on a badly designed database.

A well-designed DB facilitates data management and becomes a valuable information generator.

A poorly designed DB tends to generate errors that are likely to lead to bad decision. Because the DB is the source from which information is generated, its design is subject of detailed study.

Database design is crucial activity. It is made simpler when you use models. Models are simplified abstractions of real-world events or conditions.

Roles in database

- Design, implementation, maintenance of transaction processing systems.
- System analyst - specifies system using input from customer; provides complete description of functionality from customer's and user's point of view.
- Database designer - specifies structure of data that will be stored in database.
- Application programmer - implements application programs that access data and support enterprise rules.
- Database administrator - maintains database once system is operational: space allocation, performance optimization, database security.
- System administrator - maintains transaction processing system: monitors interconnection of modules, deals with failure and congestion.

Vocabulary to know