Database system Introduction

- What is Database Systems?
  - Database system concepts
  - Database Applications
  - Purpose of Database Systems
- Data Modeling
  - Data Models
  - Data abstraction
  - Database Design
- Database Languages
- DBMS System Structure
  - Storage manager
  - Query processing
  - Transaction manager
- Database and Application Architectures
- Database Users
- History of Database Systems

Database System Concepts

- Database System contains information about a particular organization
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database

  Collection of interrelated data (about a particular Reason)
  - Highly valuable
  - Relatively large --need Well organized
  - Accessed by multiple users and applications, often at the same time.
  - Safety of the information stored, despite system crash or unauthorized access
- DBMS is a software: Set of programs to access the data
  - Be able to manage databases.
  - Provides an environment that is both *convenient* and *efficient* to use.

Database Applications

- Enterprise Information
  - Sales: customers, products, purchases
  - Accounting: payments, receipts, assets
  - Human Resources: Information about employees, salaries, payroll taxes.
- Manufacturing: management of production, inventory, orders, supply chain.
- Banking and finance
  - customer information, accounts, loans, and banking transactions.
  - Credit card transactions
  - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- Universities: Teacher, student, registration, grades
- Airlines: reservations, schedules
- Telecommunication: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards

Database Applications

- Web-based services
  - Online retailers: order tracking, customized recommendations
  - Online advertisements
- Document databases
- Navigation systems: For maintaining the locations of varies places of interest along with the exact routes of roads, train systems, buses, etc.
- Social medias likes WeChat: Contacts, messages, group chats, hongbao, payments,

- Use of database grew fast in four decades
  - A: Since the early days, very few people interacted directly with database system, custom did not know the backend database, they got printed report, etc.
  - B: Later, Some customs access system via phone interface.
  - C: Starting from late 90s, Direct user access via Internet.

A and C are most common used nowadays

Purpose of Database Systems

- In the early days, database applications were built on top of file systems
  - *Thanking about the computing architecture in early days?*

- 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*) been hard coded in the program.
    - Hard to add new constraints or change existing ones

Purpose of Database Systems (Cont.)

- Drawbacks of using file systems (cont.)
  - Atomicity of updates
    - Failures may leave database in an inconsistent state with partial updates carried out
    - *E.g. transfer of funds from one account to another should either complete or not happen at all*
  - Concurrent access by multiple users
    - Concurrent accessed needed for performance
    - Uncontrolled concurrent accesses can lead to inconsistencies
      - *E.g. Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time*
  - Security problems
    - Data access authorization
    - Data recovery while some incidents occurs.
      - e.g. server crash, disk failure, fire disaster, earthquake, etc.
- Database systems offer solutions to all the above problems

Data Models

- A collection of conceptual tools for describing
  - data
  - data relationships
  - data semantics
  - data constraints
- Entity-Relationship model
- Relational model
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

1. Tired closely to the physical level
2. Higher performance
3. Much difficult to program.

Relational Model

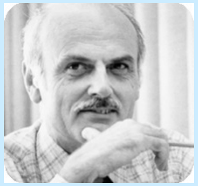
- Models a database as a collection of tables.
  - All the data is stored in various tables.
  - Each table represent either an “object” or “relationship” among the “object”s
- Example of tabular data in the relational model

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



Ted Codd
Turing Award 1981

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

Object-Based Data Models*

- Relational model: flat, atomic value, no complex type, etc.
- Object Oriented Data Model
  - Seamlessly integrate with programing languages, C++/Java/C
  - Weak support in declarative access to data
  - High performance
- Object-Relational Data Model
  - 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

Data Abstraction

- Why and What is Data Abstraction?
  - A database system is a collection of data, and a set of program to allow users to access the data;
  - A major purpose of a database system is to provide users with an **abstract view of the data**
    - Easily understood by human;
    - Easily accessed by application program;
    - Hidden certain detail of how data been stored and maintained.

- Levels of Abstraction

- Instances and Schemas

Levels of Abstraction

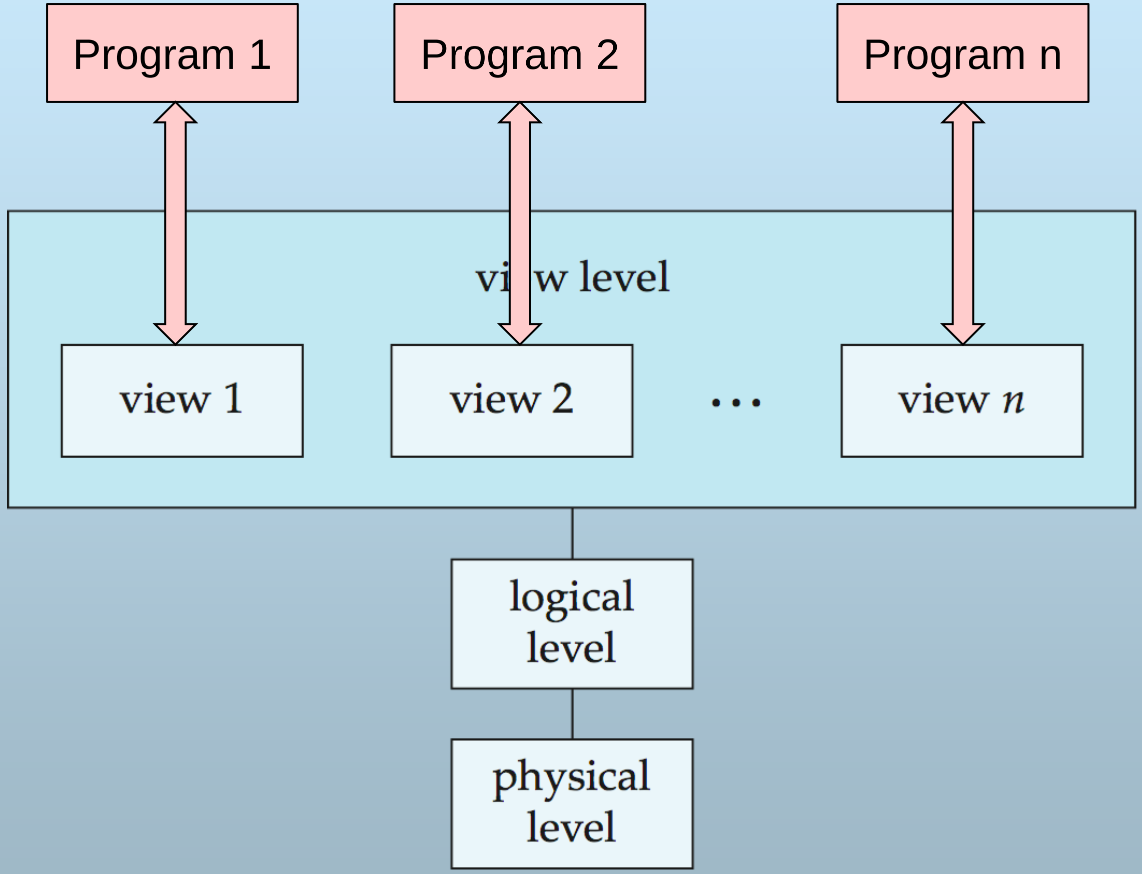
- Physical level: Describes HOW the data are stored.
- Logical level: Describes WHAT data are stored in database, and the relationships exists among the data. And don't care about how the data been stored.

  type *instructor* = **record**

  ID : string;
  name : string;
  dept_name : string;
  salary : integer;

  end;
- View level: Describes WHAT data could be accessed by application programs. View can hide details of data types, and hide information (e.g., salary) for security and privacy purposes.

Levels of Abstraction



Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - e.g., the database consists of information about a set of customers and accounts and the relationship between them)
  - Analogous to type information of a variable in a program
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
  - *However, the concept of database Instance is rare used.*

- Database Schemas
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
  - **External schema(subschema)**: database design at the view level.

Data Independency

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema, not necessary to care about how the data been stored.
  - DBMS would take the most work at the physical level.

- **Logical Data Independence** – the ability to modify the logical schema without changing the external schema
  - Applications depend on the external schema, avoid rewriting the program while the logical schema been modified in future.
  - In many cases, some of the application even don’t know the change of the database logical schema.

Database Design

Database design mainly involves the design of the database schema :

- Logical Design – Deciding on the database logical 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
  - Indexing
  - Partition

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

- *If we want to change the budget of Physics department?*
- *If Mr. Crick resigned from the university?*
- *If a new department “Math” been created?*
- *What’s the exact meaning of the “building”?*

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 6)
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory (Chapter 7)
    - Formalize what designs are bad, and test for them

Database Languages

- Data Definition Language
- Data Manipulation Language
- Database Access from Application Programs

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 tables stored in a data dictionary
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Data *storage and definition* language
    - Specifies the storage structure and access methods used
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
    - Referential integrity (**references** constraint in SQL)
    - Assertions
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Data Manipulation is:
  - The retrieval of information stored in the database;
  - The insertion/deletion of information into/from the database;
  - The modification of information stored in the database.
- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - Procedural – user specifies what data is required and how to get those data
  - Nonprocedural – user specifies what data is required without specifying how to get those data

SQL Query Language

- **SQL**: Structured Query Language
  - Most widely used database non-procedural language
  - Example: Find the instructor ID and department name of all instructors associated with a department with budget of greater than $95000.

select *instructor.ID, department.dept_name*
from *Instructor, department*
where instructor.depart_name=department.dept_name **and** department.budget > 95000

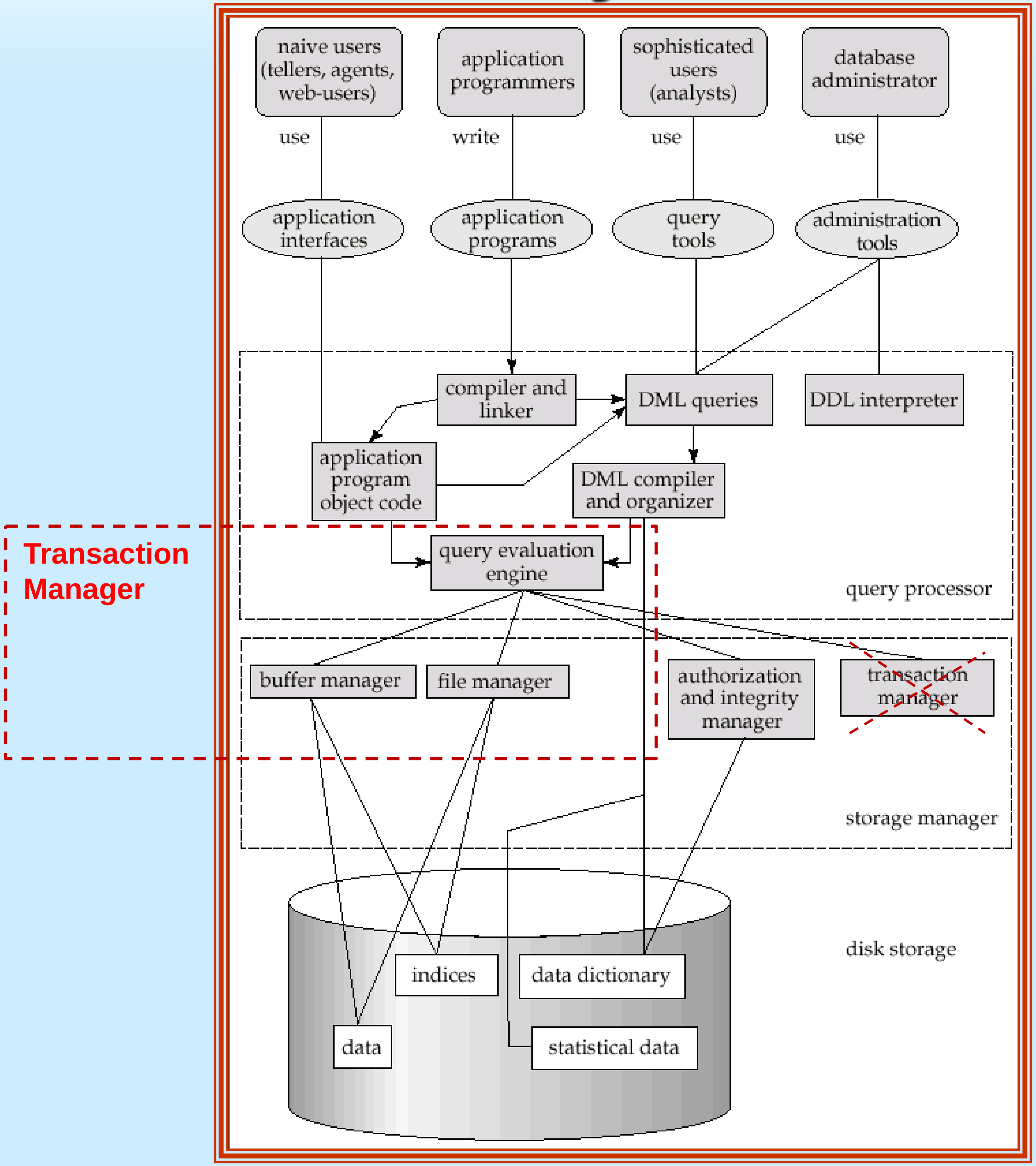
Database Access from Application Programs

- SQL is NOT a Turing machine equivalent language
  - To be able to compute complex functions SQL is usually embedded in some higher-level language
- The Database application programs are Usually written in:
  - Database Language
  - A Host language , such Java, C/C++/C#, VB, Cobol, Delphi…
- The ways to combine Database and Host languages:
  - By providing an program interface, such as: ODBC/JDBC
  - By extending the host language to embed DML.

Database Engine

- A DBMS is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a DBMS can be divided into
  - The storage manager,
  - The query processor component,
  - The transaction management component.

Overall DBMS System Structure



Storage Management

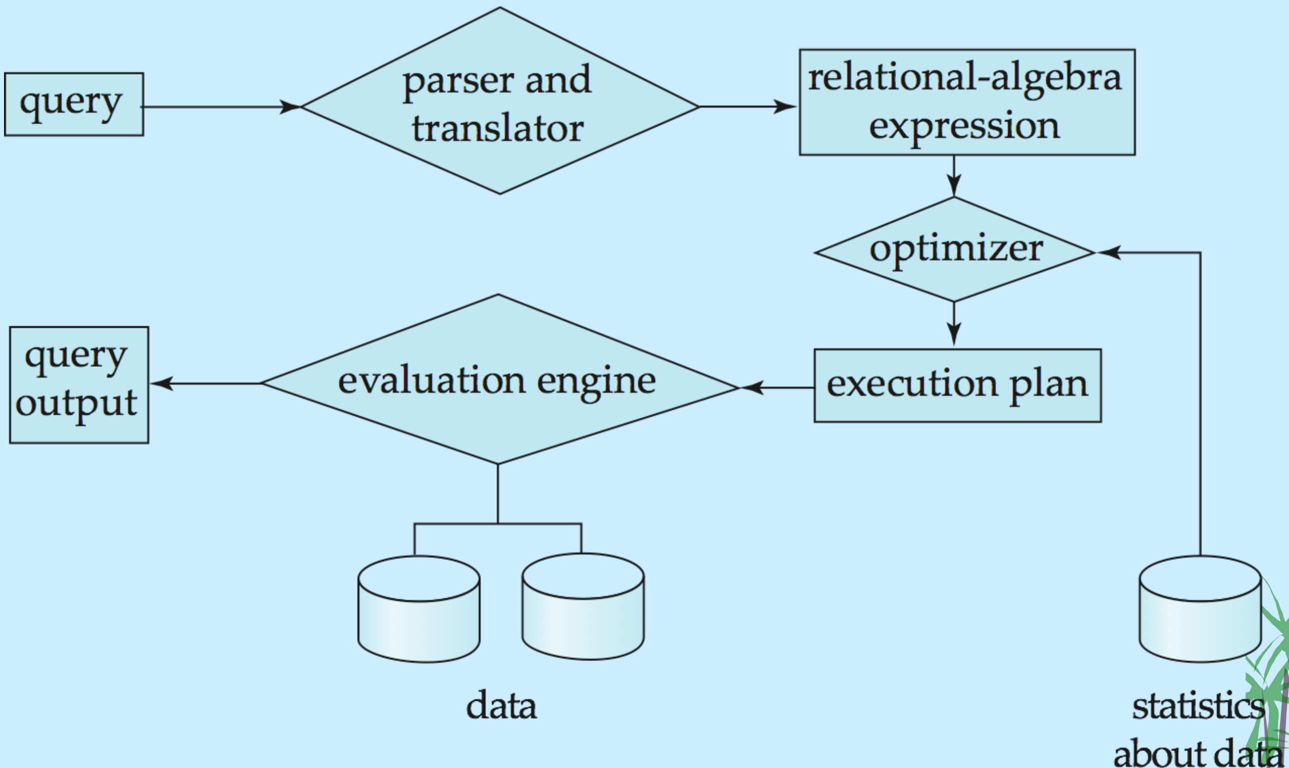
- Storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - efficient storing, retrieving and updating of data
  - File manager: interaction with the OS file system
  - Buffer manager
  - Data dictionary
- Issues:
  - Authorization and integrity
  - File organization and storage access
  - Indexing and hashing

Query Processor

- The query processor components include:
  - DDL interpreter -- interprets DDL statements and records the definitions in the data dictionary.
  - DML compiler -- translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
    - The DML compiler performs query optimization; that is, it picks the lowest cost evaluation plan from among the various alternatives.
  - Query evaluation engine -- executes low-level instructions generated by the DML compiler.

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Query Optimization

- 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

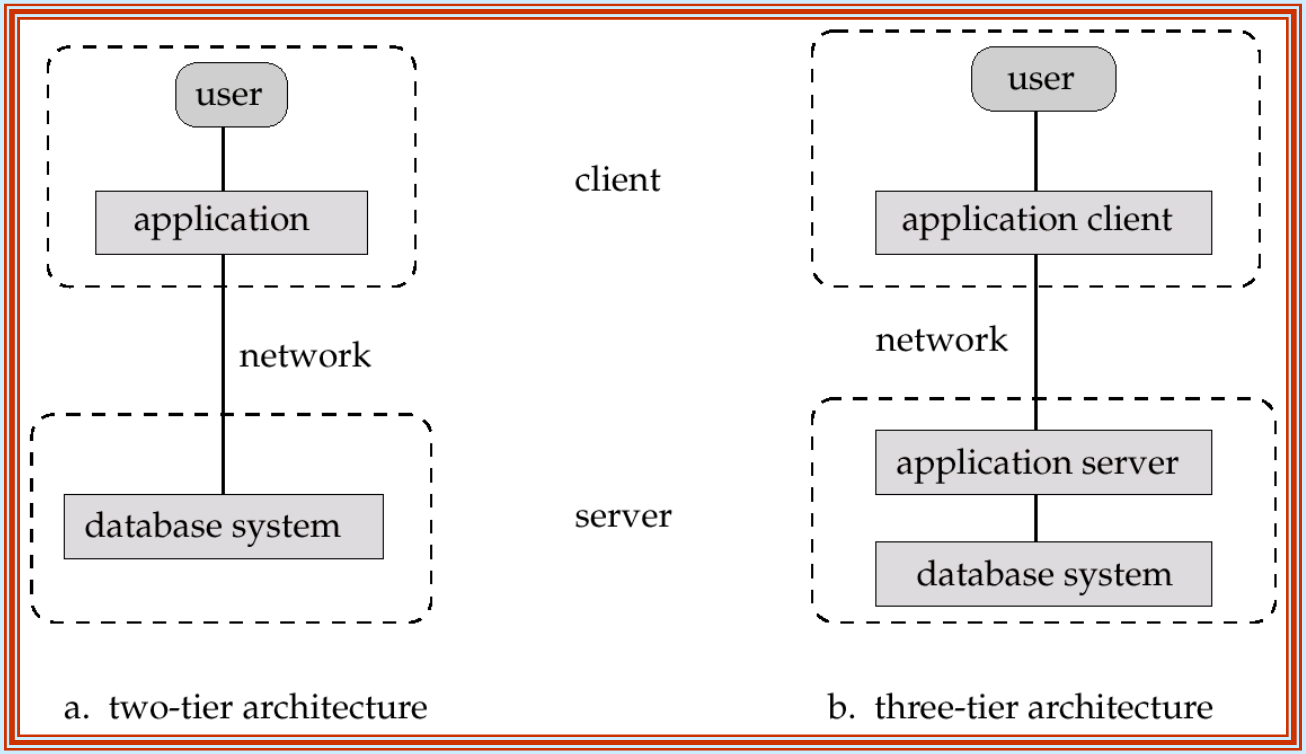
Transaction Management

- What if
  - The system fails?
  - More than one user is concurrently updating the database
- A transaction is a collection of operations that performs a single logical function in a database application
  - Atomicity: All the operations must happen in entirety or not at all
  - Consistency: preserve the consistency of the database
  - Isolation: Does not affect the execution of other transaction
  - Durability: After the successful execution, all the modification to database must be persist.
- 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

- Centralized databases
  - One to a few cores, shared memory
- Parallel databases
  - Many core shared memory
  - Shared disk
  - Shared nothing
- Distributed databases
  - Geographical distribution
  - Schema/data heterogeneity

Application Architectures



- **Two-tier architecture**: E.g. client programs using ODBC/JDBC to communicate with a database
- **Three-tier architecture**: E.g. web-based applications, and applications built using "middleware"

Database Users

Users are differentiated by the way they expect to interact with the system

- **Naïve users** – invoke one of the permanent application programs that have been written previously
  - Examples, people accessing database over the web, bank tellers, students
- **Application programmers** – computer professionals who write application programs to interact with database system
- ***Sophisticated users*** *– interact with database system without writing programs, using tools or query language to access system.*
- ***Specialized users*** *– write specialized database applications that do not fit into the traditional data processing framework. For example, CAD system, Expert system, etc.*
- **Database Administrator** – a powerful user has the central control over the database system.

Database Administrator

- Have central control to the database system.
- The database administrator has a good understanding of the enterprise’s information resources and needs.
- Database administrator's duties include:
  - Schema definition
    - Specifying integrity constraints
  - Storage structure and access method definition
  - Schema and physical organization modification
  - Granting of authorization for data access
  - Routine maintenance
    - Periodically backing up the data base
    - Ensuring enough free disk space
    - Monitoring performance and responding to changes in requirements

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provide only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allow direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Codd 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
  - Initial Object-oriented database systems
- 1990s:
  - Parallel and distributed database systems
  - Large decision support and data-mining over large data warehouses
  - Emergence of Web commerce
    - Very high performance, reliability and availability

History (cont.)

- 2000s:
  - Semi-structured data became increasingly important, XML for data exchange among different applications.
  - Open-source database systems (PostgreSQL and MySQL) widely used.
  - Big data storage systems
    - "NoSQL" systems.
  - Big data analysis: beyond SQL
    - Map reduce and friends
- 2010s
  - SQL reloaded
    - SQL front end to Map Reduce systems
    - Massively parallel database systems
    - Multi-core main-memory databases
  - Cloud native systems

End of Chapter 1