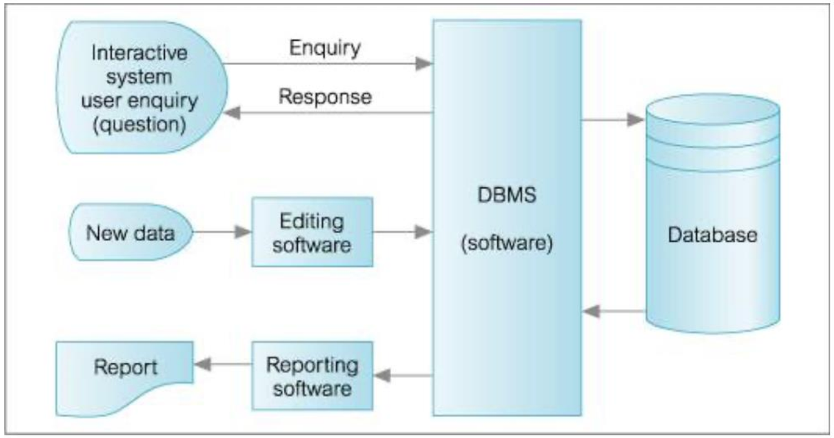
Software Engineering

Databases

Dr. Young-Woo Kwon

Basic Definitions

- **Data:**
  - Known facts that can be recorded and have an implicit meaning.
- **Database:**
  - A collection of related data.
- **Database Management System (DBMS):**
  - A software package/ system to facilitate the creation and maintenance of a computerized database.
- **Database System:**
  - **DB + DBMS (+ DB Applications)**



Types of Databases and Database Applications

- Traditional Applications:
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Biological and Genome Databases
  - Data Warehouses
  - Mobile databases
  - Real-time and Active Databases

Recent Developments (1)

- Advances in tech have led to exciting new applications of database systems.
  - Social media websites (e.g., Facebook, Twitter, LinkedIn ...) require storing *nontraditional* data (e.g. posts, tweets, images, and video clips)
    - Resulted in the emergence of new types of database systems, referred as **big data storage systems**, or **NoSQL** (Not-only SQL) systems (Hive, Pig, HBase, etc.) to manage such data for social media applications
  - Google, Amazon, Yahoo, eBay, etc., also used these types of systems to manage the data required in their Web search engines.
    - They collected their own repository of web pages for search.
    - In particular, Google Cloud and Amazon AWS provided cloud storage.

Recent Developments (2)

- NoSQL is a term used for a broad group of data management technologies varying in features and functionality
  - A SQL database is concrete concept, but NoSQL is NOT.
- Features
  - High performance writes and massive scalability: E.g., mongoDB, elasticsearch, Cassandra
  - Do not require a defined scheme for writing data
  - Primarily eventually-consistent by default
  - Support of wide range of modern programming languages (Python, Scala, Go, ...) and tools
  - Support of fault tolerance; typically, distributed computing

KNU

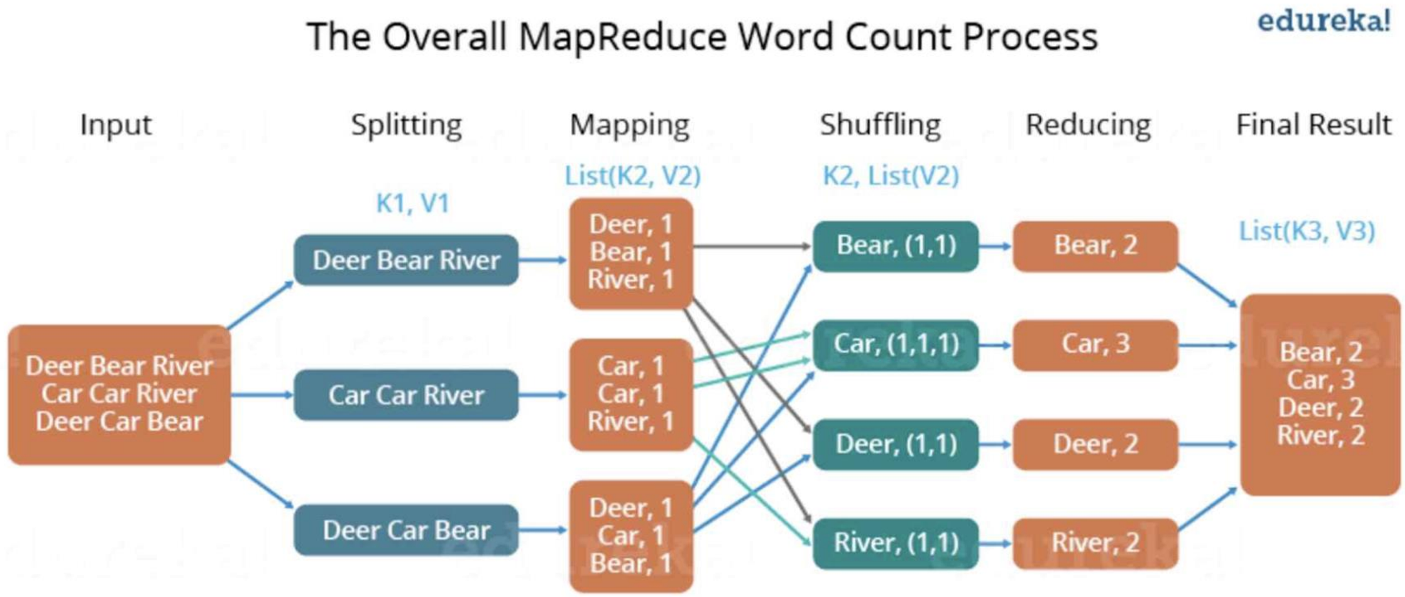
Recent Developments (3)

- MongoDB

```
{
"_id": ObjectId("5ad88534e3632e1a35a58d00"),
"name": {
"first": "John",
"last": "Doe" },
"address": [
{ "location": "work",
         "address": {
         "street": "16 Hatfields",
         "city": "London",
         "postal_code": "SE1 8DJ"},
         "geo": { "type": "Point", "coord": [
                  51.5065752,-0.109081]}},
],
"phone": [
         { "location": "work",
           "number": "+44-1234567890"},
],
"dob": ISODate("1977-04-01T05:00:00Z"),
"retirement_fund": NumberDecimal("1292815.75")
}
```

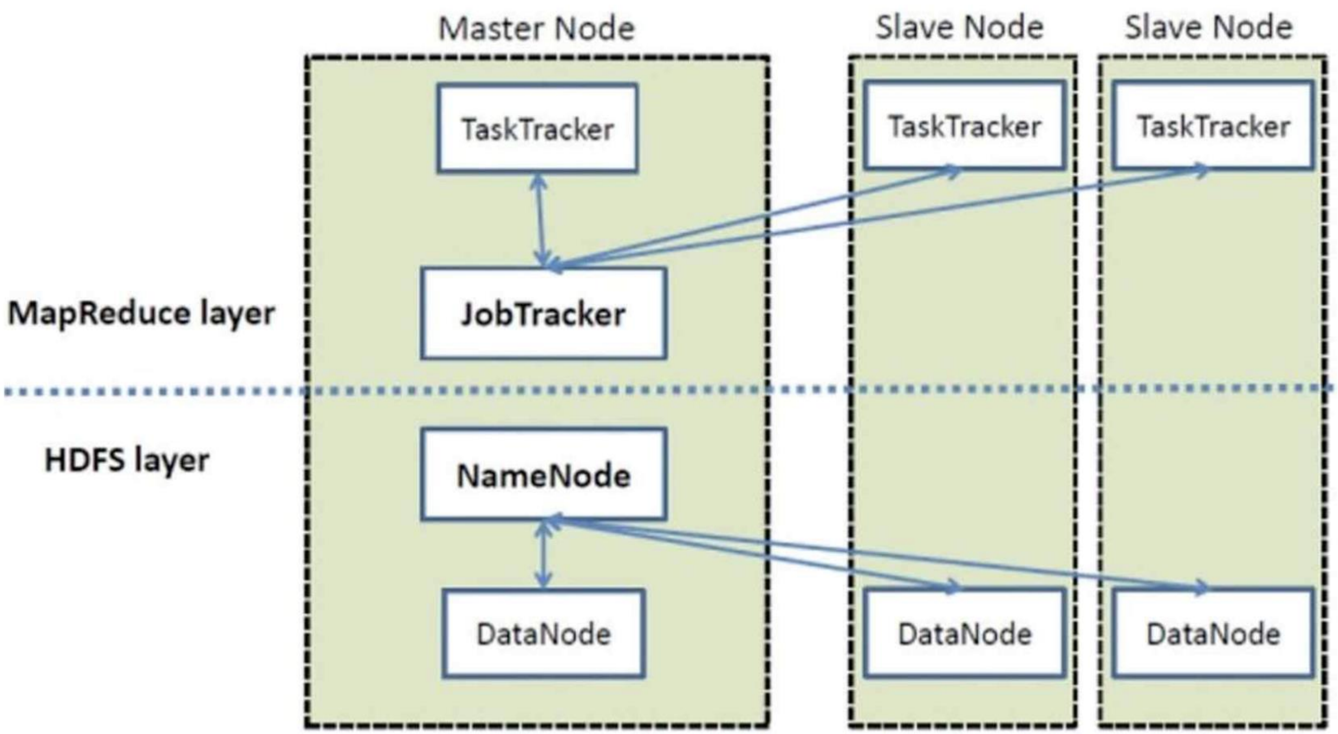
Recent Developments (4)

- Hadoop
  - Actual Implementation of *MapReduce* Programming Model for Big Data Processing
  - Map Reduce Example:



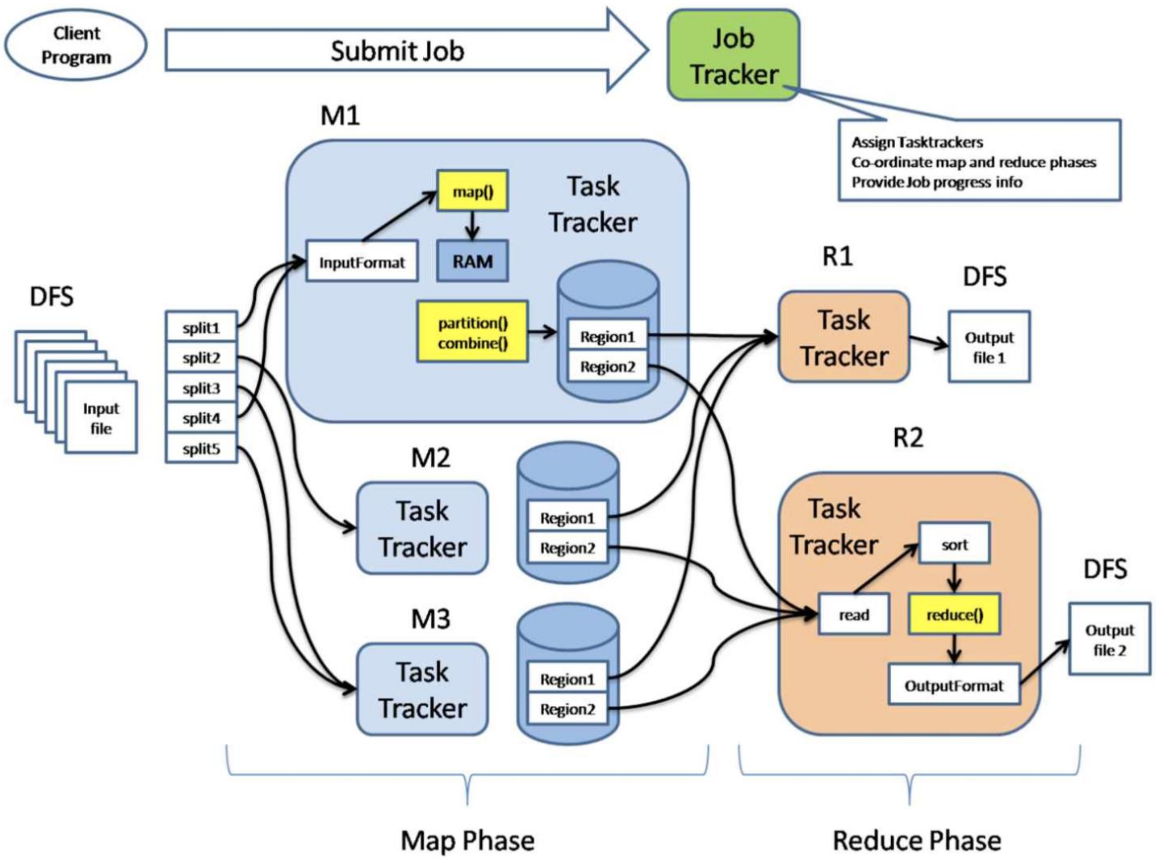
Recent Developments (5)

- Hadoop
  - High-level Architecture



Recent Developments (5)

- Hadoop
  - Detailed Architecture

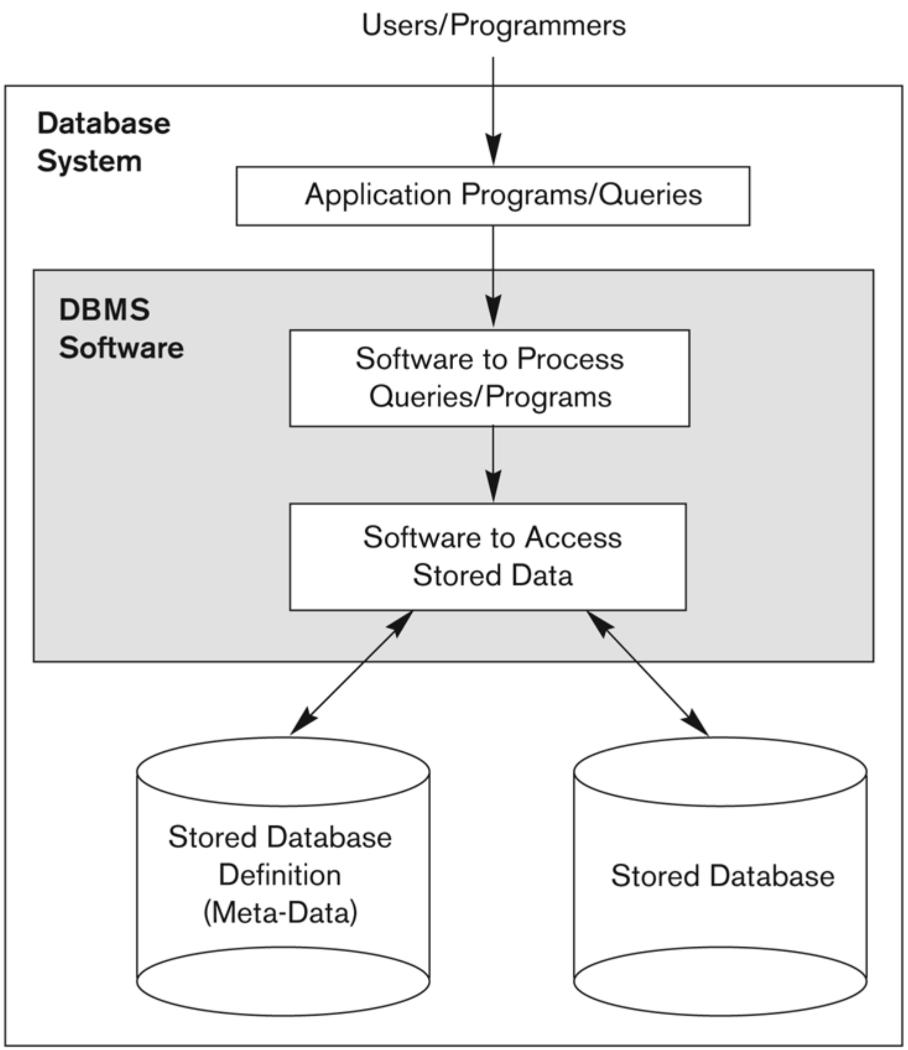


Impact of Databases and Database Technology

- Businesses: Banking, Insurance, Retail, Transportation, Healthcare, Manufacturing
- Service Industries: Financial, Real-estate, Legal, Electronic Commerce, Small businesses
- Education : Resources for content and Delivery
- More recently: Social Networks, Environmental and Scientific Applications, Medicine and Genetics
- Personalized Applications: based on smart mobile devices

DATABASE SYSTEM ENVIRONMENT

Simplified database system environment



1) Applications interact with a DB by generating:

- **Query**: typically causes some data to be retrieved
- **Transaction**: may cause some data to be read and/or to be written into the database **atomically**

2) Applications must not allow unauthorized users to access data: provide **data protection**.

3) Applications must keep up with changing user requirements against the DB; ex) password policy change, authorization change

Typical DBMS Functionality

- *Define* a particular database in terms of its data types, structures, and constraints
- *Construct* or Load the initial database contents on a secondary storage medium
- *Manipulating* the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- *Processing* and *Sharing* by a set of concurrent users and application programs – yet, keeping all data valid and consistent

Example of a Database (with a Conceptual Data Model)

- **Mini-world for the example:**
  - Part of a UNIVERSITY environment.
- **Some mini-world *entities*:**
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

Example of a Database (with a Conceptual Data Model)

- **Some mini-world *relationships*:**
  - SECTIONs *are of specific* COURSEs
  - STUDENTs *take* SECTIONs
  - COURSEs *have prerequisite* COURSEs
  - INSTRUCTORs *teach* SECTIONs
  - COURSEs *are offered by* DEPARTMENTs
  - STUDENTs *major in* DEPARTMENTs

- Note: The above entities and relationships are typically expressed in a conceptual data model, such as the ENTITY-RELATIONSHIP data model (see Chapters 3, 4)

Example of a simple database

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

MAIN CHARACTERISTICS OF THE DATABASE APPROACH

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

Example of a simplified database catalog

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Main Characteristics of the Database Approach (2)

- **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
- **Support of multiple views of the data:**
  - Each user may see a different view of the database, which describes **only** the data of interest to that user.

Main Characteristics of the Database Approach (3)

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS guarantees that each **transaction** is correctly executed or aborted
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database
  - **OLTP** (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.

KNU

Advantages of Using the Database Approach

- Controlling redundancy in data storage and in development and maintenance efforts.
  - Sharing of data among multiple users.
- Restricting unauthorized access to data. Only the DBA staff uses privileged commands and facilities.
- Providing persistent storage for program Objects
  - E.g., Object-oriented DBMSs make program objects persistent– see Chapter 12.
- Providing Storage Structures (e.g. indexes) for efficient Query Processing – see Chapter 17.

Advantages of Using the Database Approach (continued)

- Providing optimization of queries for efficient processing.
- Providing backup and recovery services.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database.
- Drawing inferences and actions from the stored data using deductive and active rules and triggers.

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).

KNU

DATABASE DESIGN

Overview of Database Design Process

- Two main activities:
  - Database design
    - Focuses on <u>conceptual database design</u>
    - To design the conceptual schema for a database application
  - Applications design
    - focuses on the programs and interfaces that access the database
    - Generally considered part of software engineering

Overview of Database Design Process

- Step 1: requirements collection and analysis
  - Functional / Data requirements
- Step 2: conceptual design
  - Conceptual schema
- Step 3:logical design (or data model mapping)
  - Logical schema
- Step 4: physical design
  - Internal schema: internal storage structures, file organizations, indexes, access paths, and parameters for database files

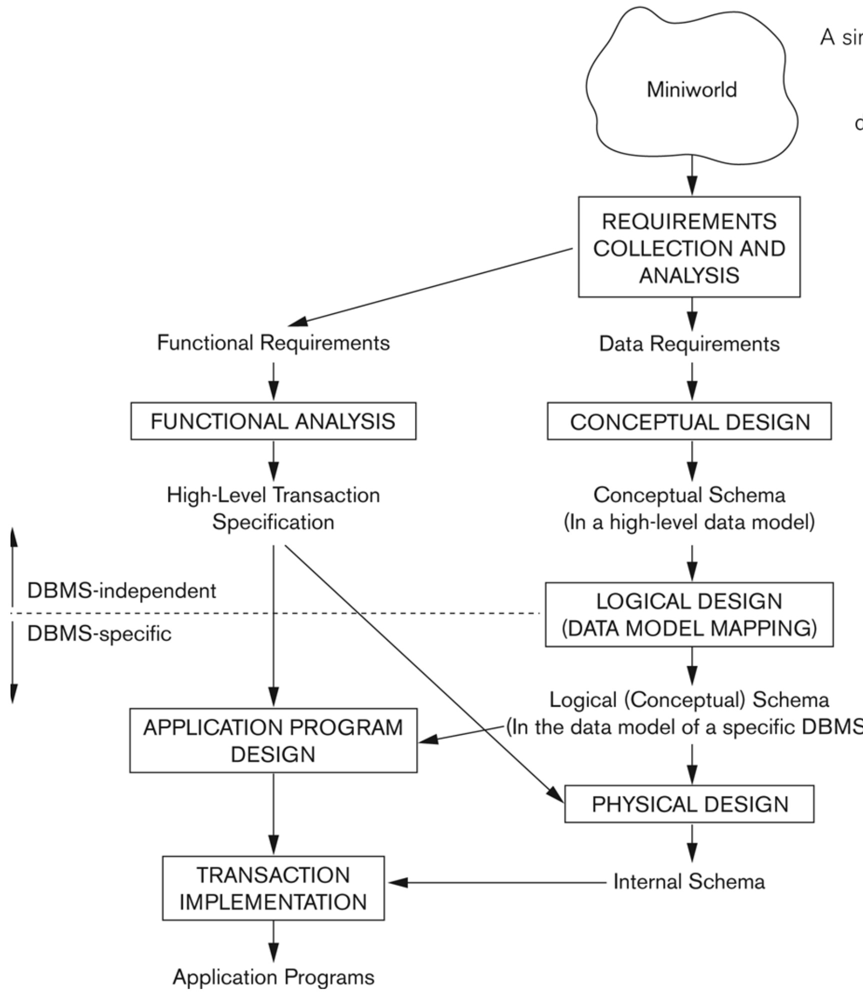


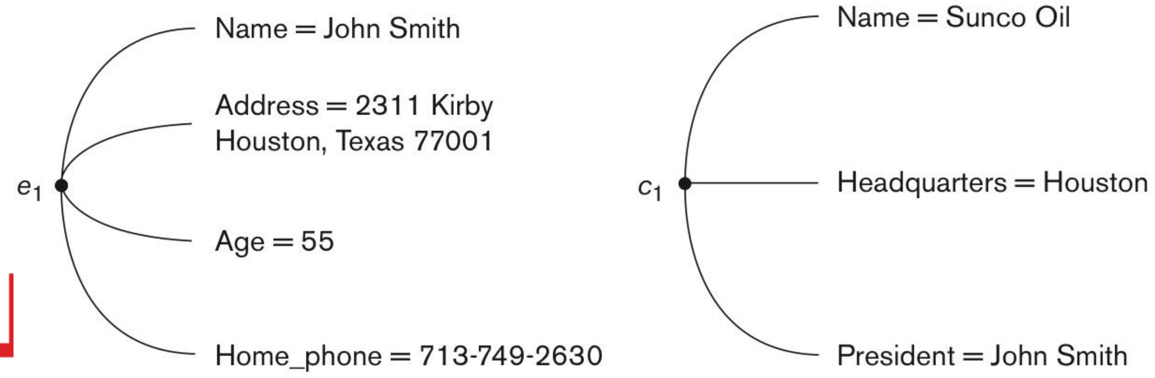
Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Methodologies for Conceptual Design

- Entity Relationship (ER) Diagrams
- Enhanced Entity Relationship (EER) Diagrams
- Use of Design Tools in industry for designing and documenting large scale designs
- The UML (Unified Modeling Language) Class Diagrams are popular in industry to document conceptual database designs

ER Model Concepts

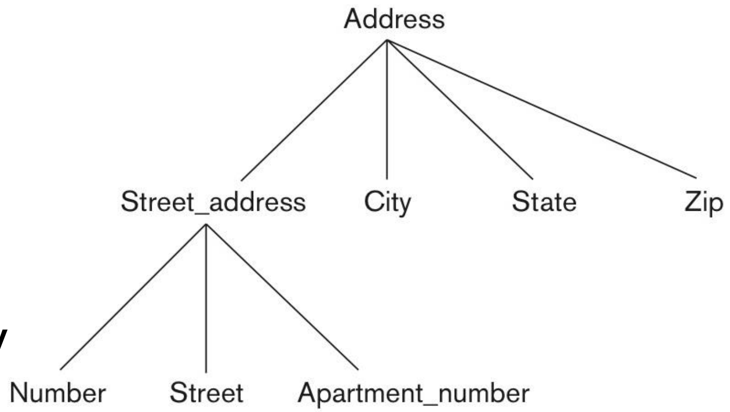
- Entities
  - Entity is a basic concept for the ER model.
  - Entities are specific things or objects in the mini-world that are represented in the database.
- Attributes
  - Attributes are properties used to describe an entity.
  - A specific entity will have a value for each of its attributes.
  - Each attribute has a value set (or data type) associated with it – e.g. integer, string, date, enumerated type, …



KNU

Types of Attributes (1)

- Simple
  - Each entity has a single atomic value for the attribute. For example, SSN or Sex.
- Composite
  - The attribute may be composed of several components. For example:
    - Address(Apt#, House#, Street, City, State, ZipCode, Country), or
    - Name(FirstName, MiddleName, LastName).
    - Composition may form a hierarchy where some components are themselves composite.



- Multi-valued
  - An entity may have multiple values for that attribute.
    - For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.

Types of Attributes (2)

- Composite and multi-valued attributes may be nested arbitrarily to any number of levels (very rare)
  - PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}
  - Multiple PreviousDegrees values can exist
  - Each has four subcomponent attributes:
    - College, Year, Degree, Field

Example COMPANY Database

- We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:
  - The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager. A department may have several locations.
  - Each department *controls* a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.

Example COMPANY Database

- The database will store each EMPLOYEE's social security number, address, salary, sex, and birthdate.
  - Each employee *works for* one department but may *work on* several projects.
  - The DB will keep track of the number of hours per week that an employee currently works on each project.
  - It is required to keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTs.
  - For each dependent, the DB keeps a record of name, sex, birthdate, and relationship to the employee.

Initial Conceptual Design of Entity Types for the COMPANY Database Schema

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT
- Their initial conceptual design is shown on the following slide
- The initial attributes shown are derived from the requirements description

Initial Design of Entity Types:

EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

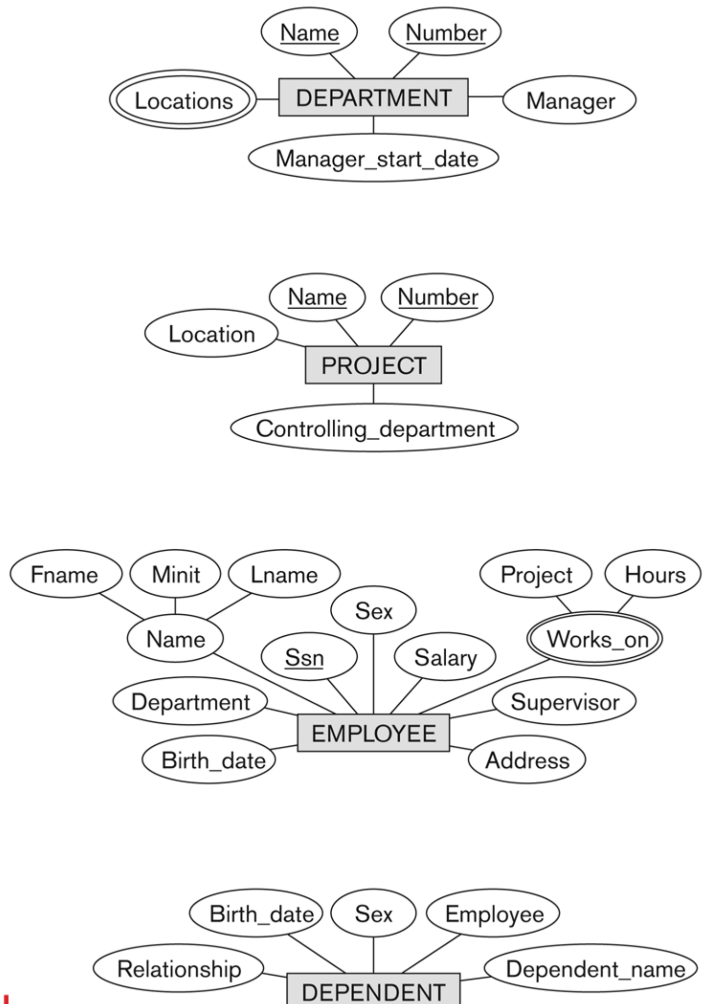
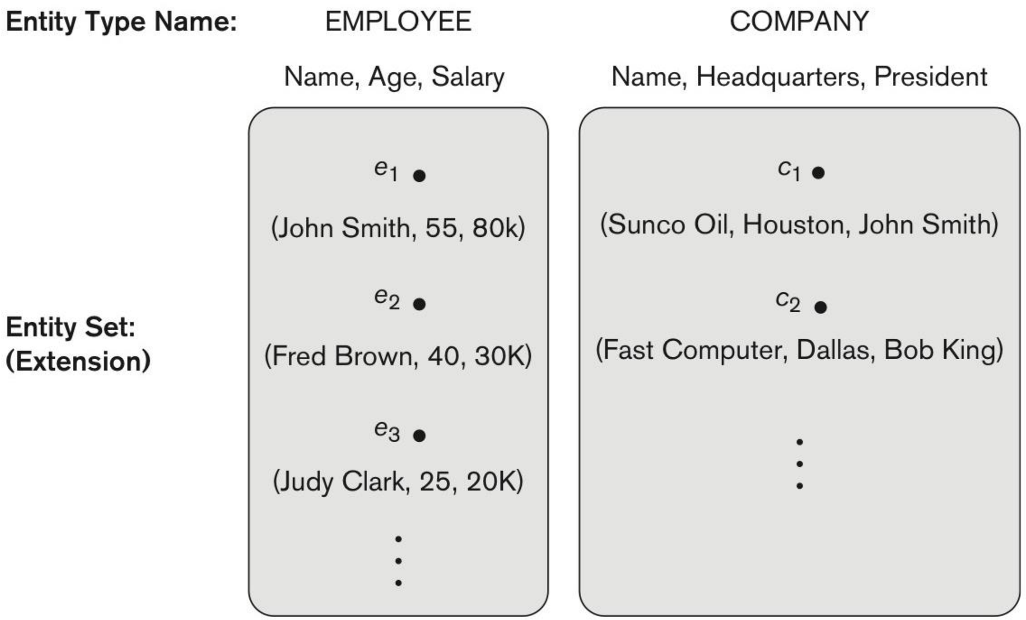


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Entity Types and Key Attributes (1)

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the entity type EMPLOYEE and PROJECT.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.
  - For example, SSN of EMPLOYEE.



Entity Types and Key Attributes (2)

- A key attribute may be composite.
  - VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
  - The CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN)
    - VehicleTagNumber (Number, State), aka license plate number.
- <u>Each key</u> is <u>underlined</u> (Note: this is different from the relational schema where only one "primary key is underlined).

Entity Set

- Each entity type will have a collection of entities stored in the database
  - Called the **entity set** or sometimes **entity collection**
- Previous slide shows three CAR entity instances in the entity set for CAR
- Same name (CAR) used to refer to both the entity type and the entity set
- However, entity type and entity set may be given different names
- Entity set is the current *state* of the entities of that type that are stored in the database

Value Sets (Domains) of Attributes

- Each simple attribute is associated with a value set
  - E.g., Lastname has a value which is a character string of up to 15 characters, say
  - Date has a value consisting of MM-DD-YYYY where each letter is an integer
- A **value set** specifies the set of values associated with an attribute
- Value sets are similar to the basic data types available in most programming languages: e.g. integer, string, double, ...

Mathematical Expression of a Value Set

- An attribute A (e.g., age) for an entity type E (e.g., PERSON) whose
- value set is V (e.g., {10,...,99}) is defined as a function

$$A : E \rightarrow P(V)$$

where P(V) indicates a power set (which means all possible subsets) of V.

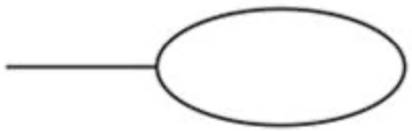
- *A*(*e*): the value of attribute *A* for entity *e*.
- The above definition covers single-valued and multivalued attributes as well.
- What about a composite attribute *A*? (E.g. birth_date(yr, mon,day))

$$V = P(P(V1) \times P(V2) \times \ldots \times P(Vn))$$

Where *V1,V2 , ... , Vn* are the value sets of the simple component attributes that form *A*.

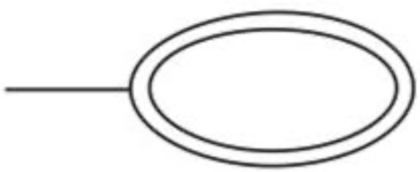
NOTATION for ER diagrams

Entity

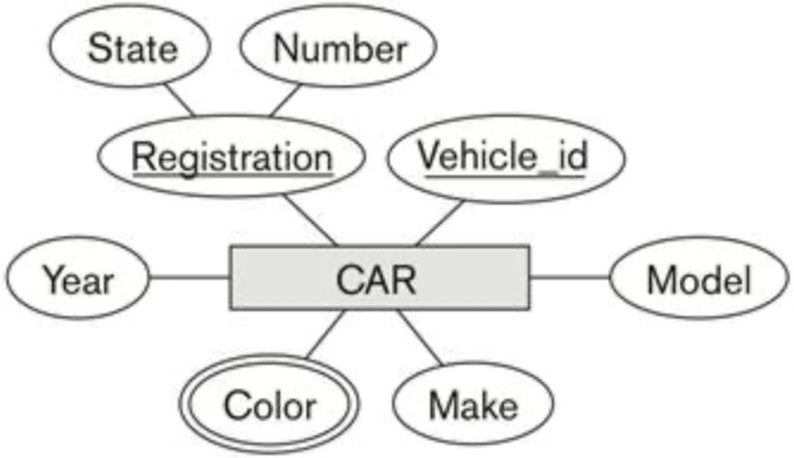


Attribute

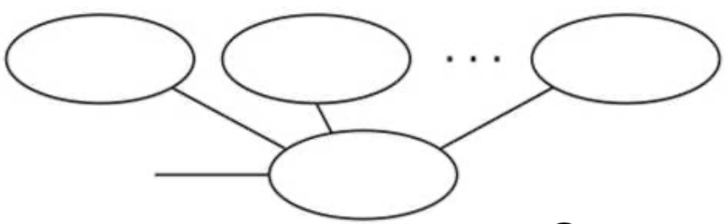
Key attribute



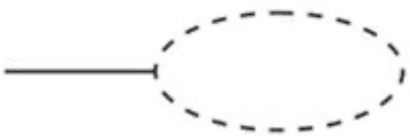
Multivalued Attribute



[ER diagram notation of the CAR entity type]



Composite attribute



Derived attribute

KNU

DBMS Interfaces

- Stand-alone query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL Shell in PostgreSQL)
- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
  - Menu-based, forms-based, graphics-based, etc.
- Mobile Interfaces: interfaces allowing users to perform transactions using mobile apps

DBMS Programming Language Interfaces

- Programmer interfaces for embedding DML in a programming languages:
  - **Embedded Approach**: e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)
  - **Procedure Call Approach**: e.g. JDBC for Java, ODBC (Open Databse Connectivity) for other programming languages as API's (application programming interfaces)
  - **Database Programming Language Approach**: e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components
  - **Scripting Languages:** PHP (client-side scripting) and Python (server-side scripting) are used to write database programs.

User-Friendly DBMS Interfaces

- Menu-based (Web-based), popular for browsing on the web
- Forms-based, designed for naïve users used to filling in entries on a form
- Graphics-based
  - Point and Click, Drag and Drop, etc.
  - Specifying a query on a schema diagram
- Natural language: requests in written English
- Combinations of the above:
  - For example, both menus and forms used extensively in Web database interfaces

Other DBMS Interfaces

- Natural language: free text as a query
- Speech : Input query and Output response
- Web Browser with keyword search
- Parametric interfaces, e.g., bank tellers using function keys.
- Interfaces for the DBA:
  - Creating user accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access paths

THE DATABASE SYSTEM ENVIRONMENT

KNU

DBMS Component Modules

- Some things usually stored on disk
  - The stored database
  - The DBMS catalog
- Access to the disk is controlled by OS, scheduling disk read/write.
- Many DBMSs have their own buffer manager to schedule disk read/write.
  - More specifically, management of buffer storage has a considerable effect; *reducing* disk I/O improves performance considerably.
- A high-level stored data manager (module) of the DBMS controls access to DBMS information stored on disk
  - Whether it's part of the database or the catalog.

Typical DBMS Component Modules

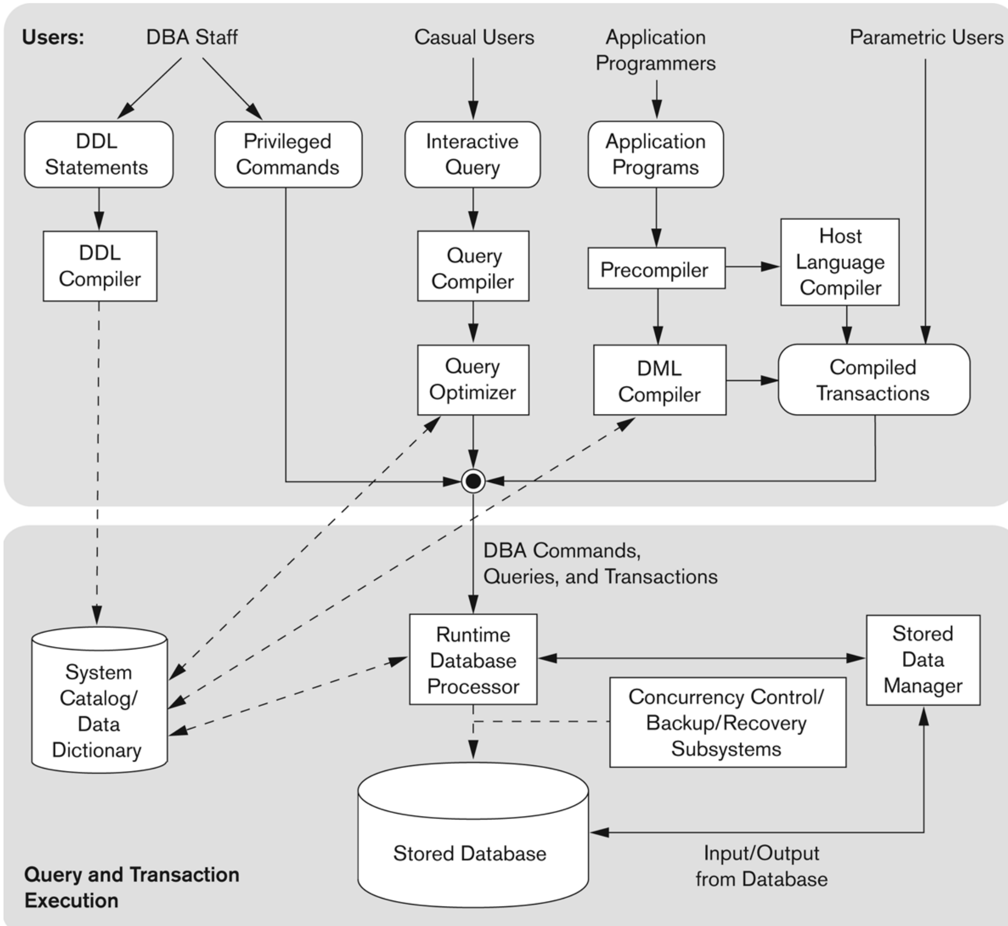
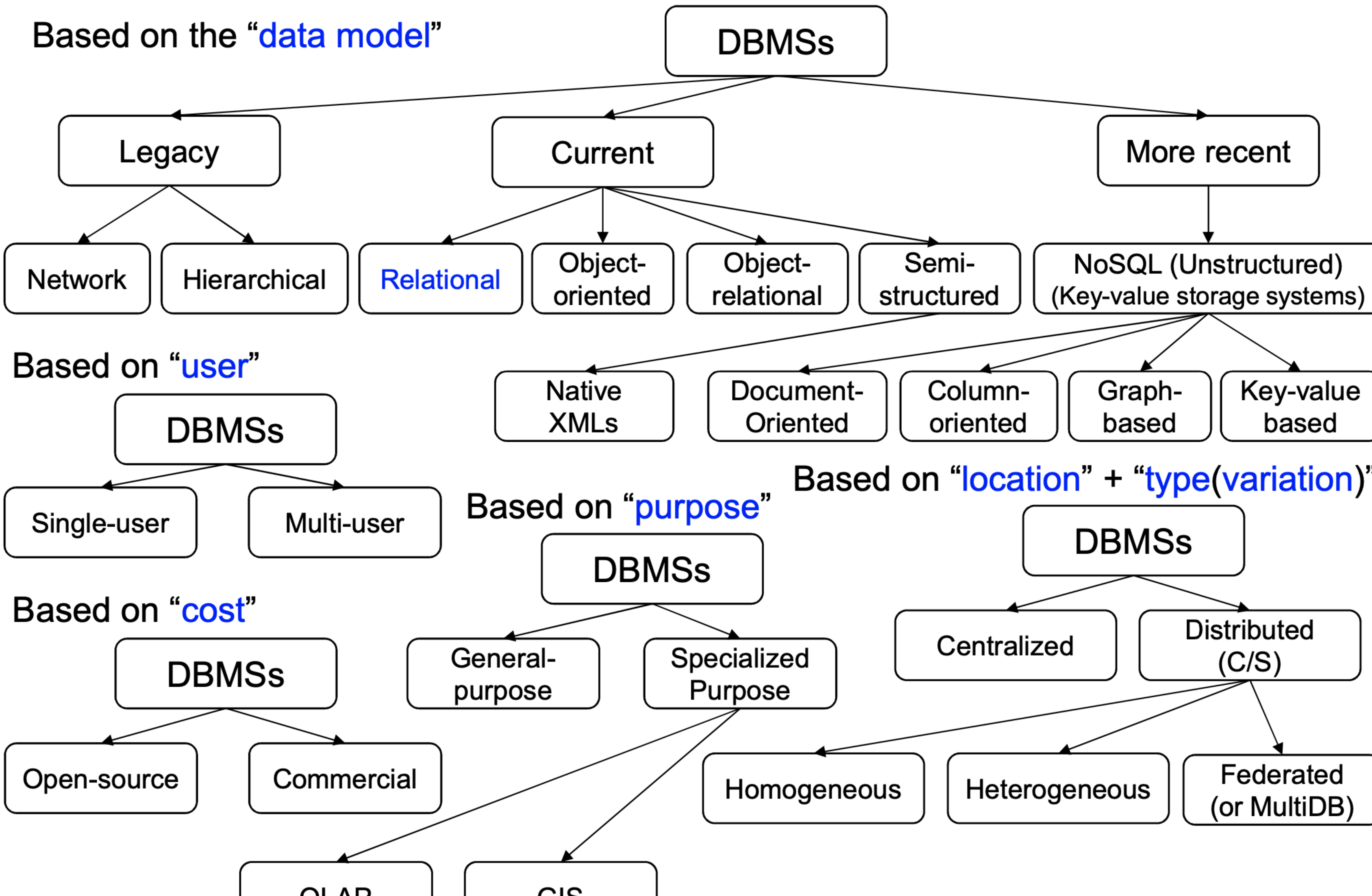


Figure 2.3
Component modules of a DBMS and their interactions.

CLASSIFICATION OF DATABASE MANAGEMENT SYSTEMS



Additionally, disk-based vs. in-memory DB

Classification of DBMSs

- Based on the data model used
  - Legacy: Network, Hierarchical.
  - Currently Used: Relational, Object-oriented, Object-relational
  - Recent Technologies: Key-value storage systems, NOSQL systems: document based, column-based, graph-based and key-value based. Native XML DBMSs.
- Other classifications
  - Single-user (typically used with personal computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (multiple computers, multiple DBs)

Centralized and Client-Server DBMS Architectures

- Centralized DBMS:
  - Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
  - User can still connect through a remote terminal – however, all processing is done at centralized site.

Centralized Architecture

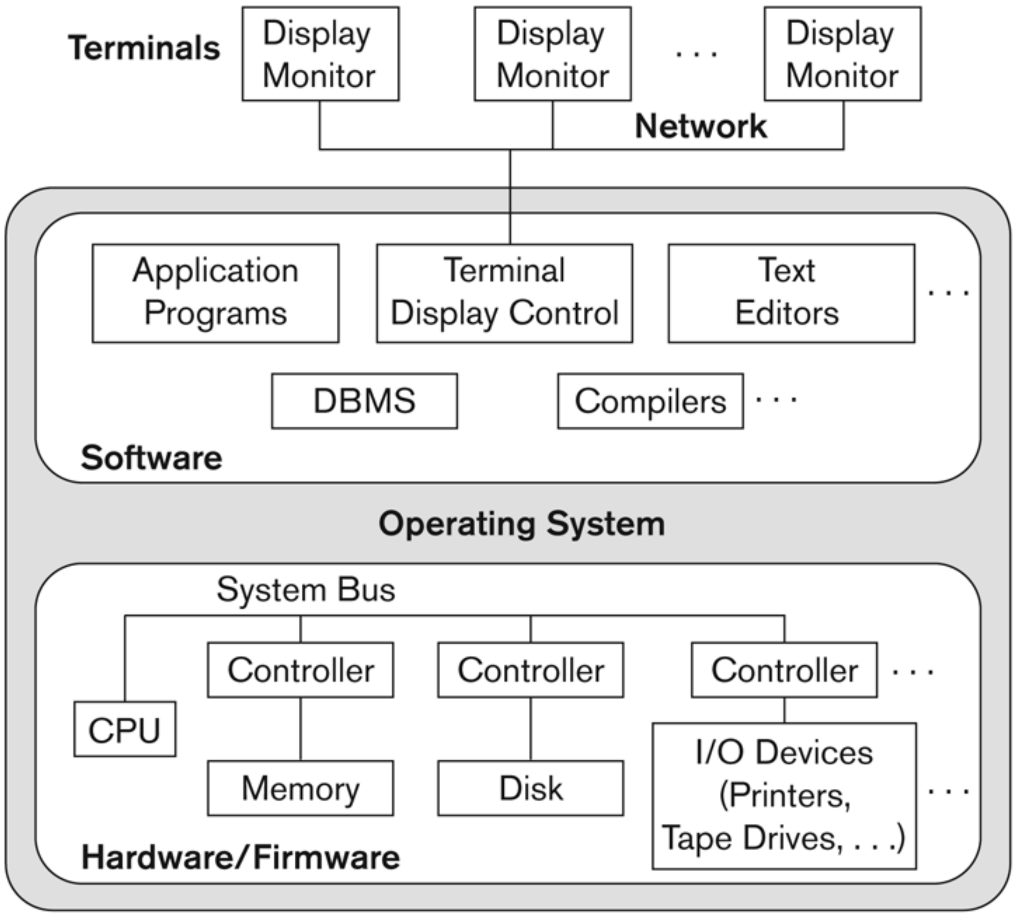
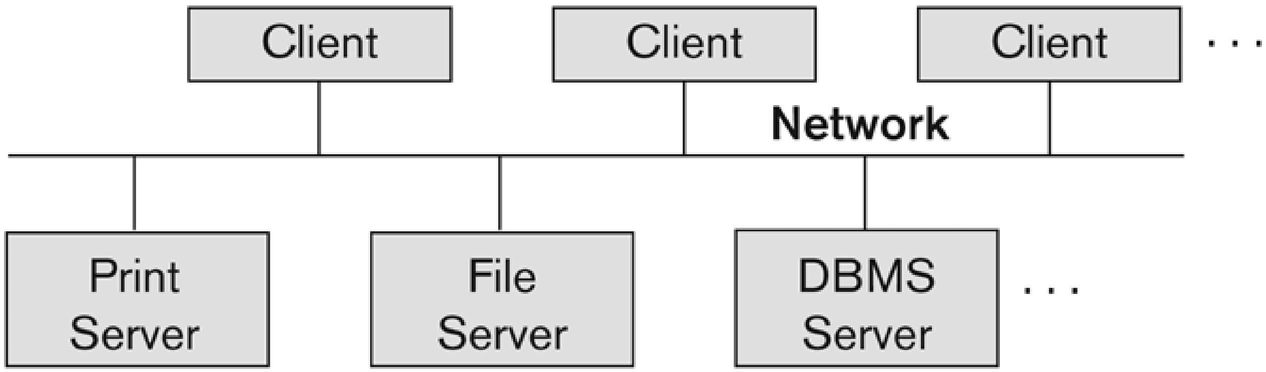


Figure 2.4
A physical centralized architecture.

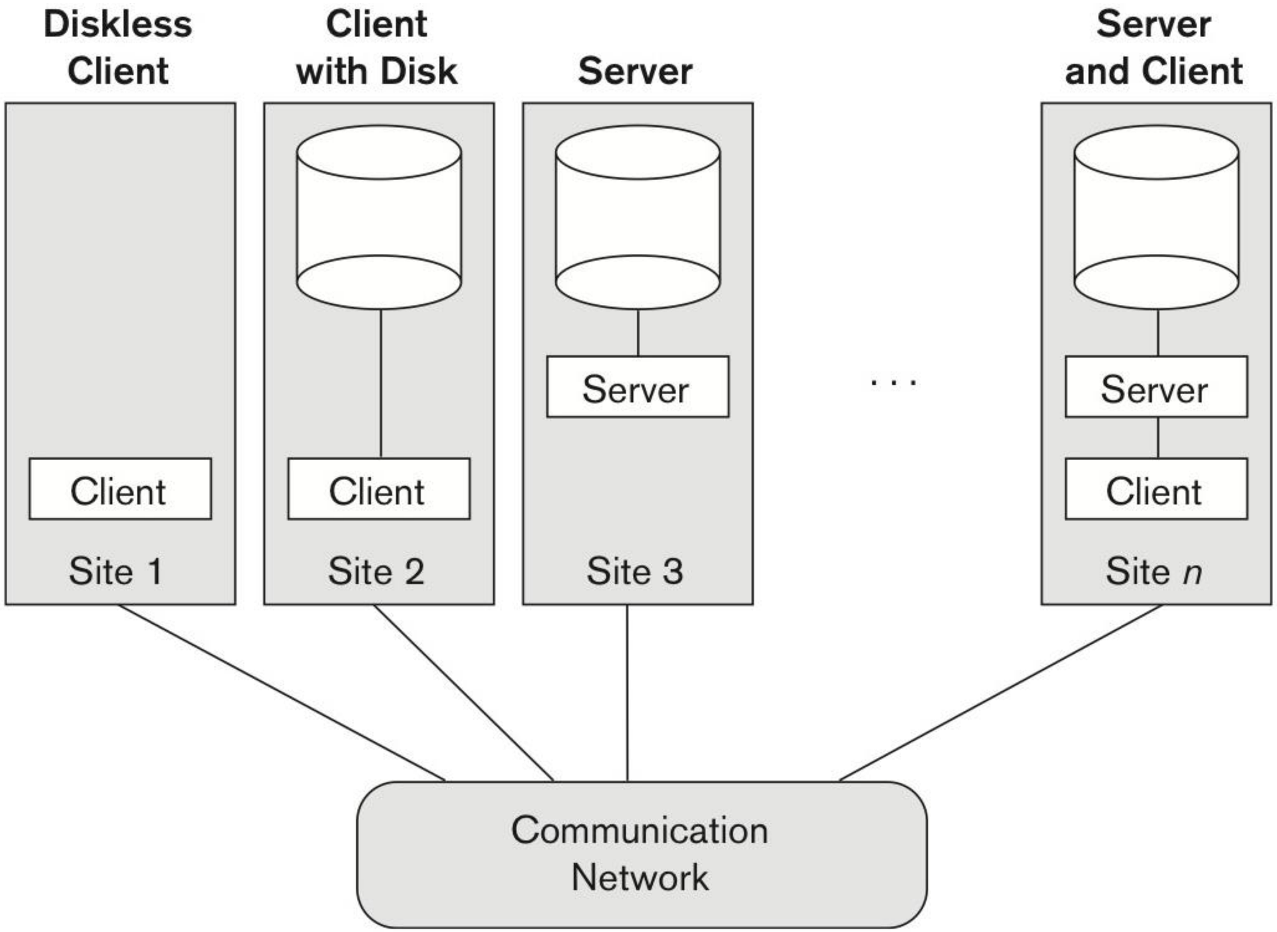
Basic 2-tier Client-Server Architectures

- Specialized Servers with specialized functions
  - Print server, file server, DBMS server, Web server, Email server
- Clients can access the specialized servers as needed

Figure 2.5
Logical two-tier client/server architecture.



Basic 2-tier Client-Server Architectures



Basic 2-tier Client-Server Architectures

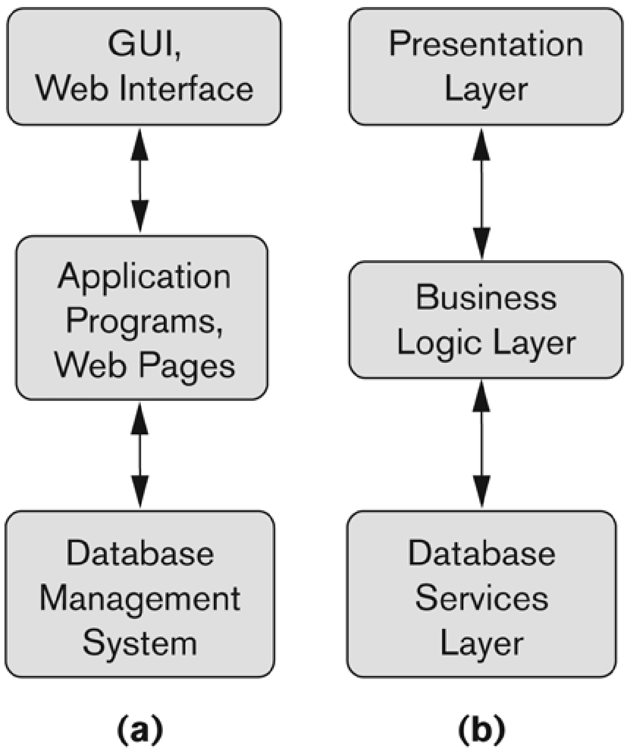
- Client
  - Provide appropriate interfaces through a client software module to access and utilize the various server resources.
  - Clients may be diskless machines or PCs or Workstations with disks with only the client software installed.
  - Connected to the servers via some form of a network.
    - (LAN: local area network, wireless network, etc.)
- DBMS Server
  - Provides database query and transaction services to the clients
  - Relational DBMS servers are often called SQL servers, query servers, or transaction servers
  - Applications running on clients utilize an Application Program Interface (API) to access server databases via standard interface such as:
    - ODBC: Open Database Connectivity standard
    - JDBC: for Java programming access

Variations of Distributed DBMSs (DDBMSs)

- Homogeneous DDBMS
- Heterogeneous DDBMS
- Federated or Multidatabase Systems
  - Participating Databases are loosely coupled with high degree of autonomy.
- Distributed Database Systems have now come to be known as client-server based database systems because:
  - They do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.

Three Tier Client-Server Architecture

- Common for Web applications
- Intermediate Layer called Application Server or Web Server:
  - Stores the web connectivity software and the business logic
  - Sends partially processed data between the database server and the client.
- Three-tier Architecture can enhance security



Cost considerations for DBMSs

- Cost Range: from free open-source systems to configurations costing millions of dollars
  - Examples of free relational DBMSs: MySQL, PostgreSQL, others
  - Commercial DBMS offer additional specialized modules, e.g. time-series module, spatial data module, document module, XML module
    - These offer additional specialized functionality when purchased separately
    - Sometimes called cartridges (e.g., in Oracle) or blades
- Different licensing options: site license, maximum number of concurrent users (seat license), single user, etc.

Other Considerations

- Type of access paths within database system
  - E.g.- inverted indexing based (ADABAS is one such system).Fully indexed databases provide access by any keyword (used in search engines)
- General Purpose vs. Special Purpose
  - E.g.- Airline Reservation systems or many others- reservation systems for hotel/car etc. Are special purpose OLTP (Online Transaction Processing Systems)