Chapter 1: Introduction

Database System Concepts, 6th Ed.

Database

- The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.
- Using the database, you can easily retrieve, insert, and delete the information.
- Databases can be very large.
- **For example:** The **college Database** organizes the data about the admin, staff, students and faculty etc.

Database Management System (DBMS)

- Database management system is **a software** which is

  used to manage the database.

  For example: MySQL, Oracle, etc are a very popular commercial database which is used in different applications.

- DBMS 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 us
- DBMS provides an interface to perform various operations like database **creation**, **storing** data in it, **updating** data, **creating a table** in the database etc.
- It provides **protection and security** to the database. In the case of multiple users, it also maintains data consistency.

Database Management System (DBMS)

- **DBMS allows users the following tasks:**
  - **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
  - **Data Updation:** It is used for the insertion, modification, and deletion of the actual data in the database.
  - **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.
  - **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

DBMS Applications

- 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
- In the early days, database applications were built directly on top of file systems

DBMS Applications University Database

- 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

DBMS Characteristics

- It uses a digital repository established on a server to store and manage the information.
- It can provide a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.
- It contains ACID properties which maintain data in a healthy state in case of failure.
- It can reduce the complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It can view the database from different viewpoints according to the requirements of the user.

Disadvantages of DBMS

- **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.
- **Size:** It occupies a large space of disks and large memory to run them efficiently.
- **Complexity:** Database system creates additional complexity and requirements.
- **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

Advantages of DBMS

- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.
- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

DBMS vs. File System

DBMS	File System
DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	File system is a collection of data. In this system, the user has to write the procedures for managing the database.
DBMS gives an abstract view of data that hides the details.	File system provides the detail of the data representation and storage of data.
DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure.	File system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will lost.
DBMS provides a good protection mechanism.	It is very difficult to protect a file under the file system.
DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	File system can't efficiently store and retrieve the data.
DBMS takes care of Concurrent access of data using some form of locking.	In the File system, concurrent access has many problems like redirecting the file while other deleting some information or updating some information.

Drawbacks of using file systems to store data

1. Data redundancy and inconsistency
    1. Multiple file formats, duplication of information in different files
2. Difficulty in accessing data
    1. Need to write a new program to carry out each new task
3. Data isolation
    1. Multiple files and formats
4. Integrity problems
    1. Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
    2. Hard to add new constraints or change existing ones
5. Security problems
    - Hard to provide user access to some, but not all, data

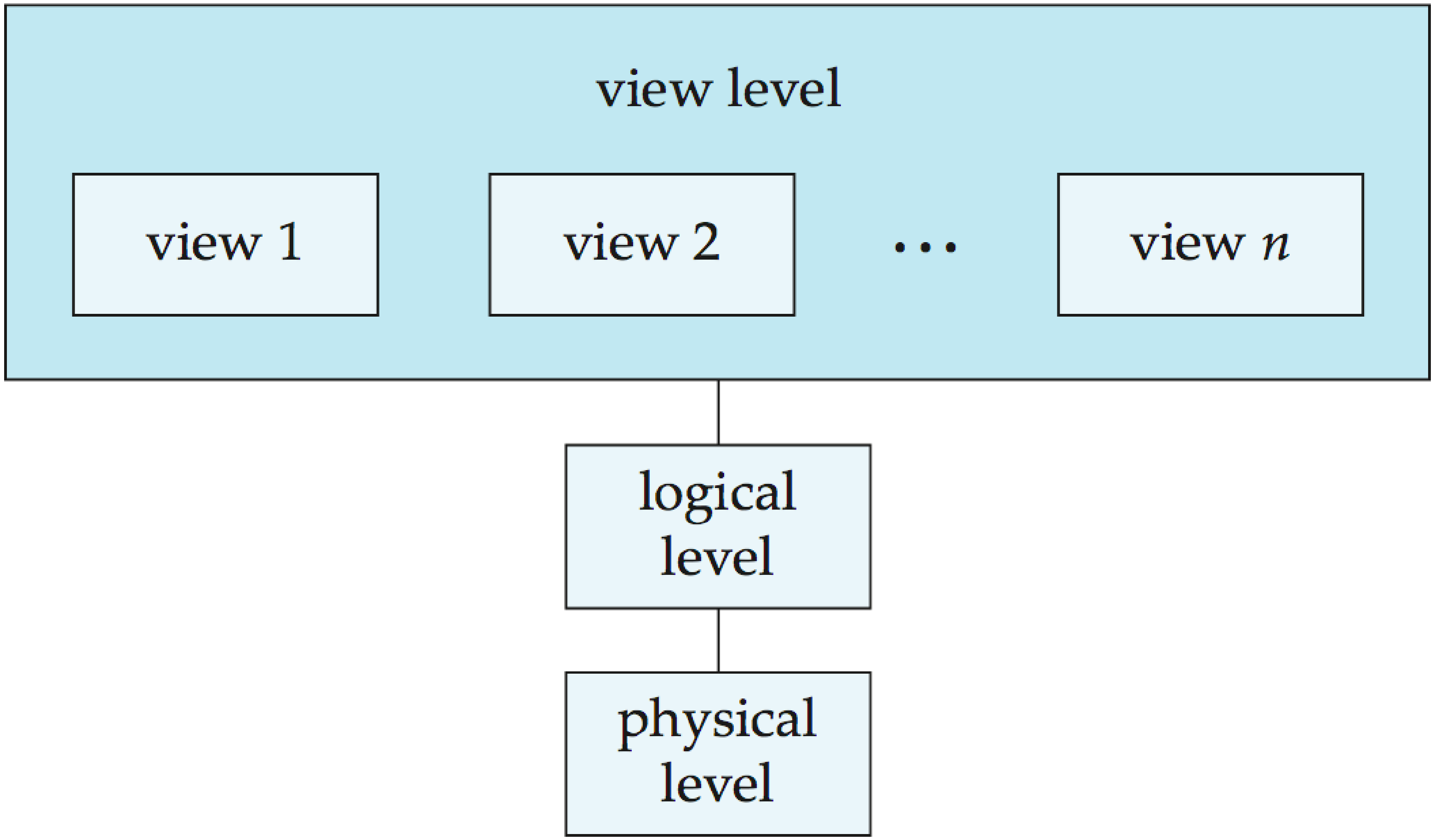
Drawbacks of using file systems to store data (Cont.)

6. Atomicity of updates
    1. Failures may leave database in an inconsistent state with partial updates carried out
    2. Example: Transfer of funds from one account to another should either complete or not happen at all
7. Concurrent access by multiple users
    1. Concurrent access needed for performance
    2. Uncontrolled concurrent accesses can lead to inconsistencies
        1. Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time

Database systems offer solutions to all the above problems

View of Data

An architecture for a database system



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.

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.

Data Models

- Is 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)

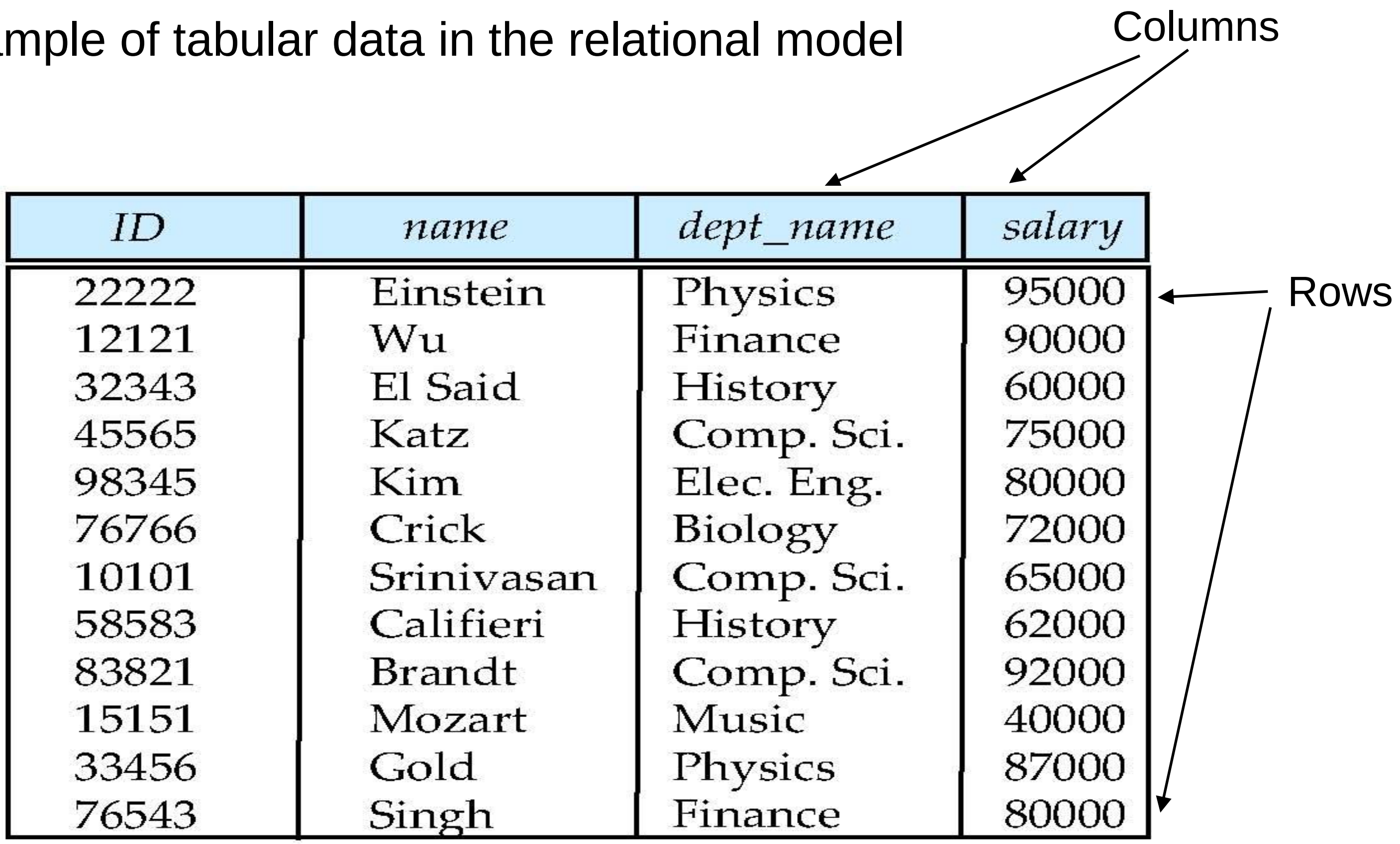
Semistructured data model (XML)

Network model

Hierarchical model

Relational Model

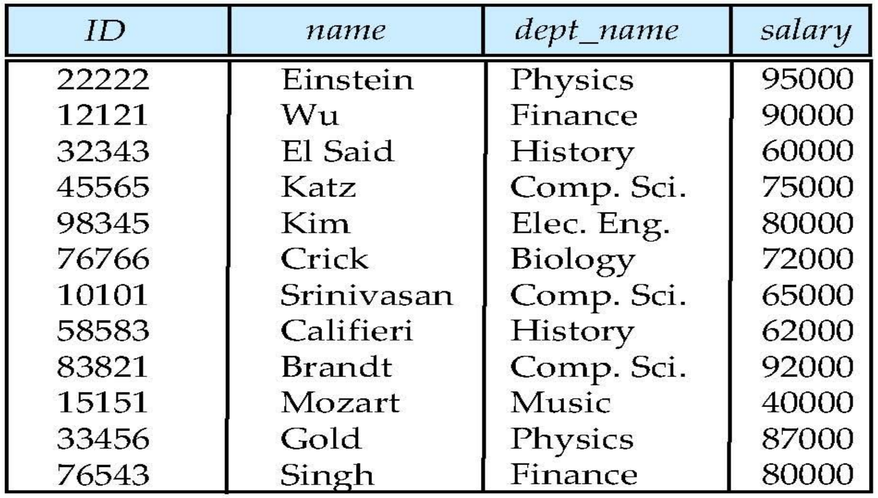
- All the data is stored in various tables.
- Example of tabular data in the relational model


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

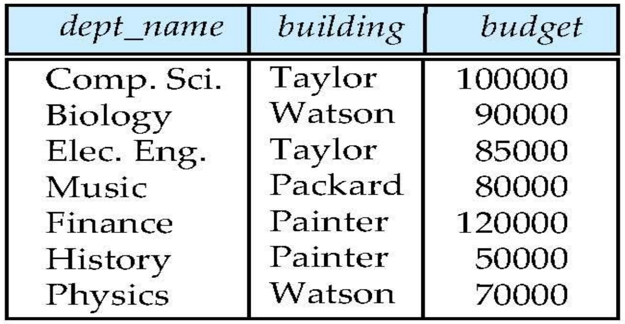
(a) The *instructor* table

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

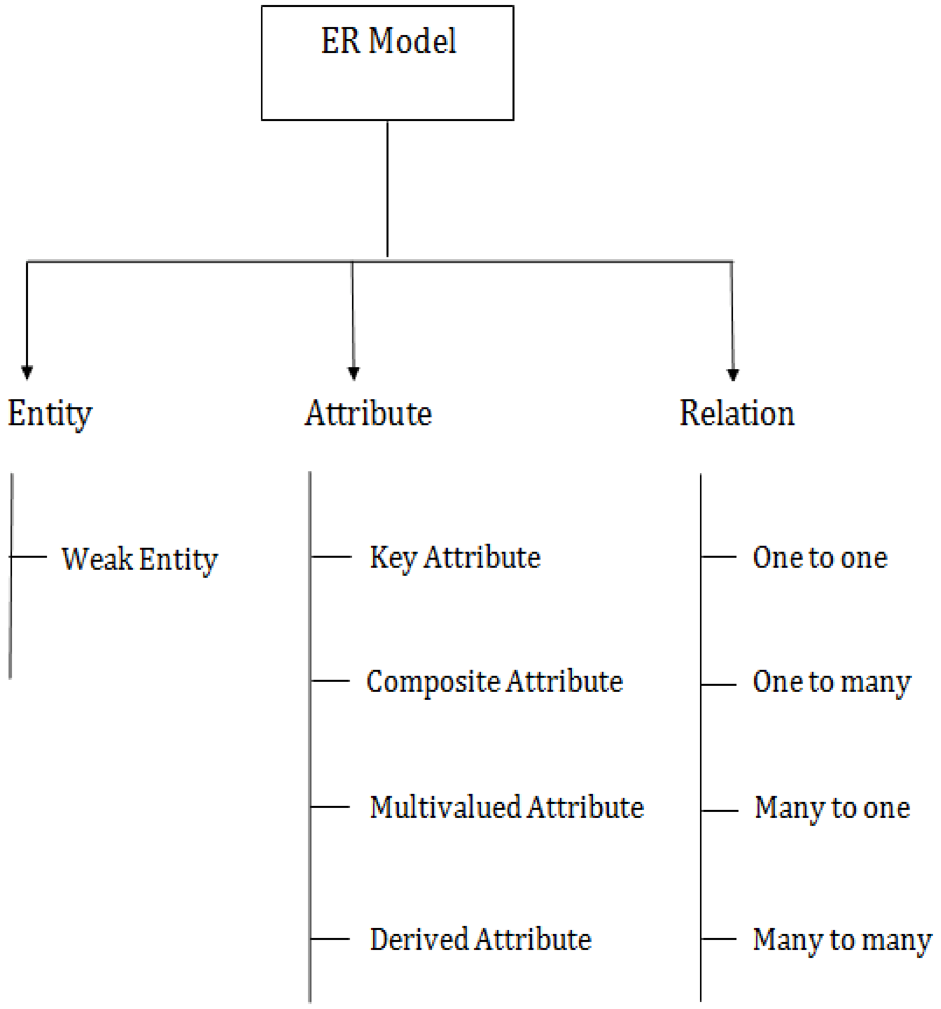
(a) The *instructor* table

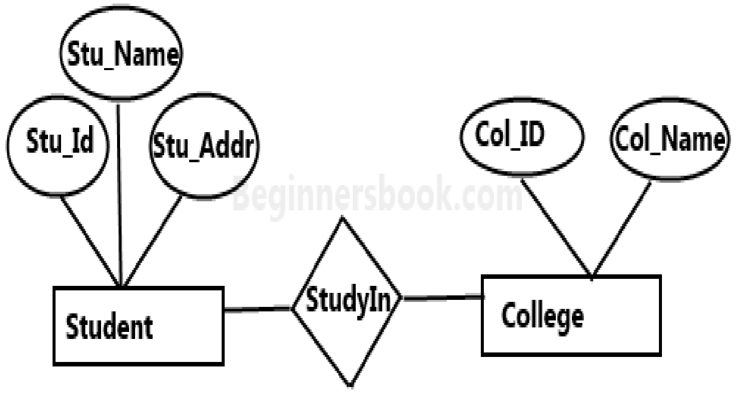


dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

ER Diagram



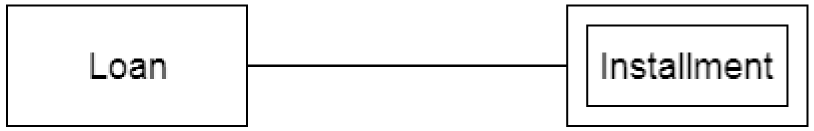


Sample E-R Diagram

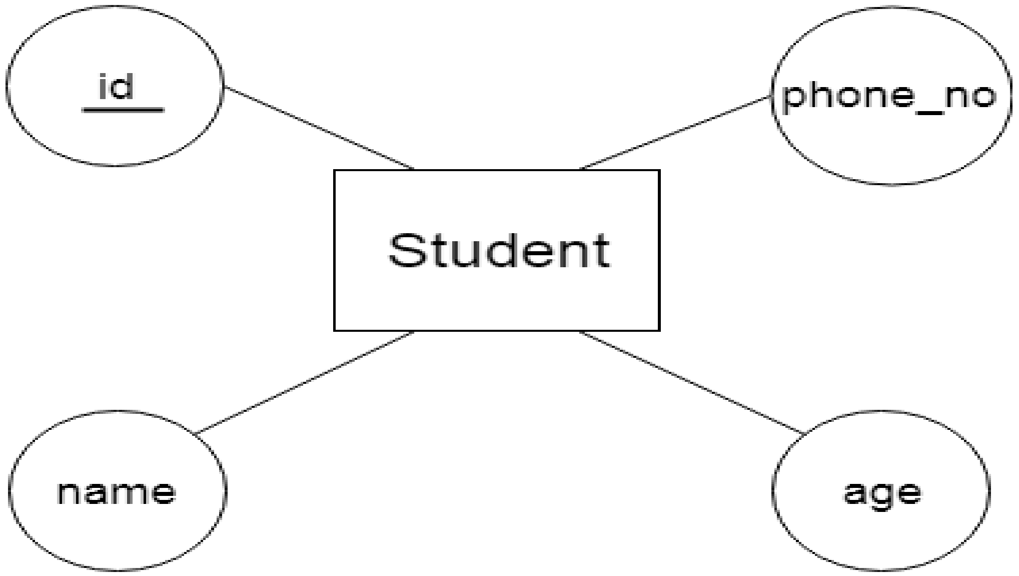
Source: https://www.javatpoint.com/dbms-mapping-constraints

ER Diagram

a. Weak Entity
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



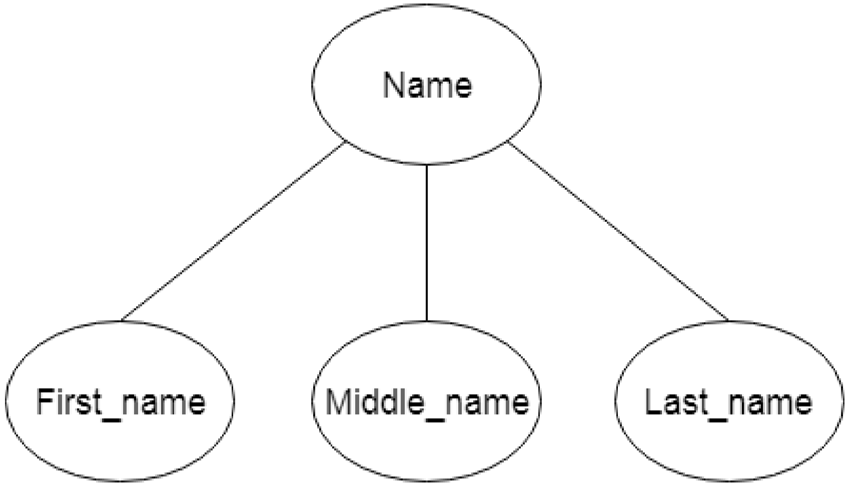
a. Key Attribute
The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



ER Diagram

b. Composite Attribute

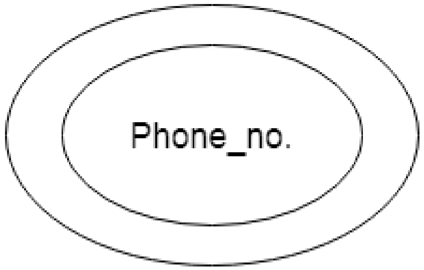
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

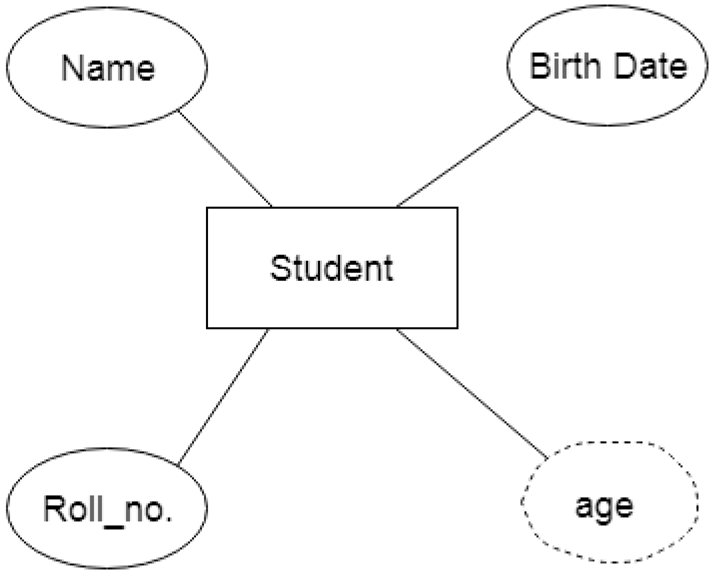
An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

For example, a student can have more than one phone number.



ER Diagram

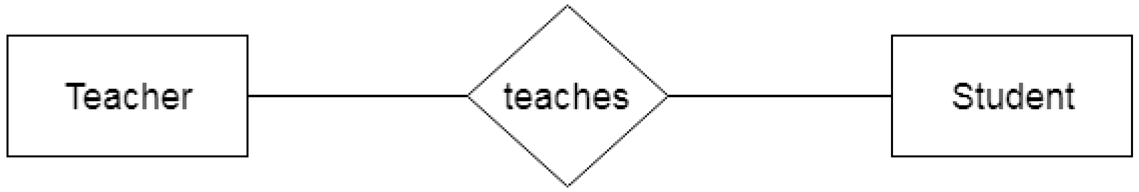
d. Derived Attribute
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.
For example, A person's age changes over time and can be derived from another attribute like Date of birth.



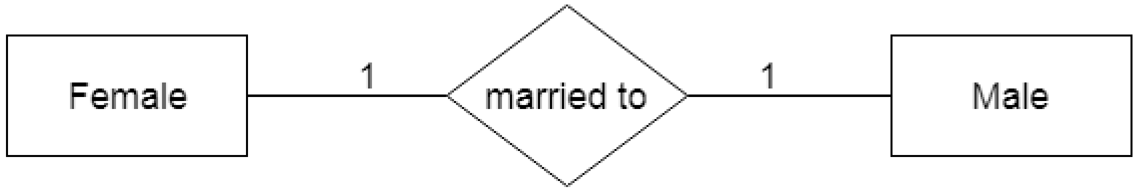
ER Diagram

3. Relationship

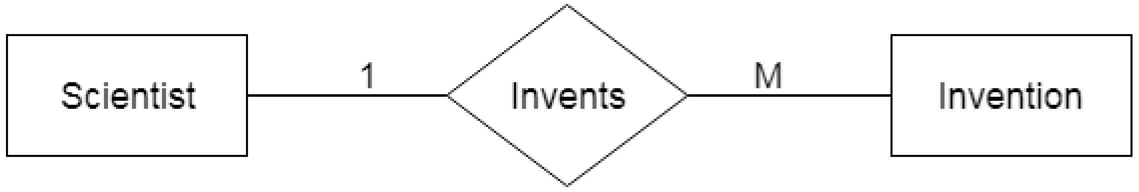
A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



a. One-to-One Relationship
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.
For example, A female can marry to one male, and a male can marry to one female.

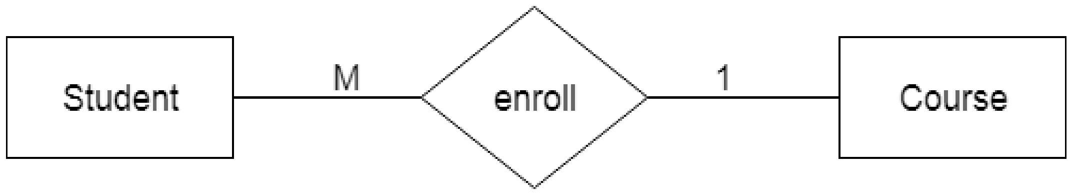


b. One-to-many relationship
When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.

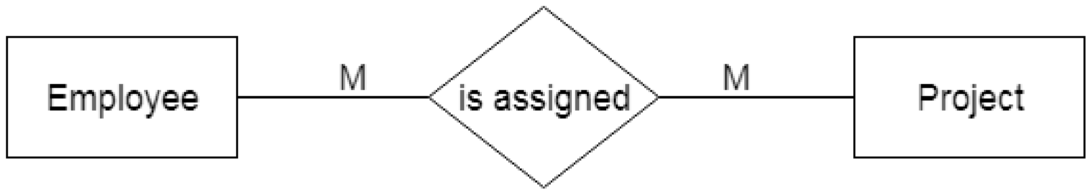


ER Diagram

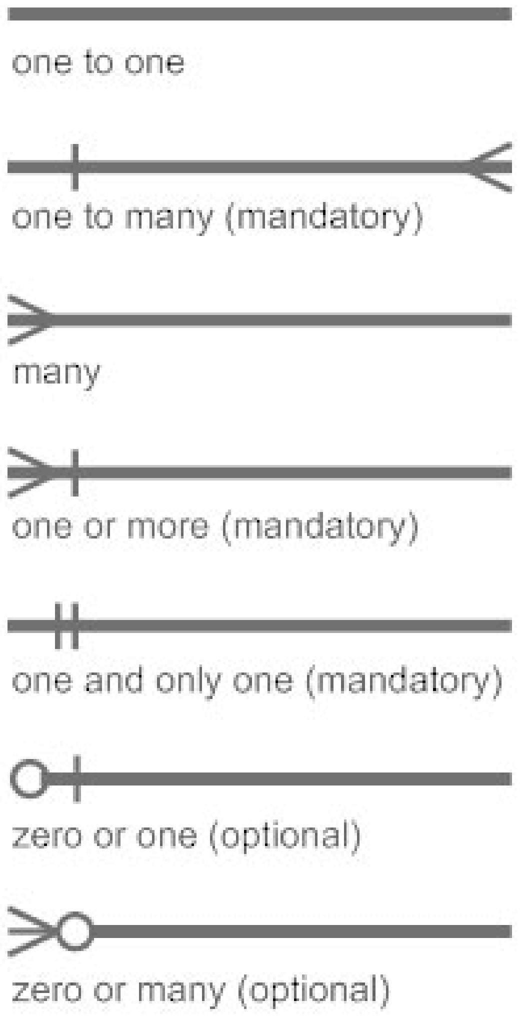
c. Many-to-one relationship
When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.
For example, Student enrolls for only one course, but a course can have many students.

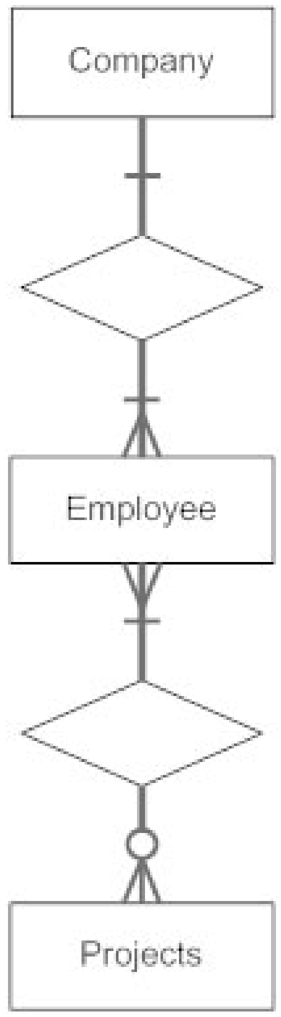


d. Many-to-many relationship
When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.
For example, Employee can assign by many projects and project can have many employees.



ER Diagram

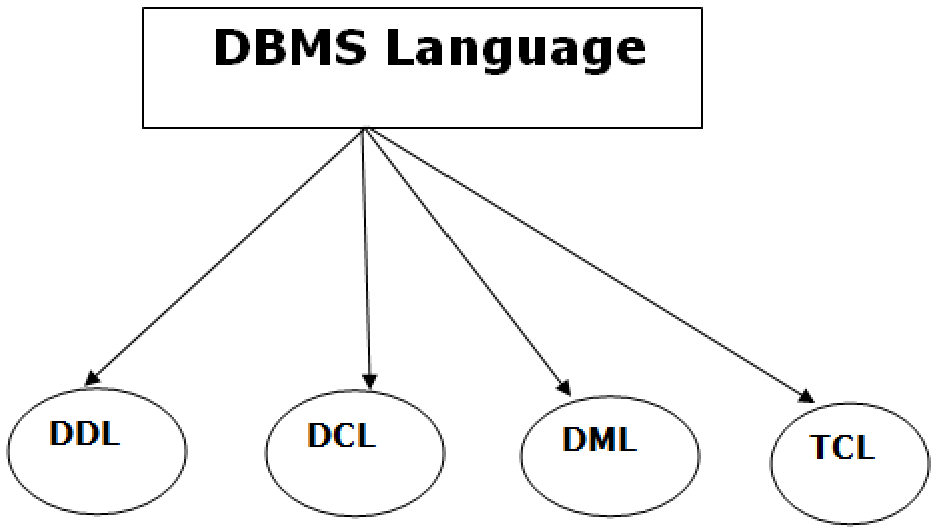

one to one
one to many (mandatory)
many
one or more (mandatory)
one and only one (mandatory)
zero or one (optional)
zero or many (optional)


Company
Employee
Projects

Question

- Create a ER diagram of any application you find in your environment.

Database Language



Database Language

Data Definition Language

- **DDL** stands for **D**ata **D**efinition **L**anguage.
- It is used to define database structure or pattern.
- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

- **Create:** It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

Data Definition Language (DDL)

- 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

Database Language

Data Manipulation Language

- **DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

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

Database Language

Data Control Language

- **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.
- (But in Oracle database, the execution of data control language does not have the feature of rolling back.)
- Here are some tasks that come under DCL:
- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.
- There are the following operations which have the authorization of Revoke:
- CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

Database Language

Transaction Control Language

- TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.
- Here are some tasks that come under TCL:
- **Commit:** It is used to save the transaction on the database.
- **Rollback:** It is used to restore the database to original since the last Commit.

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

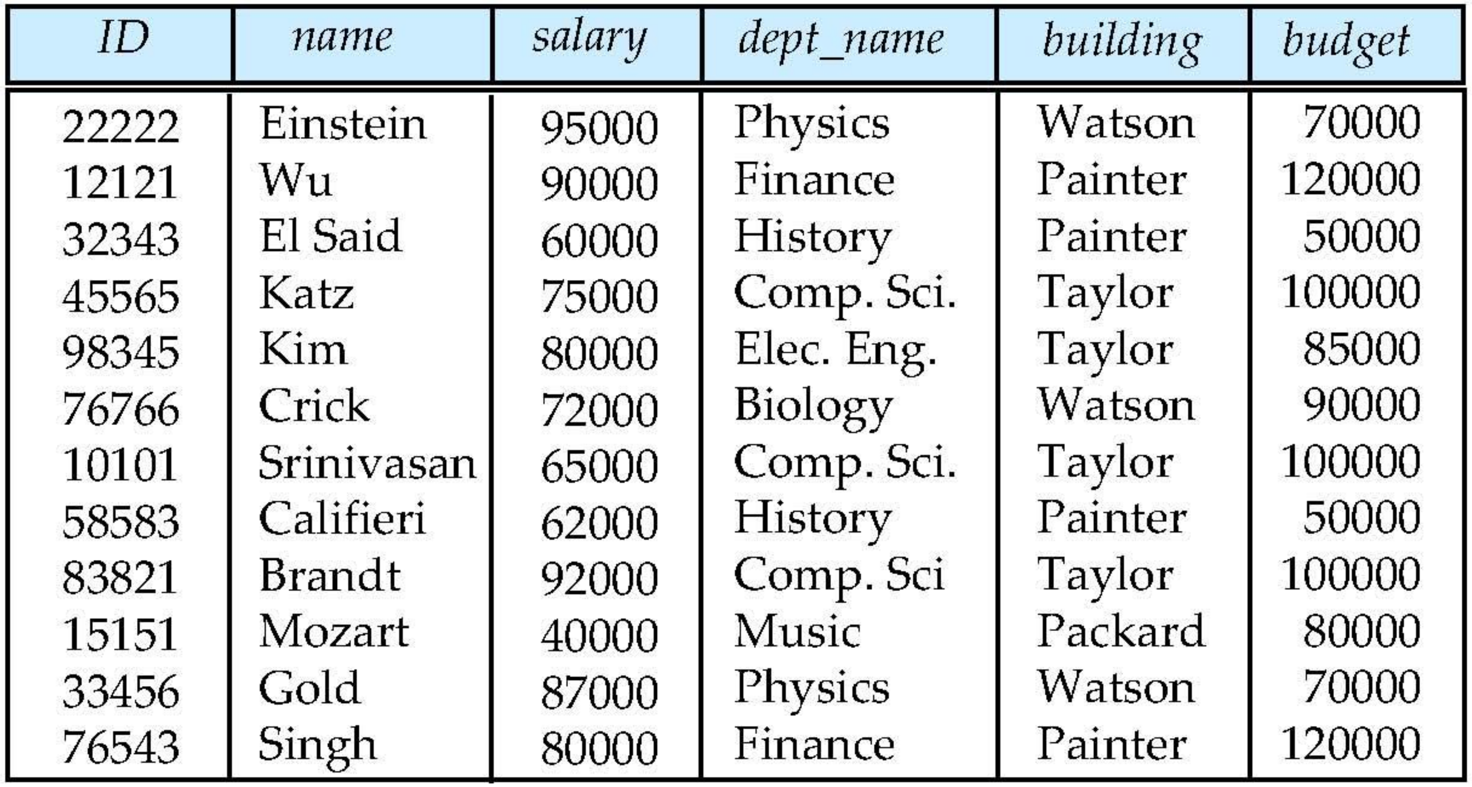
Database Design

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 (Chapter 7)
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory (Chapter 8)
    - 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

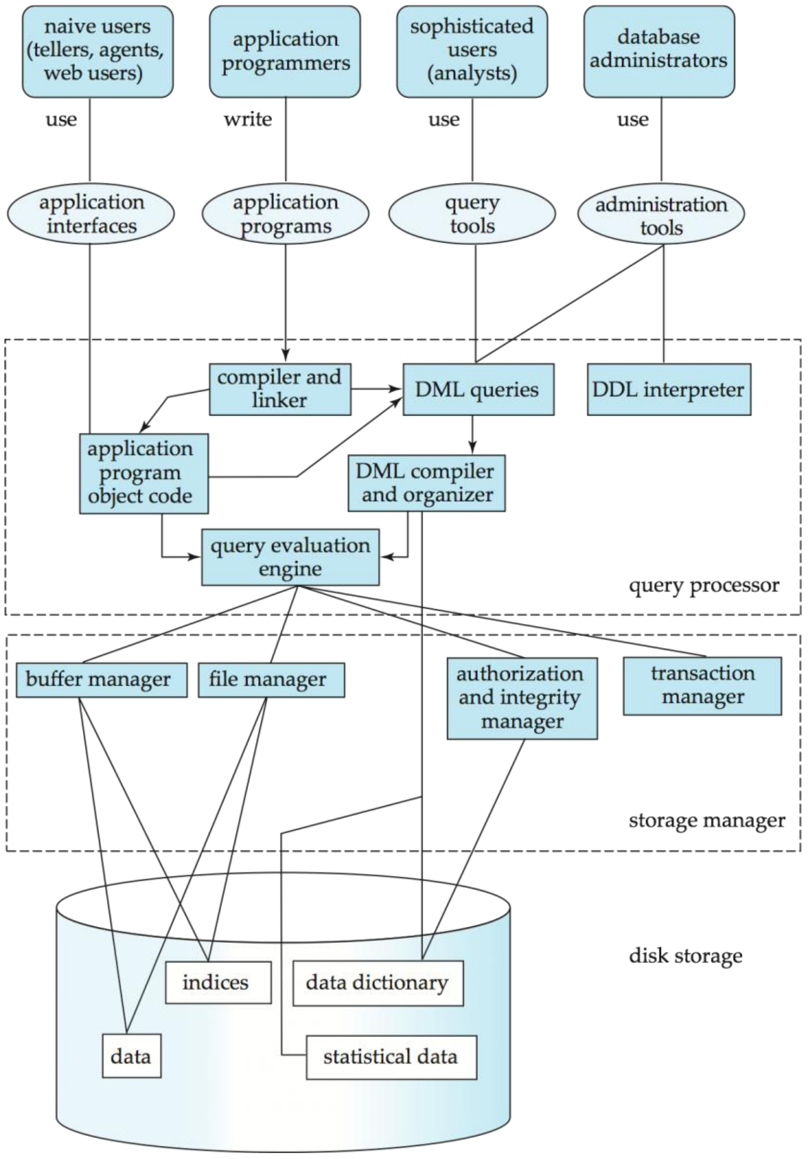
- Defined by the WWW Consortium (W3C)
- Originally intended as a **document markup language** not a database language
- The ability to **specify new tags**, and to **create nested tag** structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation

  data interchange formats.
- A wide variety of tools is available for **parsing**, **browsing** and **querying** XML documents/data

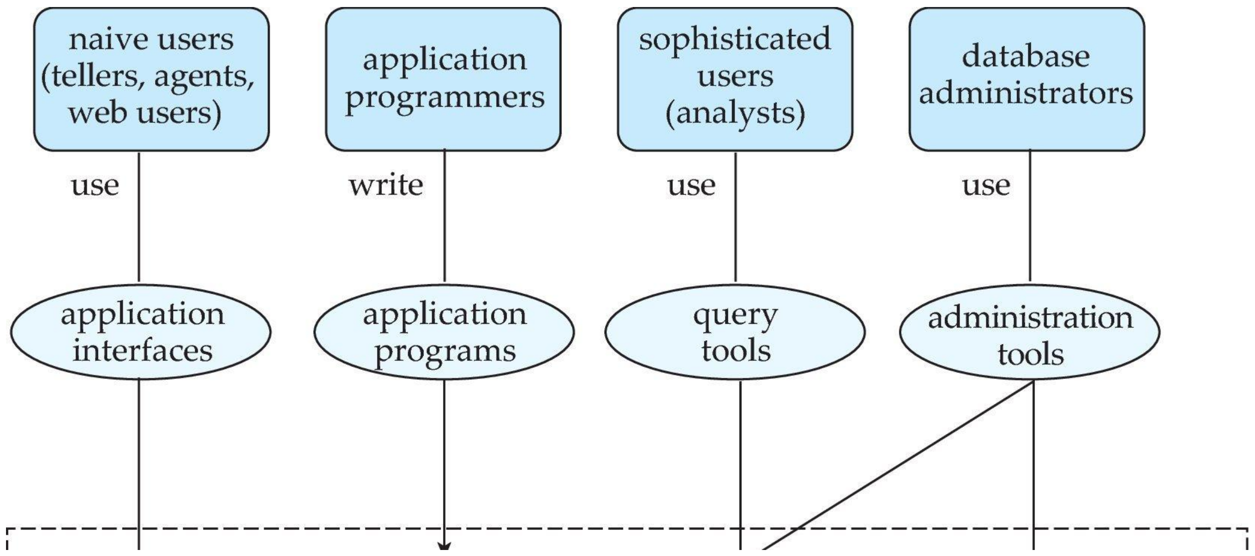
Database System Internals

Database Engine

- Storage manager
- Query processing
- Transaction manager



Database Users and Administrators

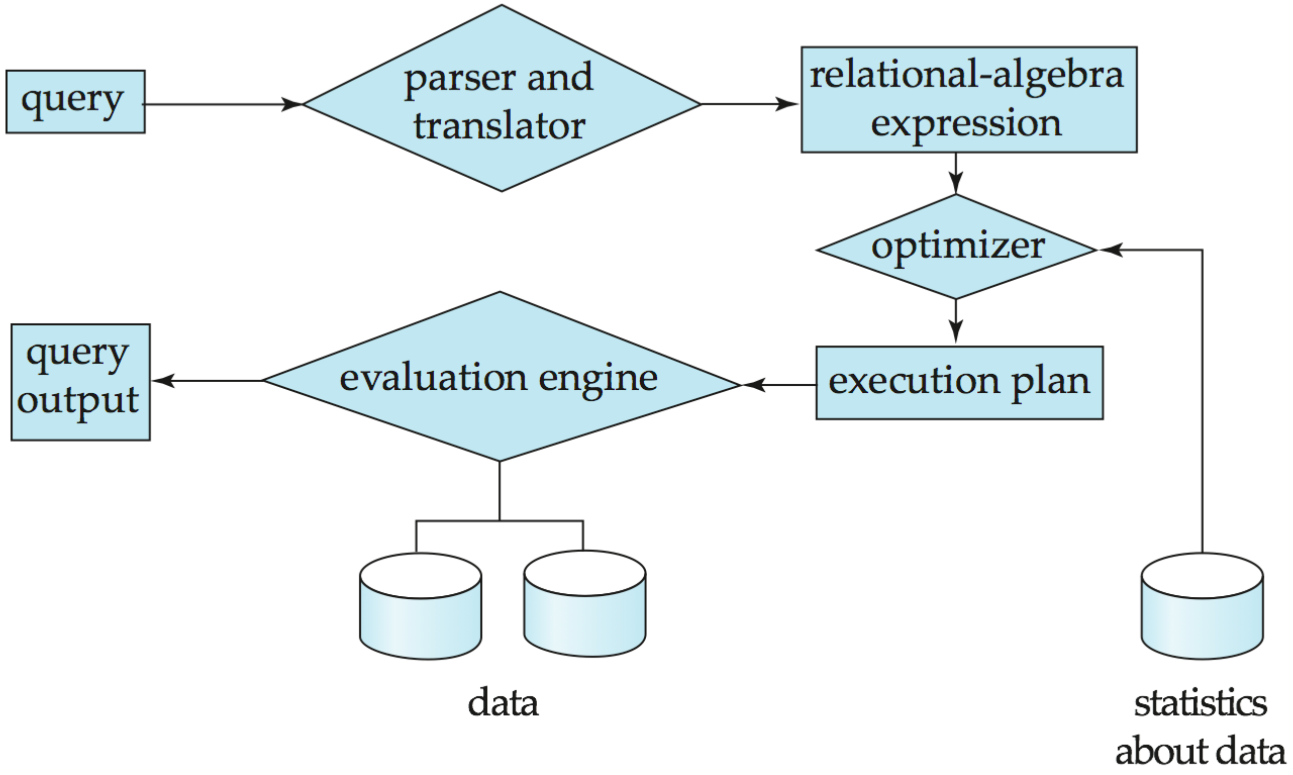


Storage Management

- **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

1. Parsing and translation
2. Optimization
3. Evaluation



Query Processing (Cont.)

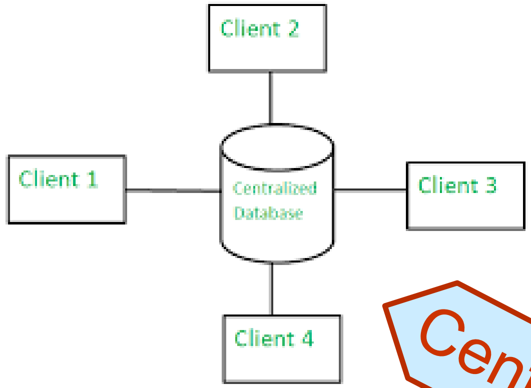
- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about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

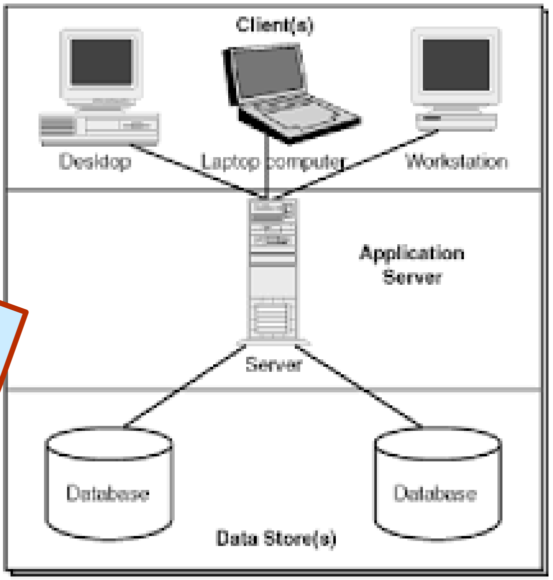
Transaction Management

- 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
- **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 Architecture

Architecture of a database systems is greatly influenced by the underlying computer system on which the database is running



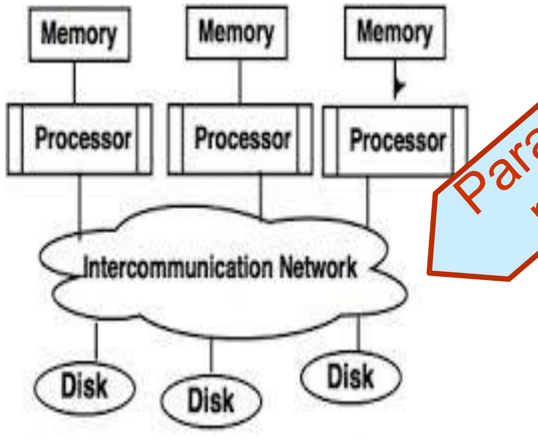


Centralized

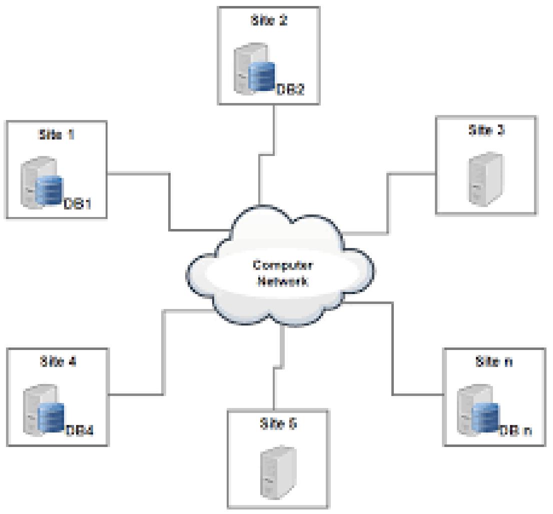
Client-server

Parallel (multi-processor)

Distributed



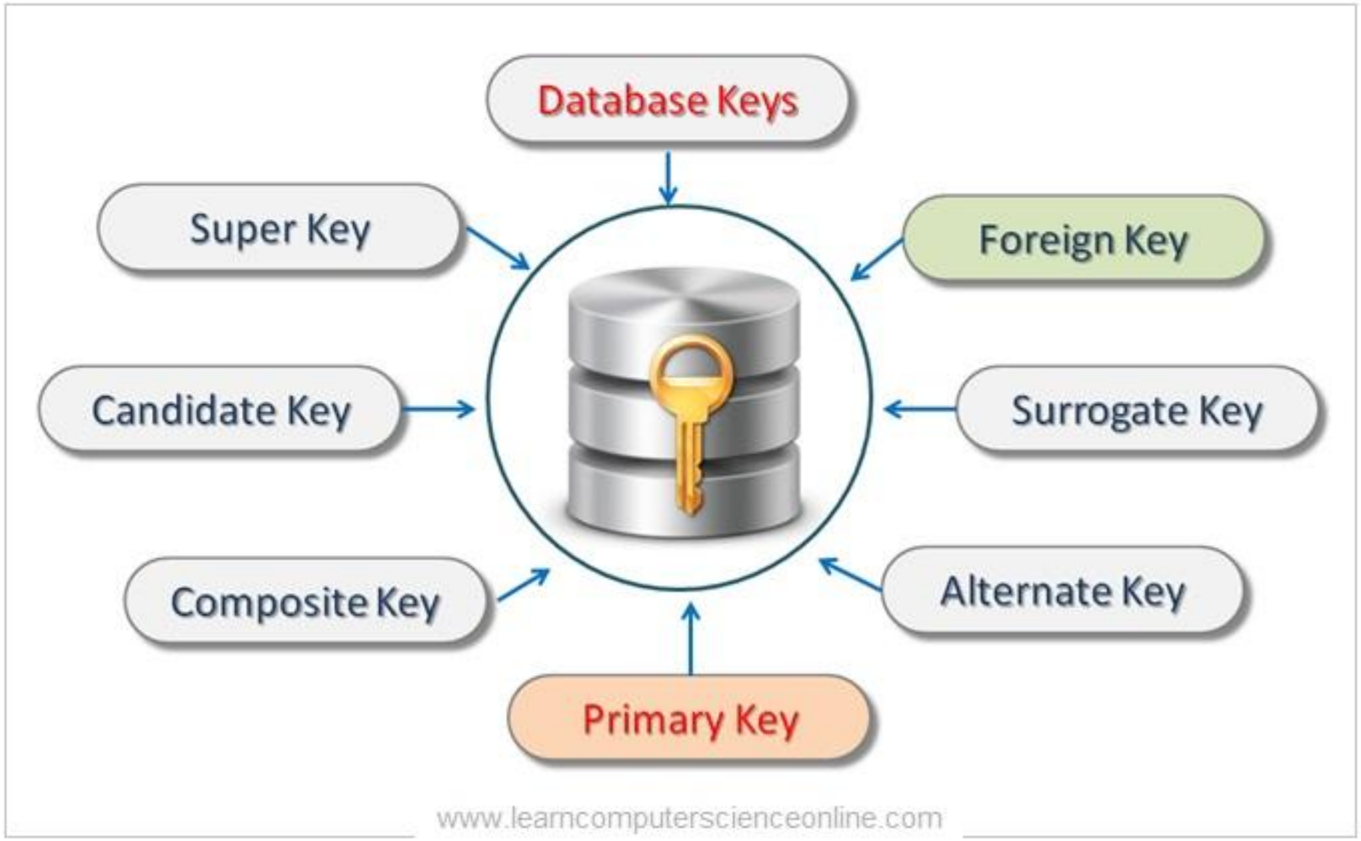
Shared disk system in Parallel Databases



Keys in DBMS

KEYS in DBMS

- is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table).
- They allow you to find the relation between two tables.
- Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.
- Key is also helpful for finding unique record or row from the table.
- Allows you to establish a relationship between and identify the relation between tables
- Help you to enforce identity and integrity in the relationship.



https://www.guru99.com/dbms-keys.html

Types of Keys in Database Management System

- **Super Key -** A super key is a group of single or multiple keys which identifies rows in a table.
- **Primary Key -** is a column or group of columns in a table that uniquely identify every row in that table.
- **Candidate Key -** is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- **Alternate Key -** is a column or group of columns in a table that uniquely identify every row in that table.
- **Foreign Key -** is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- **Compound Key -** has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.
- **Composite Key -** An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.
- **Surrogate Key -** An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

Types of Keys in Database Management System

- **Super Key -** A super key is a group of single or multiple keys which identifies rows in a table.
- **Primary Key -** is a column or group of columns in a table that uniquely identify every row in that table.
- **Candidate Key -** is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- **Alternate Key -** is a column or group of columns in a table that uniquely identify every row in that table.
- **Foreign Key -** is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- **Compound Key -** has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.
- **Composite Key -** An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.
- **Surrogate Key -** An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

Keys

- Let K $\subseteq$ R
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*
- Superkey *K* is a **candidate key** if *K* is minimal
  Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - which one?
- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - Example – *dept_name* in *instructor* is a foreign key from *instructor* referencing *department*

End

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 in widespread use
  - 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
  - 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
- 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, ..