Week 1

Need for Data Management

- Storage
- Retrieval
- Transaction
- Audit
- Archival

Physical Data Management

- aka **book-keeping**
- data management using physical ledgers or journals
- Henry Brown(American Inventor) patented a “receptacle for storing and preserving papers” on November 2, 1886.
- Herman Hollerith adapted the punch cards used for weaving looms to act as the memory
  for a mechanical tabulating machine, in 1890.

Electronic Data Management

History

- 1950s: Computer Programming started
- 1960s: Data Management with punch card/tapes and magnetic tapes
- 1970s:
  ◦ COBOL and CODASYL approach was introduced in 1971
  ◦ On October 14 in 1979, Apple II platform shipped VisiCalc, marking the birth of the
  spreadsheet
  ◦ Magnetic disks became prevalent
- 1980s: RDBMS changed the face of data management
- 1990s: With the Internet, data management started becoming global

- 2000s: e-Commerce boomed, NoSQL was introduced for unstructured data management
- 2010s: Data Science started riding high

Parameters

- Durability
- Scalability
- Security
- Retrieval
- Ease of Use
- Consistency
- Efficiency
- Cost

Why file systems are inefficient?

- time-consuming in scaled-up operations
- an upper limit on the number of rows
- consistency of data not maintained
- no means to check violations of constraints in the face of concurrent processing
- unable to give different permissions to different people in a centralised manner
- system crashes

Database Management System

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data.

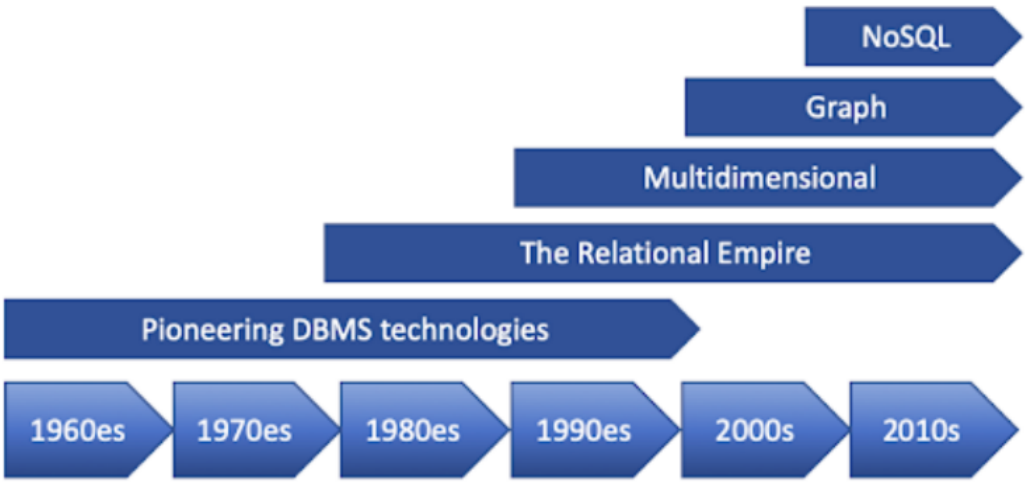
Management of data involves both defining structures for the storage of information and providing mechanisms for the manipulation of information.

History of DBMS

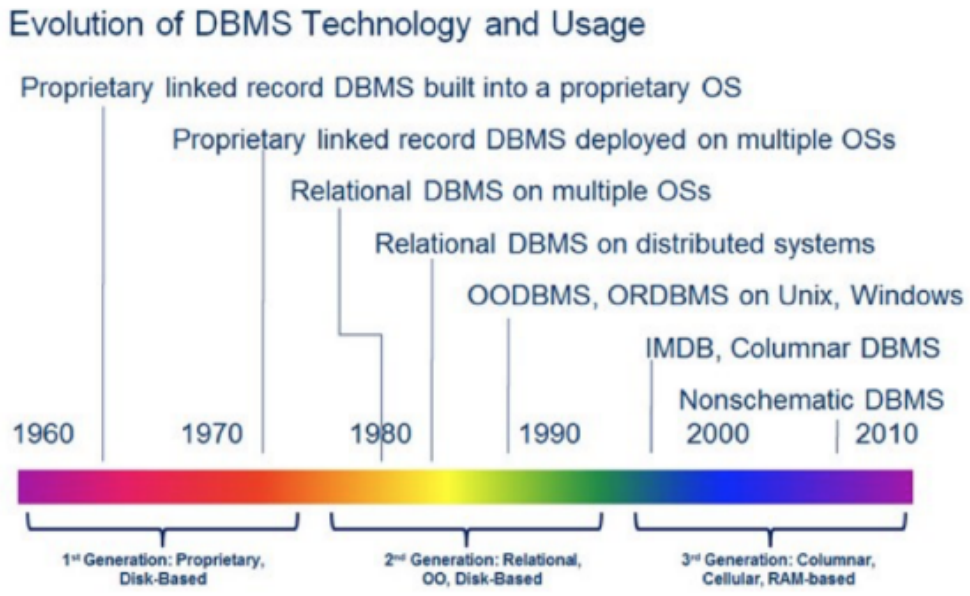
Timeline

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage which allowed only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models
  - Ted Codd defined relational data model
    - Won ACM Turing Award
    - IBM Research begins System R Prototype
    - UC Berkley begins Ingres prototype
  - High-Performance transaction processing
- 1980s:
  - SQL became industry standard
  - parallel and distributed database systems
  - object-oriented database systems
- 1990s:
  - Large decision support and data mining applications
  - Large multi-terabyte data warehouses
  - The emergence of Web Commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Late 2000s:
  - Giant data storage systems: Google BigTable Yahoo PNuts, etc.

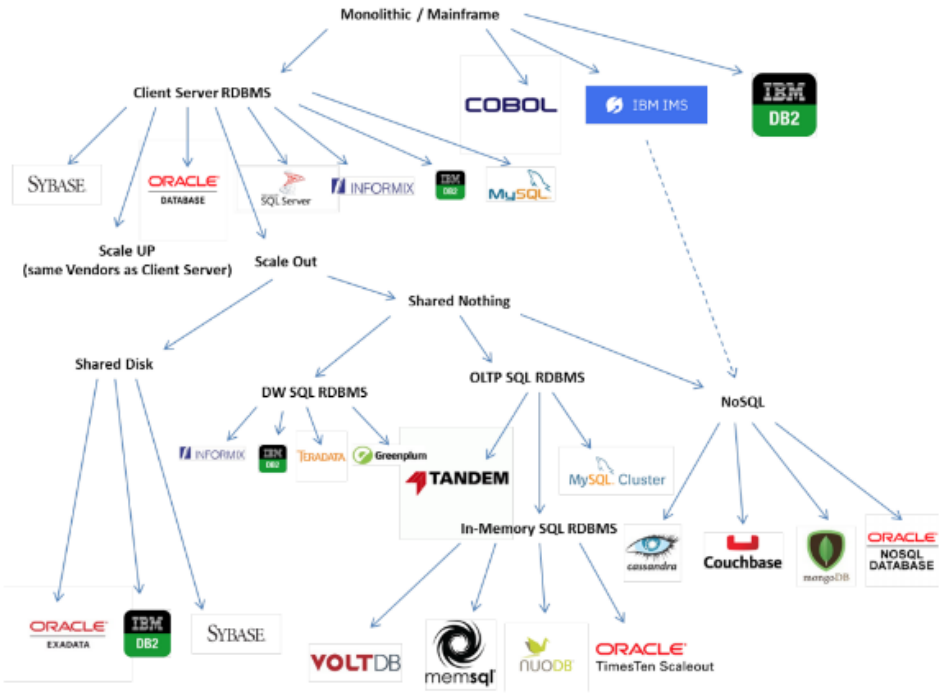
Evolution of Data Models



Evolution of DB Technology



Evolution of DB Architecture



File System vs DBMS

Parameter	File Handling via Python	DBMS
Scalability with respect to amount of data	Very difficult to handle insert, update and querying of records	In-built features to provide high scalability for a large number of records
Scalability with respect to changes in structure	Extremely difficult to change the structure of records as in the case of adding or removing attributes	Adding or removing attributes can be done seamlessly using simple SQL queries
Time of execution	In seconds	In milliseconds
Persistence	Data processed using temporary data structures have to be manually updated to the file	Data persistence is ensured via automatic, system induced mechanisms
Robustness	Ensuring robustness of data has to be done manually	Backup, recovery and restore need minimum manual intervention
Security	Difficult to implement in Python (Security at OS level)	User-specific access at database level
Programmer's productivity	Most file access operations involve extensive coding to ensure persistence, robustness and security of data	Standard and simple built-in queries reduce the effort involved in coding thereby increasing a programmer's throughput
Arithmetic operations	Easy to do arithmetic computations	Limited set of arithmetic operations are available
Costs	Low costs for hardware, software and human resources	High costs for hardware, software and human resources

Parameterised Comparison

Scalability

PPD

File handling via Python

- **Number of records:** As the # of records increases, the efficiency of flat files reduces:
  - the time spent in searching for the right records
  - the limitations of the OS in handling huge files
- **Structural Change:** To add an attribute, initializing the new attribute of each record with a default value has to be done by program. It is very difficult to detect and maintain relationships between entities if and when an attribute has to be removed.

DBMS

- **Number of records:** Databases are built to efficiently scale up when the # of records increase drastically.
  - In-built mechanisms, like indexing, for quick access of right data.
- **Structural Change:** During adding an attribute, a default value can be defined that holds for all existing records - the new attribute gets initialized with the default value. During deletion, constraints are used either not to allow the removal or ensure its safe removal

Time and Efficiency

PPD

File handling via Python

- The effort needed to implement a file handler is quite less in Python
- In order to process a 1GB file, a program in Python would typically take few seconds.

DBMS

- The effort to install and configure a DB in a DB server is expensive & time consuming
- In order to process a 1GB file, an SQL query would typically take few milliseconds.

- If the number of records is very small, the overhead in installing and configuring a database will be much more than the time advantage obtained from executing the queries.
- However, if the number of records is really large, then the time required in the initialization process of a database will be negligible as compared to the time saved in using SQL queries.

Persistence, Robustness, Security

PPD

File handling via Python

- **Persistence**: Data processed using in-memory data structures stay in the memory during processing. After updates, these are manually updated to the file on disk
- **Robustness**: Ensuring consistency, reliability and sanity is manual via multiple checks. On a system crash, a transaction may cause inconsistency or loss of data.
- **Security**: Extremely difficult to implement granular security in file systems. Authentication is at the OS level.

DBMS

- **Persistence**: Data persistence is ensured via automatic, system mechanisms. The programmer does not have to worry about the data getting lost due to manual errors
- **Robustness**: Backup, recovery & restore need minimum manual intervention. The backup and recovery plan can be devised for automatic recovery on a crash
- **Security**: DBMS provides user-specific access at the database level with restriction for to view only access

Programmer's Productivity

PPD

File handling via Python

- **Building the file handler**: Since the constraints within and across entities have to be enforced manually, the effort involved in building a file handling application is huge
- **Maintenance**: To maintain the consistency of data, one must regularly check for sanity of data and the relationships between entities during inserts, updates and deletes
- **Handling huge data**: As the data grows beyond the capacity of the file handler, more efforts are needed

DBMS

- **Configuring the database**: The installation and configuration of a database is specialized job of a DBA. A programmer, on the other hand, is saved the trouble
- **Maintenance**: DBMS has in-built mechanisms to ensure consistency and sanity of data being inserted, updated or deleted. The programmer does not need to do such checks
- **Handling huge data**: DBMS can handle even terabytes of data - Programmer does not have to worry

Arithmetic Operations

PPD

File handling via Python

- **Extensive support for arithmetic and logical operations**: Extensive arithmetic and logical operations can be performed on data using Python. These include complex numerical calculations and recursive computations.

DBMS

- **Limited support for arithmetic and logical operations**: SQL provides limited arithmetic and logical operations. Any other complex computation has to be done outside the SQL.

Costs and Complexity

PPD

File handling via Python

- File systems are cheaper to install and use. No specialized hardware, software or personnel are required to maintain filesystems.

DBMS

- Large databases are served by dedicated database servers need large storage and processing power
- DBMSs are expensive software that have to be installed and regularly updated
- Databases are inherently complex and need specialized people to work on it - like DBA
- The above factors lead to huge costs in implementing and maintaining database management systems

Levels of Abstraction

Physical Level

describes how a record is stored

Logical Level

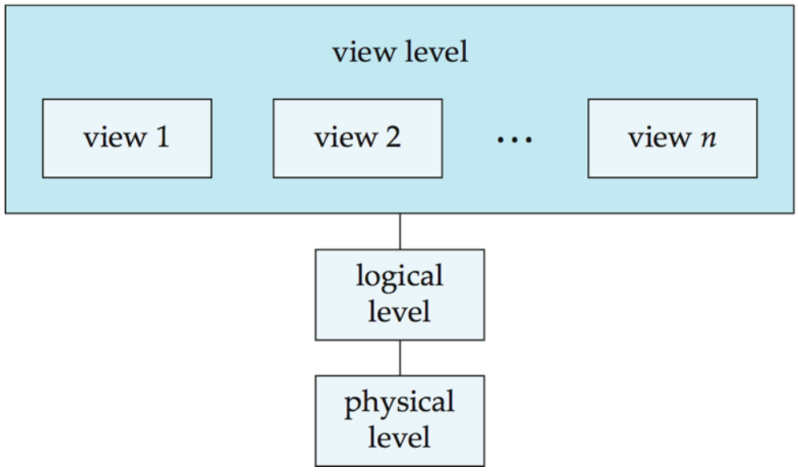
describes data stored in the database along with their relationships

View Level

- hide details of datatypes
- for application programmers
- can show info in derived form

View of Data

An architecture for a database system



Schema and Instances

Logical Schema

the overall logical structure of the database

Physical Schema

the overall physical structure of the database

Instance

data stored in the database at a given point in time

Physical and Logical Data Independence

Basically, Data Independence allows you to change the Database schema at one level of a database schema without changing the schema at the next higher level. Level of abstraction is Physical level —> Logical level —> View level.

So, in Logical Data Independence, if you modify the schema at Logical level, it will not affect the view level.

And, in Physical Data Independence, a change in Physical level should not affect either the View level or the Logical level.

Data Definition Language (DDL)

- for defining database schema
- DDL compiler creates a set of table templates stored in a **data dictionary**
- Data dictionary contains metadata (schema, integrity constraints, authorization)

Data Manipulation Language (DML)

- for accessing and manipulating database schema
- aka query language
- Two classes of language:
  - **Pure:** Relational Algebra, Tuple Relational Calculus, Domain relational calculus
  - **Commercial:** SQL

Structured Query Language (SQL)

- not a Turing Machine Equivalent Language
- To be able to compute complex functions, SQL is usually embedded in some
  higher-level language

Database Design

Logical Design

deciding database schema, planning attributes

Physical Design

physical layout of database

Data Model

Underlying the structure of a database is the data model: a collection of conceptual
tools for describing data, data relationships, data semantics, and consistency constraints. A data model provides a way to describe the design of a database at
the physical, logical, and view levels.

Categories:

- Relational
- E-R
- Object-based
- Semi-structured(XML)

Historically, the network data model and the hierarchical data model preceded the relational data model.

Object-Relational Data Model

- **allows attributes of tuples to have complex data types** in contrast to the relational data model which allows only atomic or flat values
- preserves relational foundations
- Provide upward compatibility with existing relational languages

XML: Extensible Markup Language

- Defined by WWW Consortium (W3C)
- allows the creation of tags and nested tag structures
- basis for all new generation data interchange formats
- management of unstructured data

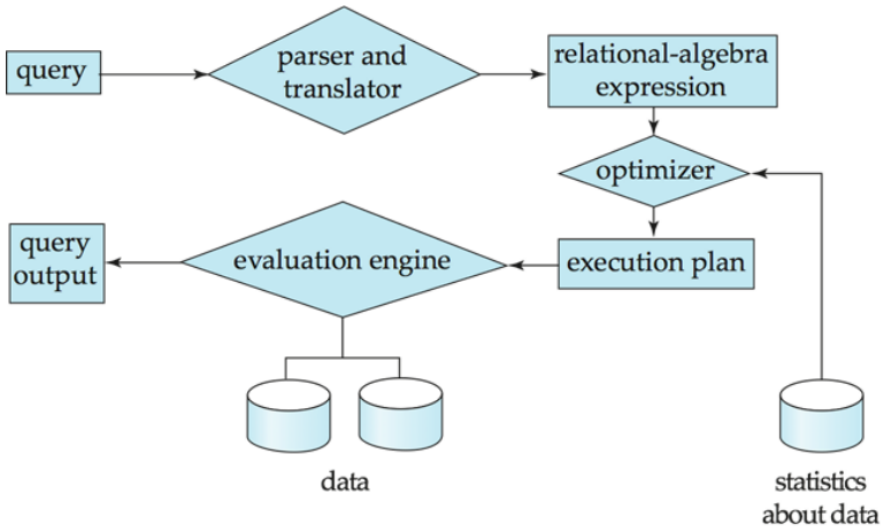
Database Engine

Storage Manager

- **interface between low-level data stored in the database and application programs & queries submitted to the system**
- interacts with OS file manager
- efficient storage, retrieval and update of data
- Issues:
  - storage access
  - file organisation
  - indexing and hashing

Query Processing

- parsing and translation
- optimization
- evaluation
- alternative ways of evaluating a given query
- cost estimate of operations

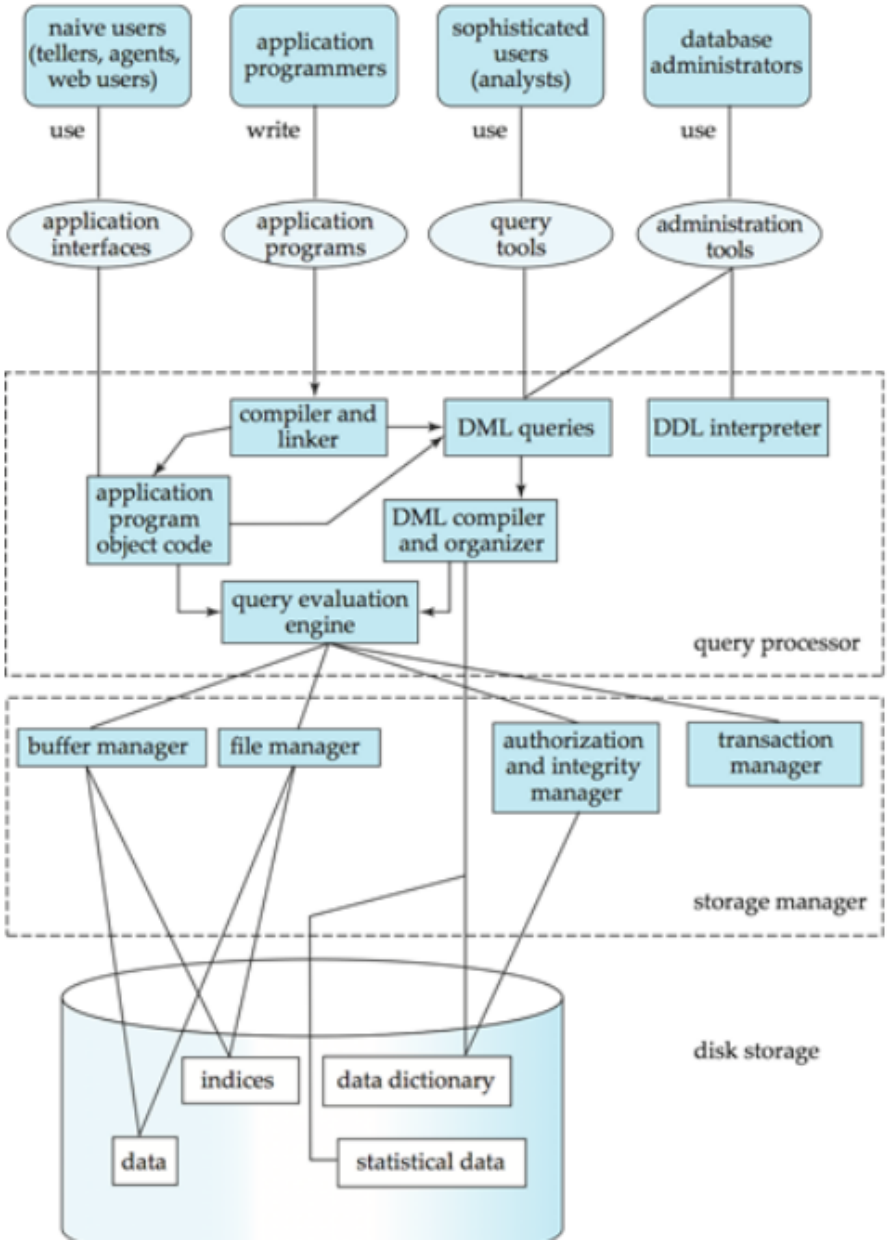


Transaction Manager

- A transaction is a **collection of operations that performs a single logical function in a database application**
- transaction management ensures that the database remains consistent

- Concurrency control

Database System Internals



Database Architecture

- **Centralised**
- **Client-sever** (Sybase, IBM DB2, PostgresSQL, SQL Server, Oracle, Informix)

- **Parallel**
- **Distributed**
- **Cloud**