DBMS

Dr. Debasish Jana
BE(CS)(JU), MMATH(CS)(UW, Canada), MBA(Fin)(IGNOU), PhD(CS, JU), FIE(I), FIETE, SMIEEE, SMACM
debasishj871 at gmail dot com

2024

Introduction to DBMS: Concept & Overview of DBMS

The Need for Databases

Data Models

Relational Databases

Database Design

Storage Manager

Query Processing

Transaction Manager

Relational Model

1. Fundamentals of Database Systems by E. Navathe
2. Database System Concepts by Korth and Silberschatz
3. Commercial Application Development Using Oracle Developer – 2000 by Bayross
4. Data Mining: Concepts and Techniques by J. Han & M. Kamber
5. Data Mining: Practical Machine Learning Tools and Techniques by Ian H. Witten and Eibe Frank

Data Models, Database Languages, Database Administrator

Database Users, Three Schema architecture of DBMS

Database Design based on ER/EER Diagram

Functional Dependency

Database Management System (DBMS)

- DBMS
  - helps manage and organize collection of data in a database
  - enables users to create, access, modify, and manage/maintain database
  - provides an efficient and secure way to store, retrieve, and manipulate data.
  - provides various features such as data integrity, security, and concurrency control
  - contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Data models
  - logical representations of data that describe how data is organized, stored, and manipulated within a database
  - various types of data models, including hierarchical, network, relational, and object-oriented models
  - The most widely used data model is the relational data model, which uses tables to represent data and defines relationships between them.

Database Management System (DBMS)…contd

- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Biological and Genome Databases
  - Data Warehouses
  - Mobile databases
  - Real-time and Active Databases
- Databases can be very large.
- Databases touch all aspects of our lives
- The relational model uses a set of rules called relational algebra to manipulate data.

Recent Developments (1)

- Social Networks started capturing a lot of information about people and about communications among people-posts, tweets, photos, videos in systems such as:

- Facebook

- Twitter

- Linked-In

- All of the above constitutes data
- Search Engines- Google, Bing, Yahoo : collect their own repository of web pages for searching purposes

Recent Developments (2)

- New Technologies are emerging from the so-called non-database software vendors to manage vast amounts of data generated on the web:
- Big Data storage systems involving large clusters of distributed computers
- NOSQL (Not Only SQL) systems
- A large amount of data now resides on the “cloud” which means it is in huge data centers using thousands of machines.

University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - Multiple files and formats
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data (Cont.)

- 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 access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Main Characteristics of the Database Approach

- **Self-describing nature of a database system:**
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
  - The description is called **meta-data***.
  - This allows the DBMS software to work with different database applications.
- **Insulation between programs and data:**
  - Called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

----

* Some newer systems such as a few NOSQL systems need no meta-data: they store the data definition within its structure making it self describing

Levels of Abstraction

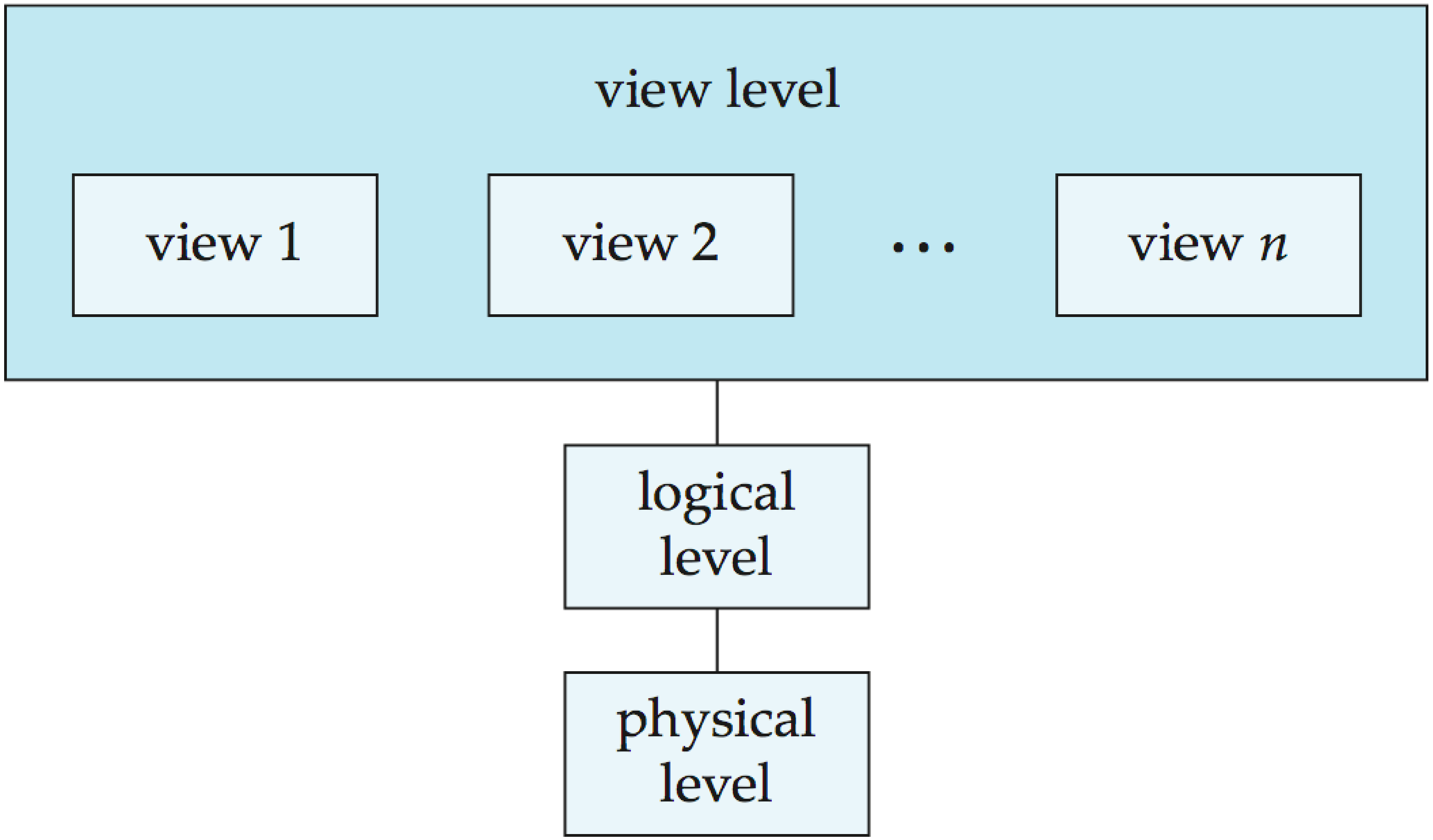
- **Physical level:** describes how a record (e.g., instructor) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

```
type instructor = record
        ID : string;
        name : string;
        dept_name : string;
        salary : integer;
    end;
```

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

- Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- **Physical schema**– the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Three-schema architecture of a DBMS

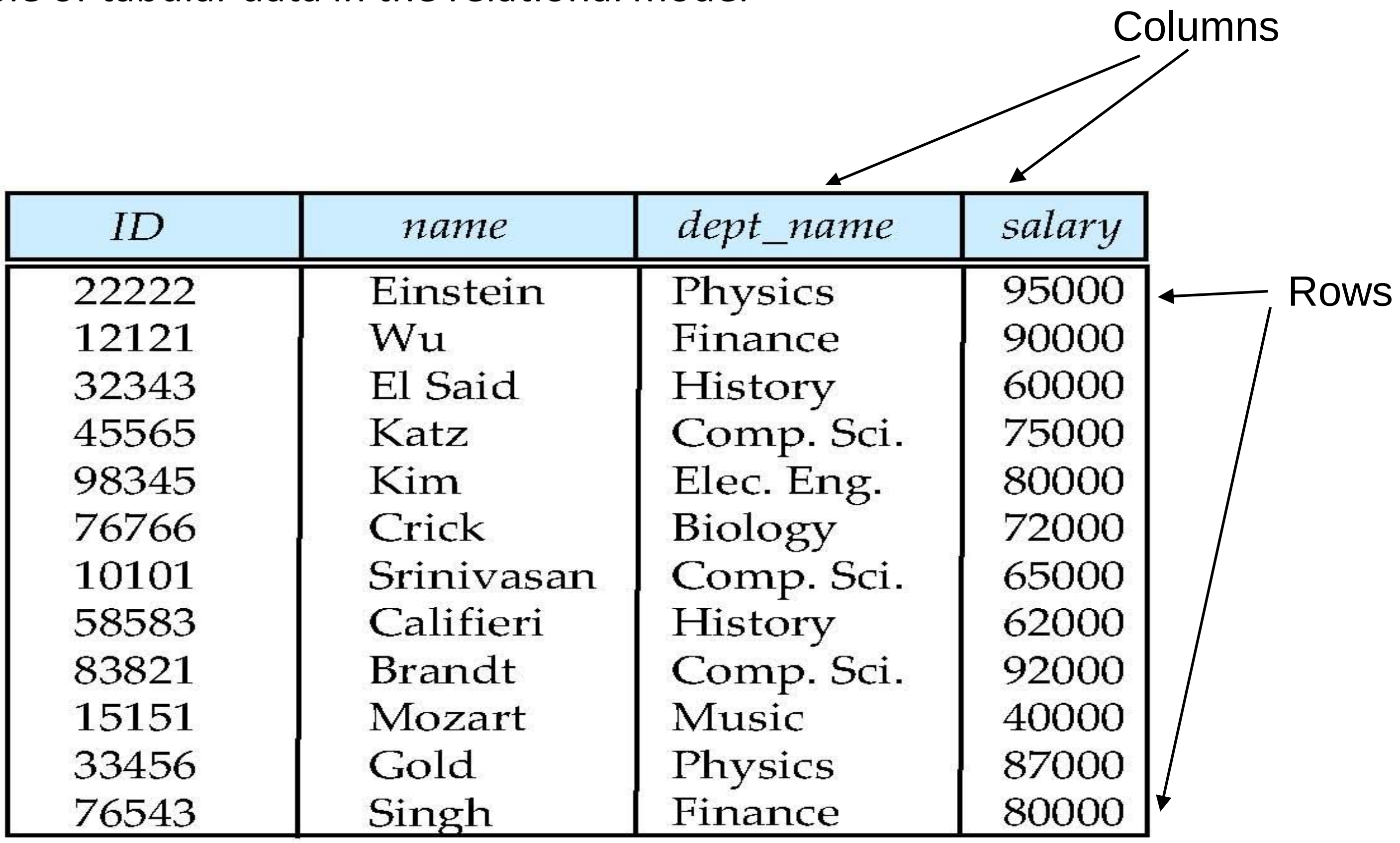
- The external schema represents how users view the data and defines the user's interface to the database.
- The conceptual schema describes the overall logical structure of the database and defines the relationships between the various data elements.
- The internal schema represents the physical storage structure of the database.
- This separation of concerns enables changes to be made to one schema without affecting the others, making it easier to maintain and modify the database over time.

Data Models

- **Data Abstraction:**
  - A **data model** is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details
- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Relational Model

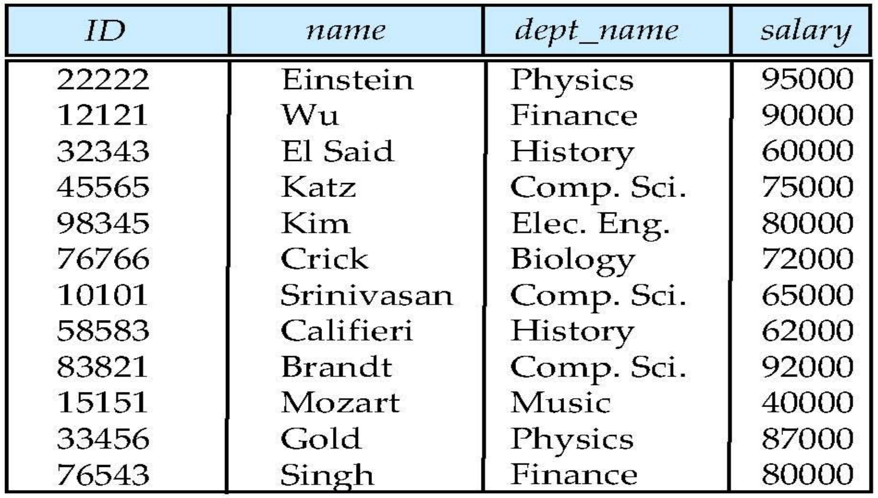
- All the data is stored in various tables.
- Example of tabular data in the relational model


Columns
Rows

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

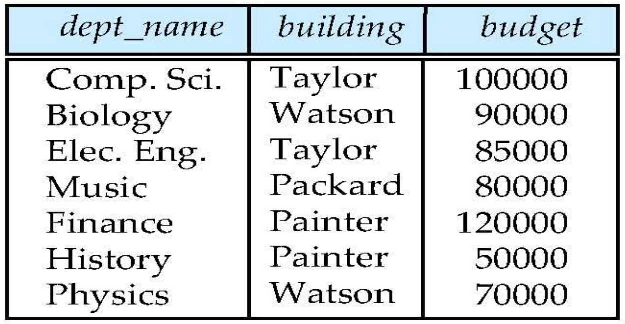
(a) The *instructor* table

A Sample Relational Database



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table



dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Data Definition Language (DDL)

- Database languages: used to interact with a database
  - Data Definition Language (DDL) and Data Manipulation Language (DML).
  - DDL is used to create, modify, and delete the structure of a database
  - DML is used to manipulate the data stored within a database.
- Specification notation for defining the database schema

  Example: **create table** *instructor* (
  ID **char**(5),
  name **varchar**(20)**,**
  dept_name **varchar**(20),
  salary **numeric**(8,2))
- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language

SQL

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent language
- SQL is NOT a Turing machine equivalent language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

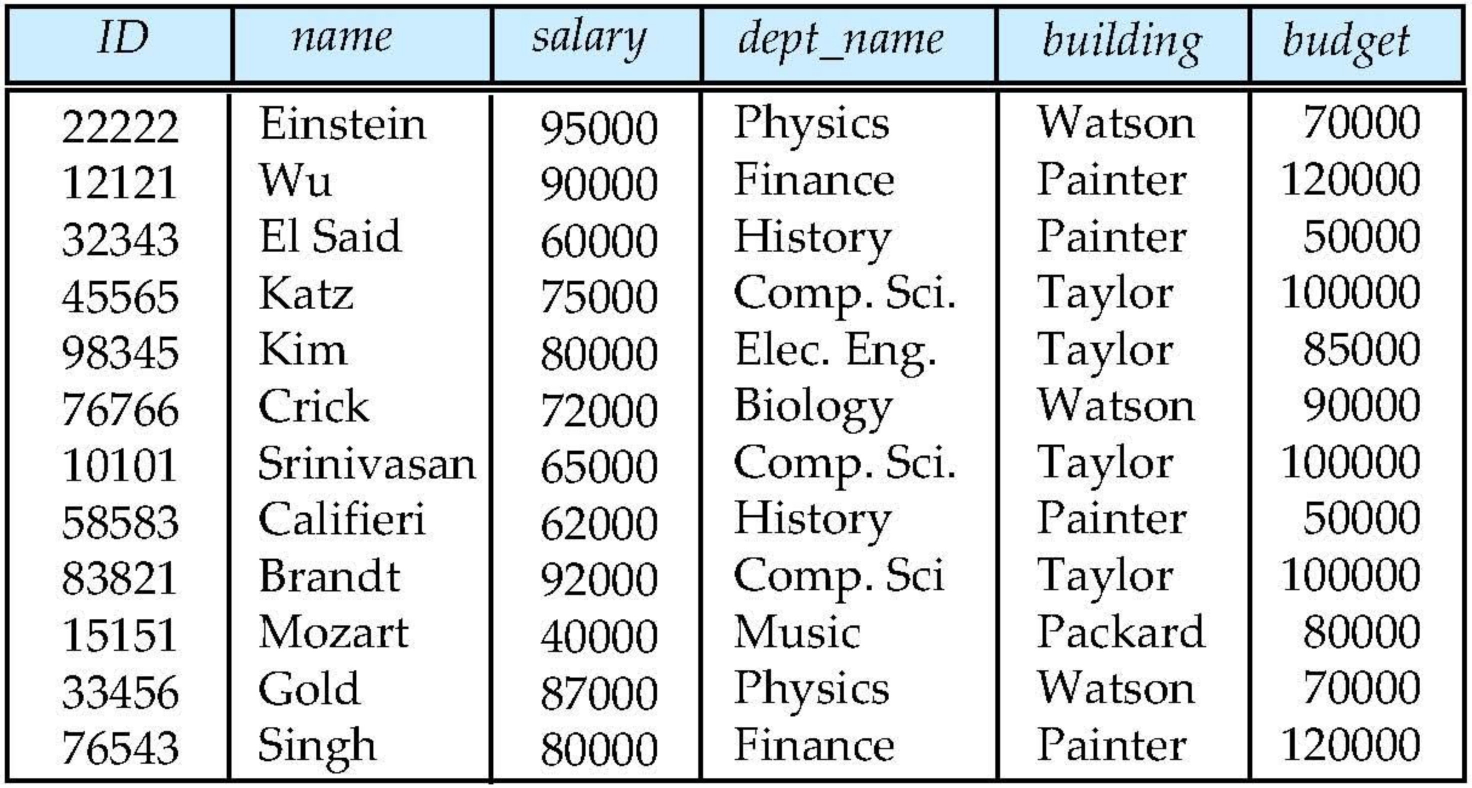
Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Design (Cont.)

- Is there any problem with this relation?



ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Design Approaches

- Need to come up with a methodology to ensure that each of the relations in the database is "good"
- Two ways of doing so:
  - Entity Relationship Model
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory
    - Formalize what designs are bad, and test for them

Object-Relational Data Models

- Relational model: flat, “atomic” values
- Object Relational Data Models
    - Extend the relational data model by including object orientation and constructs to deal with added data types.
    - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
    - Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
    - Provide upward compatibility with existing relational languages.

XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data

Database Engine

- Storage manager
- Query processing
- Transaction manager

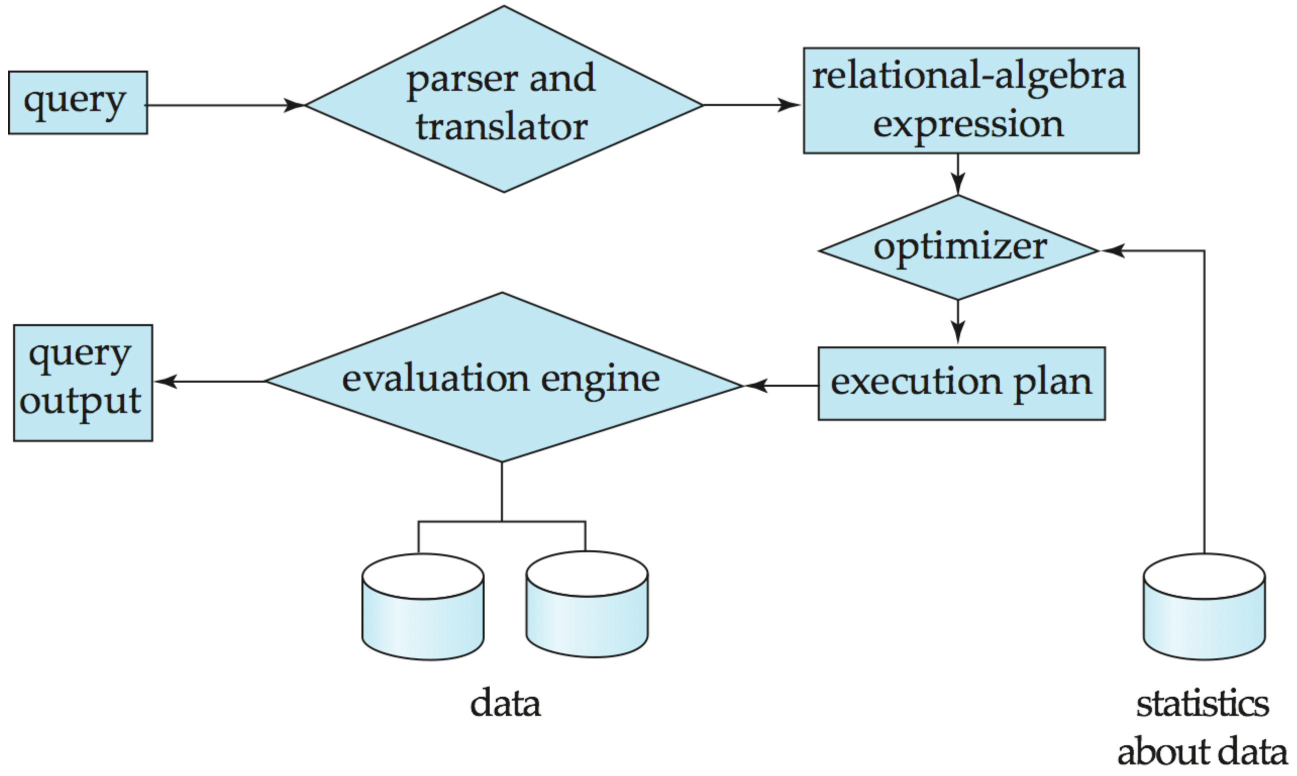
Storage Management

- **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.
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

Query Processing

Query access different parts of data and formulate the result of a request

1. Parsing and translation
2. Optimization
3. Evaluation



Query Processing (Cont.)

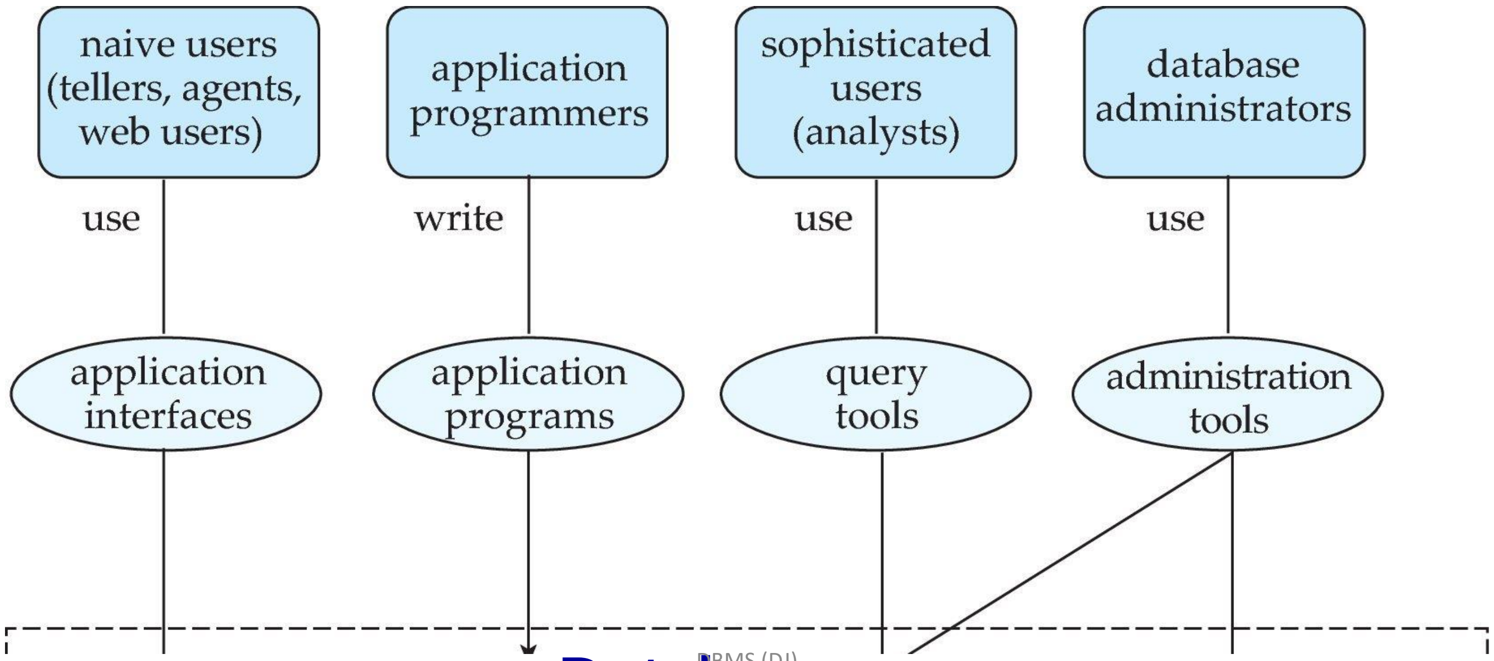
- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions

Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- A **transaction** is a collection of operations that performs a single logical function in a database application, e.g. that may read some data and "update" certain values or generate new data and store that in the database
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Database Users and Administrators

A database administrator (DBA) is responsible for managing and maintaining a database. Their duties include creating and managing user accounts, monitoring database performance, ensuring data security, and performing backups and recovery operations.
Database users are individuals or applications that interact with a database. They may perform various operations such as querying the database for information, inserting new data, updating existing data, and deleting data.



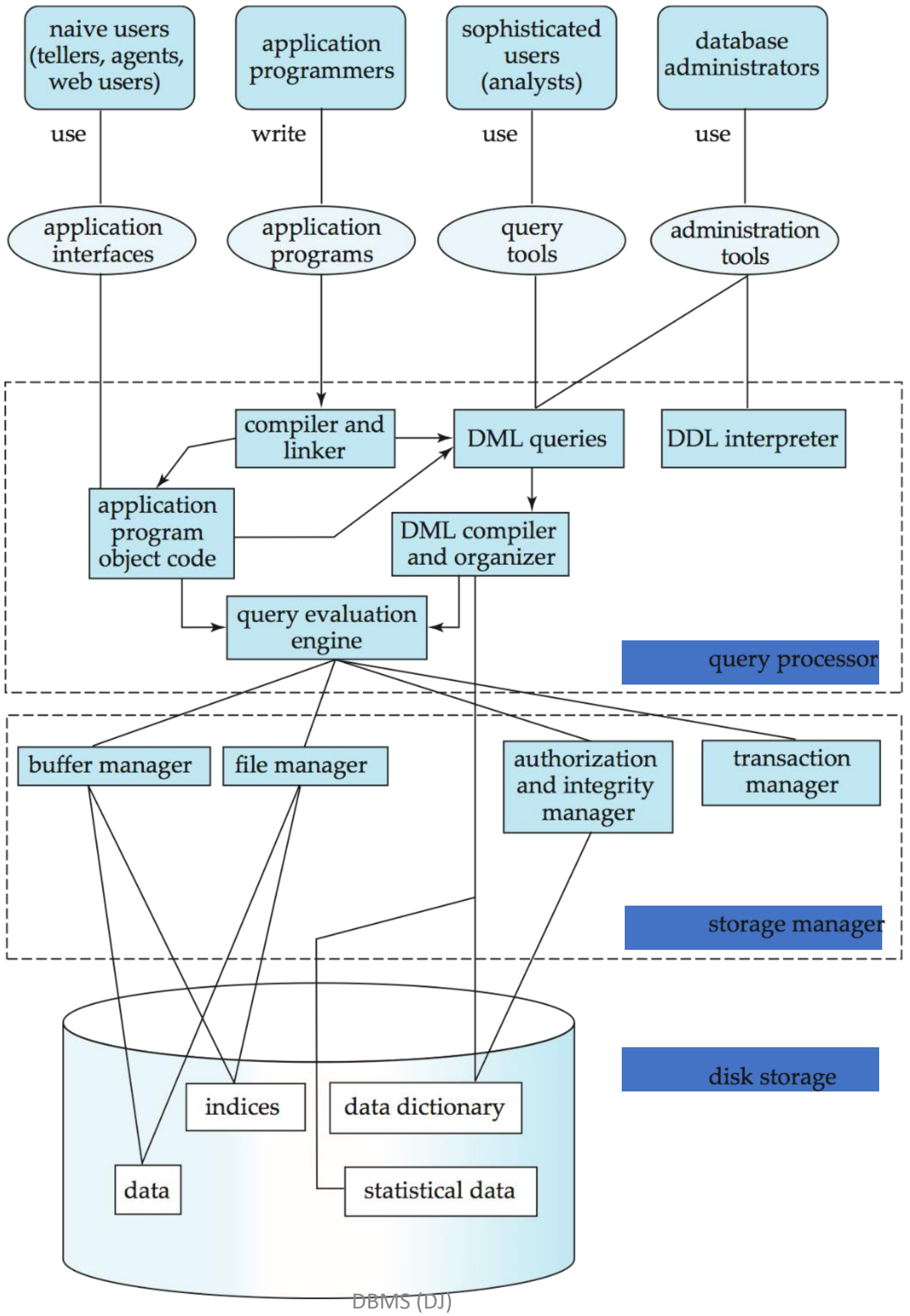
Database

Few Concepts

- Database Design based on ER/EER Diagram:
- Entity-relationship (ER) diagrams are a visual representation of the entities and relationships in a database. They are used to design and model databases before implementation. An enhanced entity-relationship (EER) diagram is an extension of the ER diagram that includes additional constructs such as subclasses, superclasses, and inheritance.

- Functional Dependency:
- Functional dependency is a concept in database design that describes the relationship between two attributes in a table. It is a constraint that specifies that the value of one attribute determines the value of another attribute. In other words, if you know the value of one attribute, you can determine the value of another attribute. Functional dependency is used to ensure data integrity and eliminate redundant data in a database.

Database System Internals



Database Architecture

The architecture of a database systems is greatly influenced by

the underlying computer system on which the database is running:

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models were introduced in mid 1960s and dominated during the seventies.
  - A bulk of the worldwide database processing still occurs using these models, particularly, the hierarchical model using IBM's IMS system
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley begins Ingres prototype
    - Relational DBMS Products emerged in the early 1980s.
  - High-performance (for the era) transaction processing

History (cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems, to cater to the need of complex data processing in CAD and other applications, although their use has not taken off much.
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..

Historical Development of Database Technology

- Data on the Web and E-commerce Applications:
  - Web contains data in HTML (Hypertext markup language) with links among pages.
  - This has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language).
  - Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database
  - Also allow database updates through Web pages

Extending Database Capabilities (1)

- New functionality is being added to DBMSs in the following areas:
  - Scientific Applications – Physics, Chemistry, Biology - Genetics
  - Earth and Atmospheric Sciences and Astronomy
  - XML (eXtensible Markup Language)
  - Image Storage and Management
  - Audio and Video Data Management
  - Data Warehousing and Data Mining – a very major area for future development using new technologies (see Chapters 28-29)
  - Spatial Data Management and Location Based Services
  - Time Series and Historical Data Management
- The above gives rise to *new research and development* in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.

Extending Database Capabilities (2)

- Background since the advent of the 21st Century:
  - First decade of the 21st century has seen tremendous growth in user generated data and automatically collected data from applications and search engines.
  - Social Media platforms such as Facebook and Twitter are generating millions of transactions a day and businesses are interested to tap into this data to “understand” the users
  - Cloud Storage and Backup is making unlimited amount of storage available to users and applications

Extending Database Capabilities (3)

- Emergence of Big Data Technologies and NOSQL databases
  - New data storage, management and analysis technology was necessary to deal with the onslaught of data in petabytes a day (10**15 bytes or 1000 terabytes) in some applications – this started being commonly called as “Big Data”.
  - Hadoop (which originated from Yahoo) and Mapreduce Programming approach to distributed data processing (which originated from Google) as well as the Google file system have given rise to Big Data technologies. Further enhancements are taking place in the form of Spark based technology.
  - NOSQL (Not Only SQL- where SQL is the de facto standard language for relational DBMSs) systems have been designed for rapid search and retrieval from documents, processing of huge graphs occurring on social networks, and other forms of unstructured data with flexible models of transaction processing.

When not to use a DBMS

- Main inhibitors (costs) of using a DBMS:
  - High initial investment and possible need for additional hardware.
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions.
- When a DBMS may be unnecessary:
  - If the database and applications are simple, well defined, and not expected to change.
  - If access to data by multiple users is not required.
- When a DBMS may be infeasible:
  - In embedded systems where a general purpose DBMS may not fit in available storage

When not to use a DBMS

- When no DBMS may suffice:
  - If there are stringent real-time requirements that may not be met because of DBMS overhead (e.g., telephone switching systems)
  - If the database system is not able to handle the complexity of data because of modeling limitations (e.g., in complex genome and protein databases)
  - If the database users need special operations not supported by the DBMS (e.g., GIS and location based services).