Overview of DBMS

Uma Seshadri

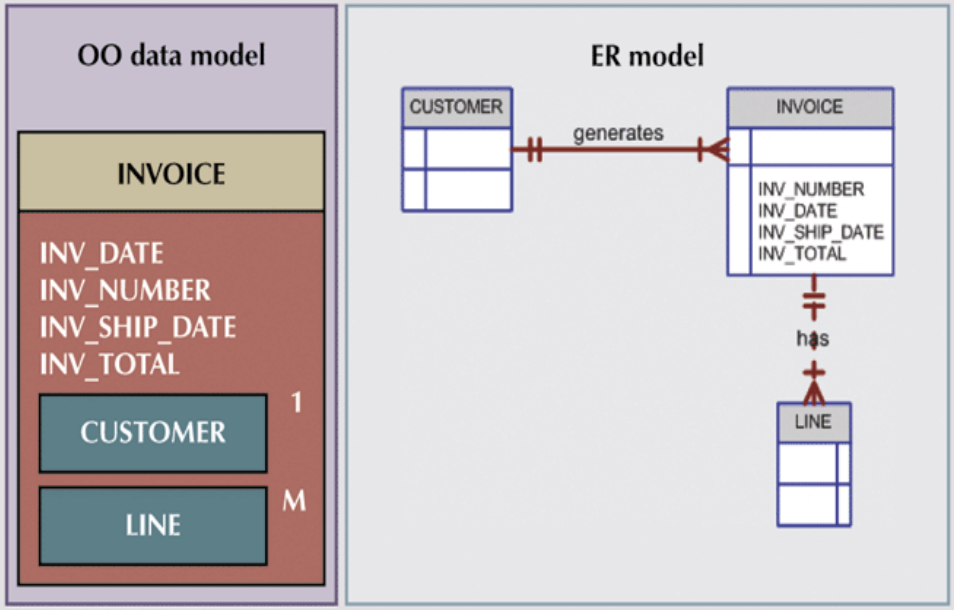
Object-Oriented Database

- Semantic Data Model (SDM)
  - Modeled both data and their relationships in a single structure (object)
    - Developed by Hammer & McLeod in 1981
- Object-oriented concepts became popular in 1990s
  - Modularity facilitated program reuse and construction of complex structures
  - Ability to handle complex data types (e.g. multimedia data)
- Object-Oriented Database Model (OODBM)
  - Maintains the advantages of the ER model but adds more features
  - Object = entity + relationships (between & within entity)
    - consists of attributes & methods
      - attributes describe properties of an object
      - methods are all relevant operations that can be performed on an object
    - self-contained abstraction of real-world entity
  - Class = collection of similar objects with shared attributes and methods
    - e.g. EMPLOYEE class = (employ1 object, employ2 object, …)
    - organized in a class hierarchy
      - e.g. PERSON > EMPLOYEE, CUSTOMER
  - Incorporates the notion of inheritance
    - attributes and methods of a class are inherited by its descendent classes

OO Database Model vs. E-R Model

OODBM:
- can accommodate relationships within a object
- objects to be used as building blocks for autonomous structures

FIGURE 2.7 A comparison of the OO model and the ER model



Object-Oriented Database: Pros & Cons

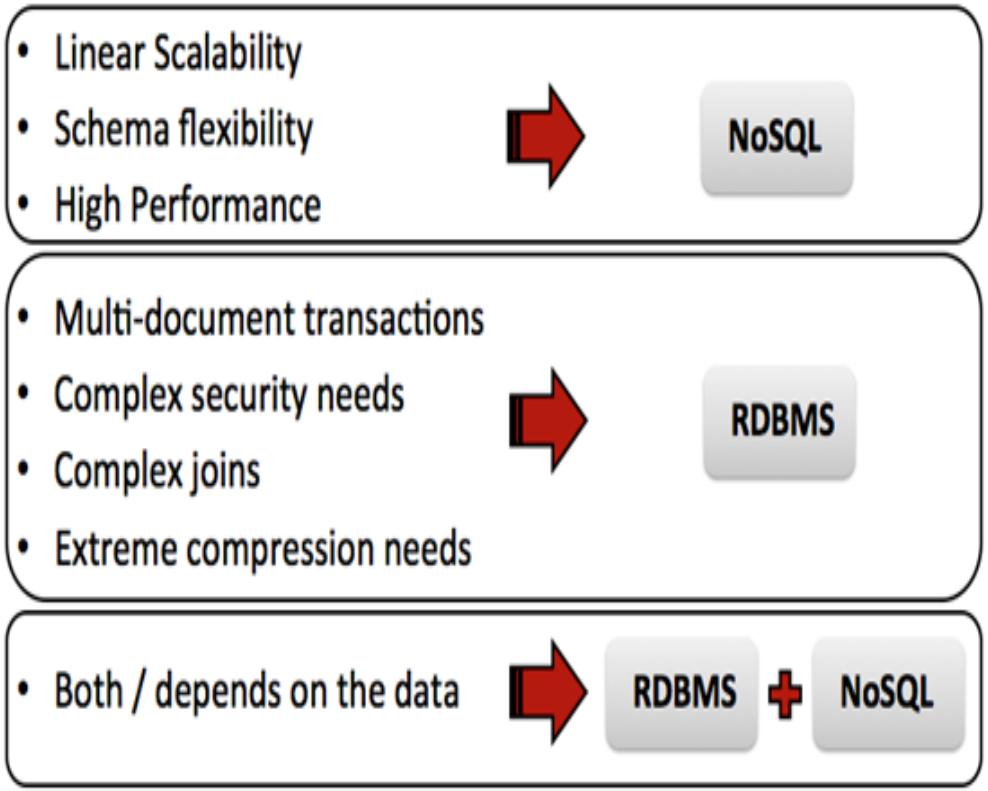
- Advantages
  - Semantic representation of data
    - fuller and more meaningful description of data via object
  - Modularity, reusability, inheritance
  - Ability to handle
    - complex data
    - sophisticated information requirements
- Disadvantages
  - Lack of standards
    - no standard data access method
  - Complex navigational data access
    - class hierarchy traversal
  - Steep learning curve
    - difficult to design and implement properly
  - More system-oriented than user-centered
  - High system overhead
    - slow transactions

Web Database

- Internet is emerging as a prime business tool
  - Shift away from models (e.g. relational vs. O-O)
  - Emphasis on interfacing with the Internet
- Characteristics of “Internet age” databases
  - Flexible, efficient, and secure Internet access
  - Support for complex data types & relationships
  - Seamless interfaces with multiple data sources and structures
  - Ease of use for end-user, database architect, and database administrator
    - Simplicity of conceptual database model
    - Many database design, implementation, and application development tools
    - Powerful DBMS GUI

NoSQL

- NoSql is not literally “no sql”. They are non relational data stores.
- Next Generation Databases being non-relational, distributed, open-source and horizontally scalable have become a favorite back end storage for cloud community . High performance is the driving force.



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.

Database Engine

- Storage manager
- Query processing
- Transaction manager

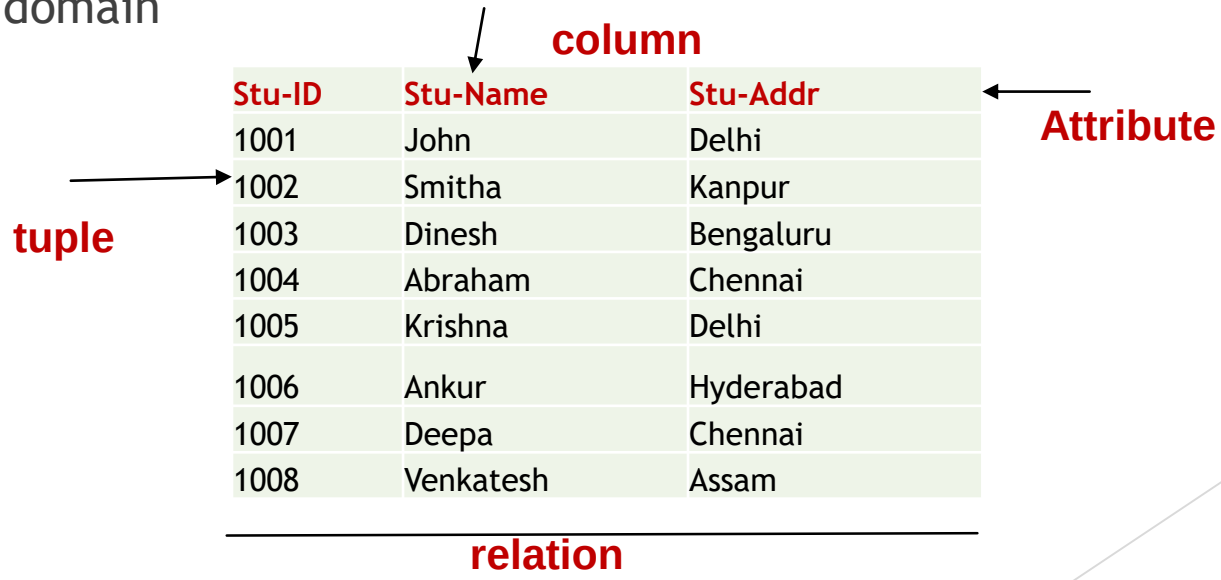
NoSQL

- Pros
  - open source (Cassandra, CouchDB, Hbase, MongoDB, Redis)
  - Elastic scaling
  - Key-value pairs, easy to use
  - Useful for statistical and real-time analysis of growing lists of elements (tweets, posts, comments)
- Cons
  - Security (No ACID: ACID (Atomicity, Consistency, Isolation, Durability)
  - No indexing support
  - Immature
  - Absence of standardization

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumber": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

DBMS - Data Model : Relational Model

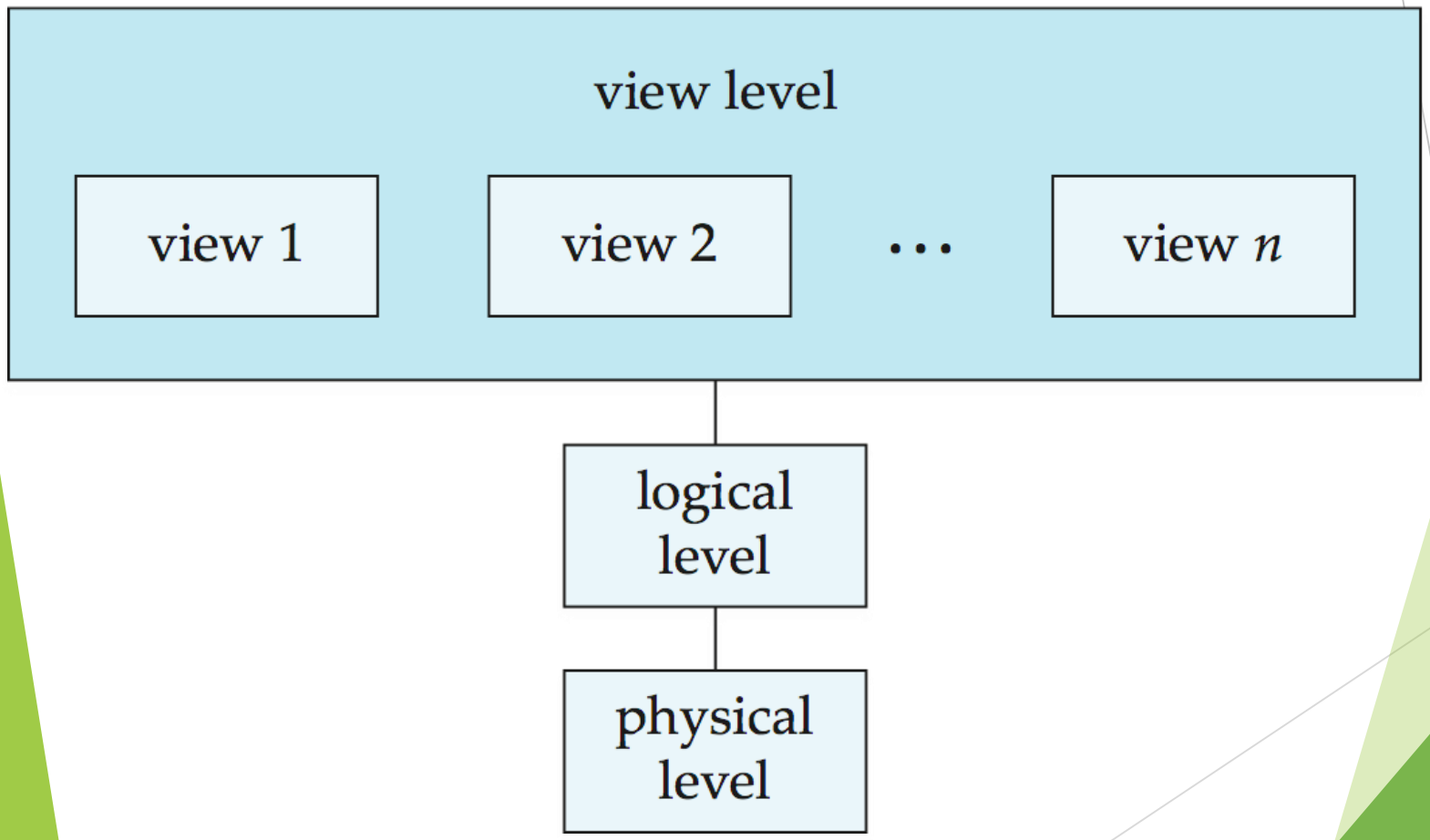
- Data is stored in tables called **relations**
- Table is defined in n-ary
- Each row in a relation contains a unique value
- Relations can be normalized
- In normalized relations, values saved are atomic values
- Each column in a relation contains values from a same domain



Stu-ID	Stu-Name	Stu-Addr
1001	John	Delhi
1002	Smitha	Kanpur
1003	Dinesh	Bengaluru
1004	Abraham	Chennai
1005	Krishna	Delhi
1006	Ankur	Hyderabad
1007	Deepa	Chennai
1008	Venkatesh	Assam

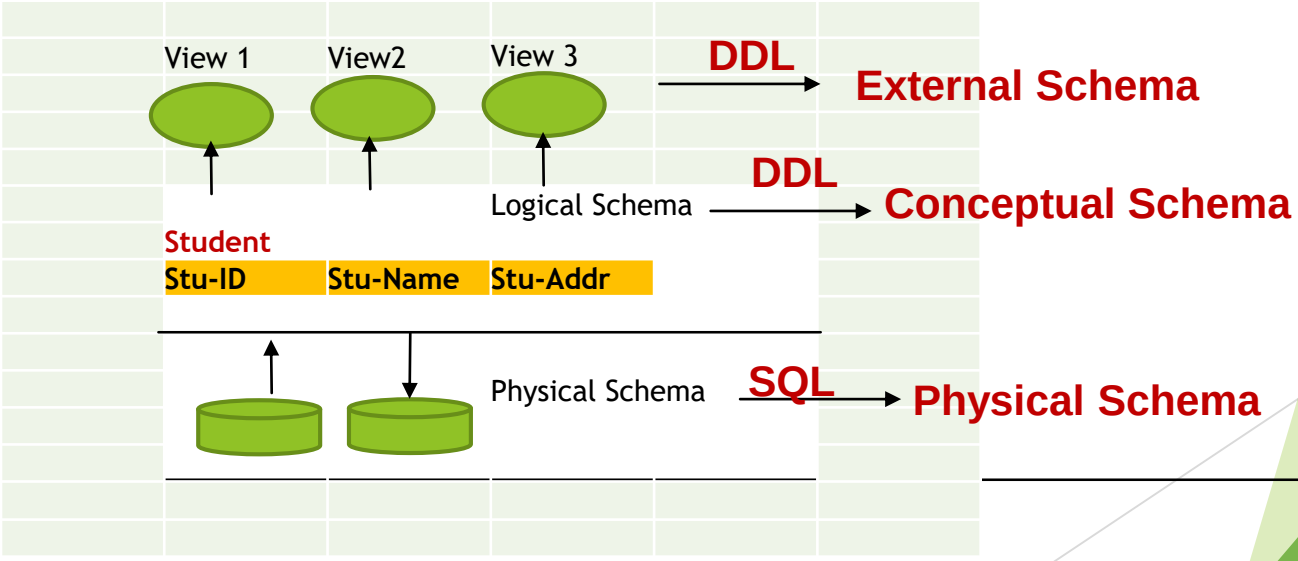
DBMS - Views

An architecture for a database system



DBMS - Data Schemas

- Represents the logical view of the entire database
- Designed before physical database is created
- Defines its entities and the relationship among them
- Defines how the data is organized
- Defines how the relations among them are associated.
- Formulates all the constraints that are to be applied on the data



DBMS - Levels of Abstraction

- **Physical Schema:** describes how a record is stored.

- **Logical Schema:** describes data stored in database(table), and the relationships among the data, constraints applied to data.

Students(Stu-ID : string;
Stu-name : string;
Stu-addr : string)

Stu-ID	Stu-Name	Stu-Addr
1001	John	Delhi
1002	Smitha	Kanpur
1003	Dinesh	Bengaluru
1004	Abraham	Chennai

- **View level:** application programs hide details of data types and information.

DBMS - Database Instances

- State of operational database with data at any given time.
- Snapshot of the database.
- Change with time
- Exists in a valid state (validations, constraints, and conditions that the database designers have imposed
  - Analogous to the value of a variable

DBMS -Data independence

- **Logical Data Independence**
  - mechanism, which liberalizes itself from actual data stored on the disk.
  - Any changes done to table format table format will not affect data residing on the disk.
- **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.

Note: Similar to types and variables in programming languages

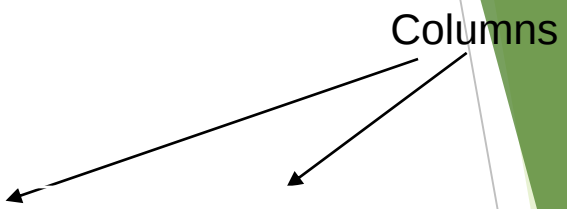
- All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.
- For example, in case we want to change or upgrade the storage system itself – suppose we want to replace hard-disks with SSD – it should not have any impact on the logical data or schemas.

Data Models

- 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

- Data is stored in various tables.
- Tabular data in the relational model



Rows

Student ID	Student Name	Department Name	Course Name
1001	Arun	Mechanical	Engg.Maths
1002	Bhaskar	Electrical	Chemistry
1003	John	Computer Science	Cloud computing
1004	Kumar	Electronics	Data Analytics
1005	Sara	Electrical	DSP
1006	Abraham	Mechanical	CAD
1007	Anil Kumar	Computer Science	Programming Lang
1008	Sweta	Mechanical	Therrmo dynamics
1009	Rubeena	Computer Science	AI
1010	Neela	Civil	Structures

Students Database

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

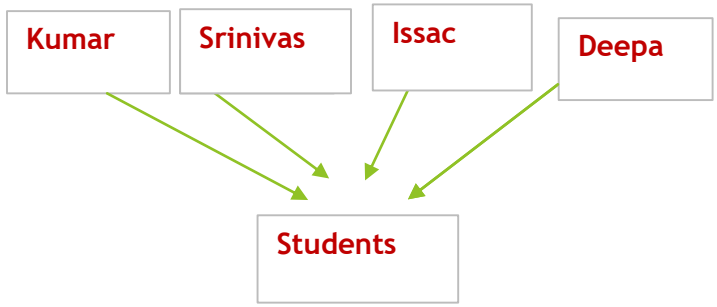
dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

DBMS - Generalization

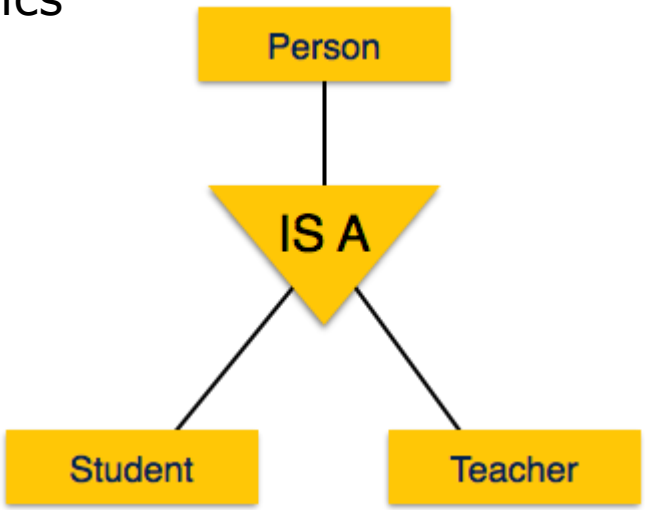
Generalization:

- entities are clubbed together to represent a more generalized view
- generalized entities contain the properties of all the generalized entities



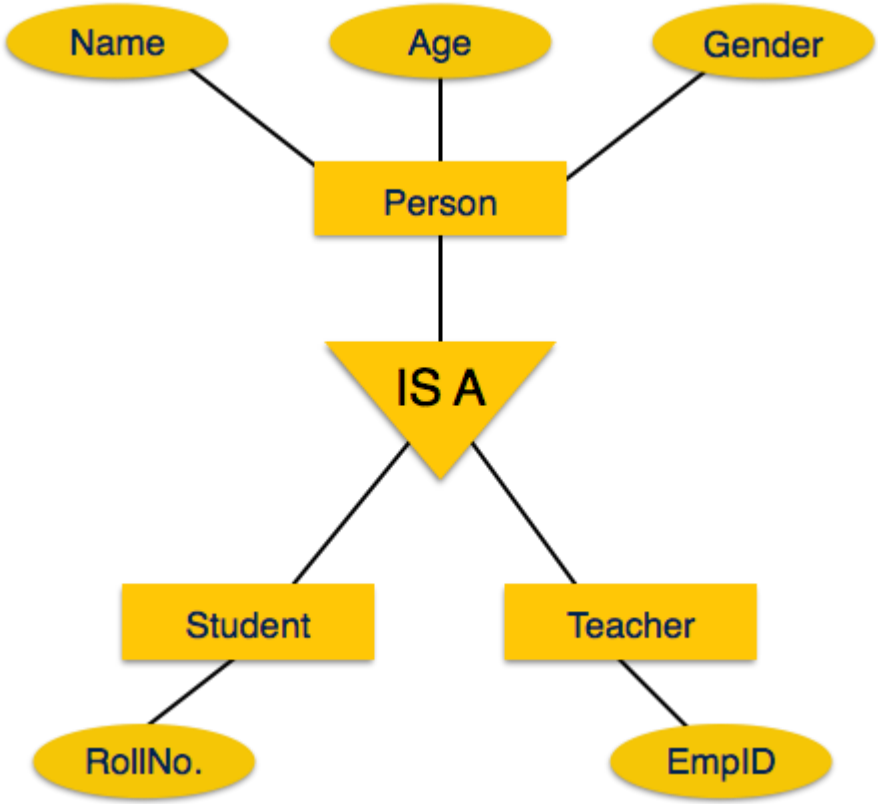
Specialization

- Specialization is the opposite of generalization.
- Group of entities is divided into sub-groups based on their characteristics



DBMS - Inheritance

- Important feature of Generalization and Specialization.
- Allows lower-level entities to inherit the attributes of higher-level entities



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?

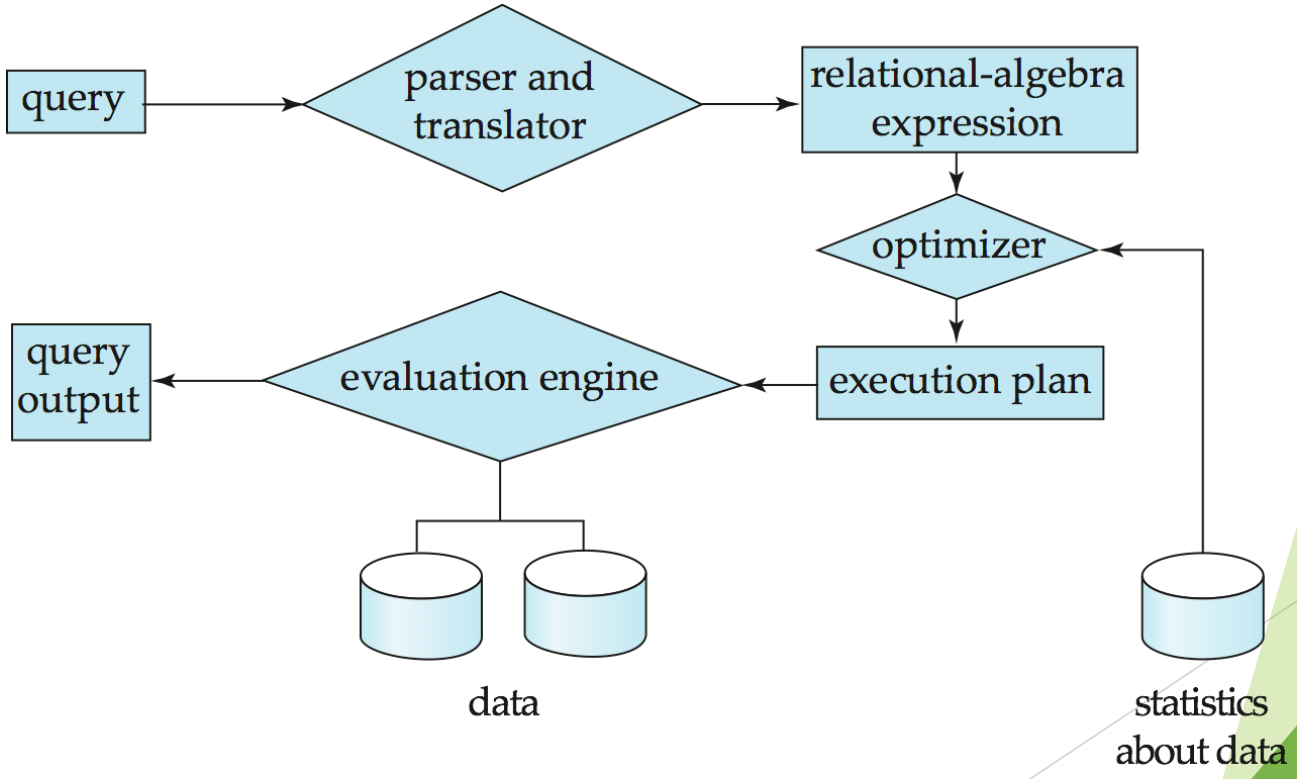
ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	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

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



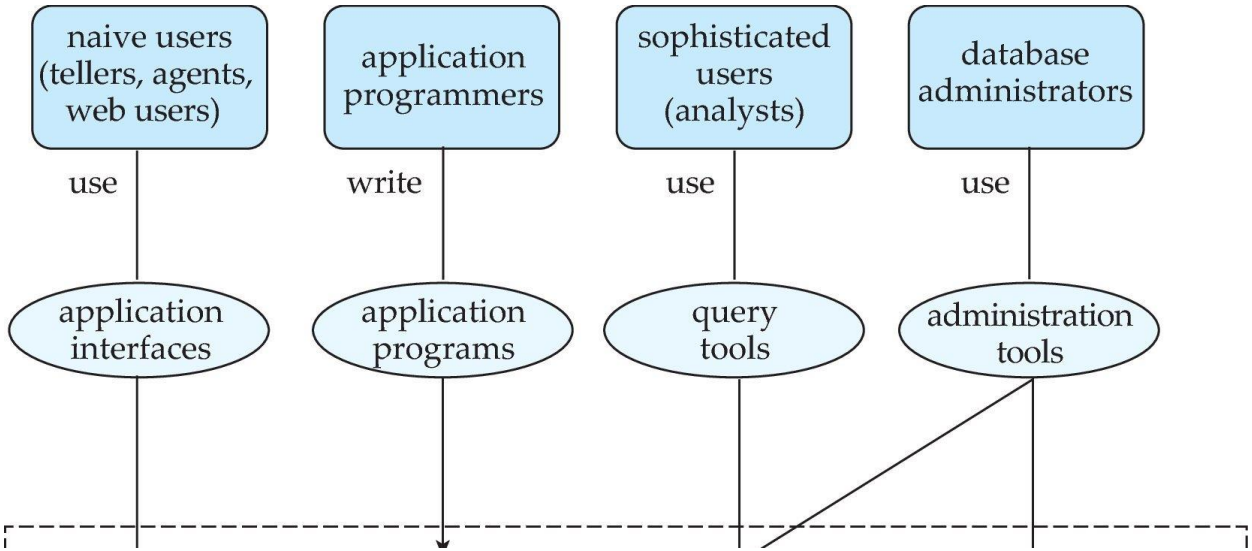
Query Processing (Cont.)

- 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

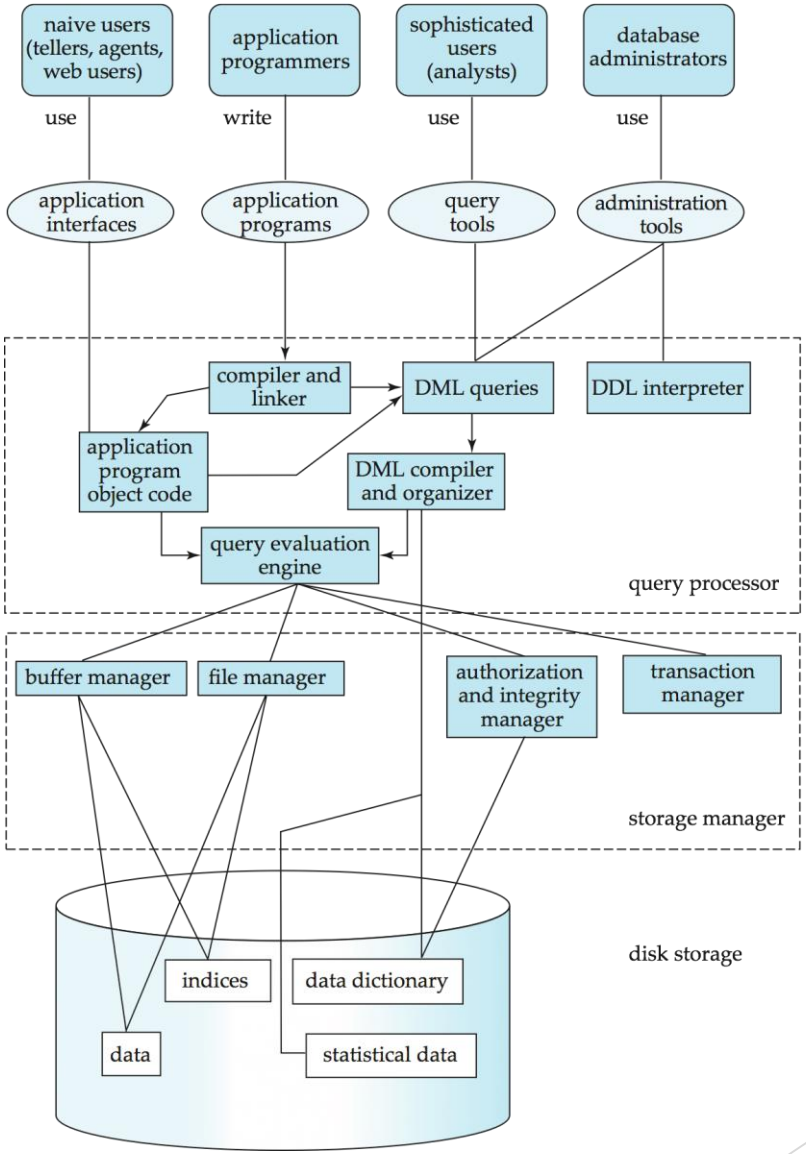
Transaction Management

- 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 Users and Administrators



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

End of Chapter 1