Database Systems

Session 1 – Main Theme
Introduction to Database Systems
Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Presentation material partially based on textbook slides
Fundamentals of Database Systems (7th Edition)
by Ramez Elmasri and Shamkant Navathe
Slides copyright © 2022

Agenda

Who am I?

- Profile -

- 39 years of experience in the Information Technology Industry, including 24 years of experience working for leading business technology consulting firms such as Computer Sciences Corporation
- CEO, Archemy Inc. and Professor of Computer Science at New York University
- PhD in Computer Science from University of Colorado at Boulder (UCB)
- Held senior management and technical leadership (SVP, CTO, CIO) roles in many large business technology strategy and modernization projects for fortune 500 corporations in the insurance, banking, investment banking, pharmaceutical, retail, and information management industries
- Contributed to several high-profile ARPA and NSF research projects
- Played an active role as a member of the OMG, ODMG, and X3H2 standards committees and as a Professor of Computer Science at Columbia initially and New York University since 1997
- Proven record of delivering business solutions on time and on budget
- Original designer and developer of jcrew.com and the suite of products now known as IBM InfoSphere DataStage
- Creator of the Enterprise Architecture Management Framework (EAMF) and main contributor to the creation of various maturity assessment methodologies
- Developed partnerships between several companies and New York University to incubate new methodologies (e.g., EA maturity assessment methodology developed in Fall 2008), develop proof of concept software, recruit skilled graduates, and increase the companies' visibility

How to reach me?

	Cell	(212) 203-5004
	Email	jcf@cs.nyu.edu, jcf@archemy.com
	Facebook	Jean-Claude Franchitti
	LinkedIn	http://www.linkedin.com/in/jcfranchitti
	Twitter	http://twitter.com/jcfranchitti
	Skype	jcfranchitti
	WeChat	metacomp

What is the Course About?

- Course description and syllabus:
  - » http://www.nyu.edu/classes/jcf/CSCI-GA.2433-001
  - » http://cs.nyu.edu/courses/fall22/CSCI-GA.2433-001/
- Textbooks:
  - » ***Fundamentals of Database Systems (7th Edition)***

Ramez Elmasri and Shamkant Navathe

Pearson

ISBN-10: 0133970779, ISBN-13: 978-0133970777 - 7th Edition (06/18/15)

Icons / Metaphors

Information

Common Realization

Knowledge/Competency Pattern

Governance

Alignment

Solution Approach

Course Objectives

- Gain understanding of fundamental concepts of state-of-the art databases (more precisely called: Database Management Systems)
- Get to know some of the tools used in the design and implementations of databases
- Know enough so that it is possible to read/skim a database system manual and
  - Start designing and implementing small databases
  - Start managing and interacting with existing small to large databases
- Experiment and practice with industry leading vendor solutions:
  - Quest's Erwin for design of relational database
  - Oracle Database, Microsoft SQL Server, IBM DB2, and other DB products for writing relational queries
- Latest data management trends including Big Data, NoSQL, Cloud storage, etc.

Key Material Covered (1/2)

- Methodology used for modeling a business application during the database design process, focusing on entity-relationship model and entity relationship diagrams
- Relational model and implementing an entity-relationship diagram
- Relational algebra (using SQL syntax)
- SQL as data manipulation language
- SQL as data definition language
- Refining a relational implementation, including the normalization process and the algorithms to achieve normalization
- Physical design of the database using various file organization and indexing techniques for efficient query processing

Key Material Covered (2/2)

- Concurrency Control
- Recovery
- Query execution
- Data warehouses and Data Lakes
- NoSQL DBMSs
- Distributed and Parallel Processing frameworks (e.g., Hadoop, Spark)
- Massively Parallel Processing databases (e.g., MariaDB)
- Cloud data services
- Additional topics may be covered, as time allows, these topics are covered in greater depth in other courses but PowerPoint presentations for them will still be provided
- The course material is partially derived from the textbook slides and material covered as part of the Database Systems course offered at NYU Courant in previous semesters

Software Requirements

- Software tools will be available from the Internet or from the course Web site under demos as a choice of freeware or commercial tools
  - » Database Modeling Tools
  - » Database Management Software Tools
  - » etc.
- References will be provided on the course Web site

Agenda

Section Outline

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Types of Database Users
- Advantages of Using the Database Approach
- Historical Development of Database Technology
- Extending Database Capabilities
- When Not to Use Databases

Types of Databases and Database Applications

- Traditional database applications
  - Store textual or numeric information
- More recent applications:
  - » Object Databases (ODMSs)
  - » Graph Databases
  - » Multimedia databases
    - Store images, audio clips, and video streams digitally
  - » Geographic information systems (GIS)
    - Store and analyze maps, weather data, and satellite images
  - » Biological and Genome Databases
  - » Data warehouses and online analytical processing (OLAP) systems
    - Extract and analyze useful business information from very large databases
    - Support decision making
  - » Mobile databases
  - » Real-time and active databases
    - Control industrial and manufacturing processes
- Business Intelligence Platforms
- Unstructured vs. Structured Databases
  - » Big Data

Recent Developments

- Social Networks started capturing a lot of information about people and about communications among people-posts, tweets, photos, videos in systems such as:
  - Facebook
  - Twitter
  - Linked-In
- All of the above constitutes data
- Search Engines (Google, Bing, Yahoo): collect their own repository of web pages for searching purposes
- New Technologies are emerging from the so-called non-database software vendors to manage vast amounts of data generated on the web:
- Big Data storage systems involving large clusters of distributed computers
- NoSQL (Not Only SQL) and NewSQL systems
- A large amount of data now resides on the “cloud” which means it is in huge data centers using thousands of machines.

NoSQL Database

- **Key-Value Store** – Has a Big Hash Table of keys & values
  - » e.g., Riak, Amazon S3 (Dynamo)
- **Document-based Store** – Stores documents made up of tagged elements
  - » e.g., CouchDB, MongoDB
- **Column-based Store** – Each storage block contains data from only one column
  - » e.g., HBase, Cassandra
- **Graph-based** – A network database that uses edges and nodes to represent and store data
  - » e.g., Neo4J

Many Historical "Paradigm Shifts" in the Database Field

- **"Paradigm Shifts - Joel Barker"**
  - https://www.kanopy.com/product/new-business-paradigms
- **From Managing the Storage of Data to Managing a Wealth of Information …**
  - i.e., Data Management vs. Computer/Data Science
  - See Session 1 Sub-Topic 1: "Data Science Roadmap"

Basic Definitions

- Database
  - Collection of related data
  - Logically coherent collection of data with inherent meaning
  - Built for a specific purpose
- Data
  - Known facts that can be recorded and have an implicit meaning
- Mini-world or Universe of Discourse (UoD)
  - Represents some aspect of the real world about which data is stored in a database (e.g., student grades and transcripts at a university)
- Example of a large commercial database
  - Amazon.com
- Database management system (DBMS)
  - » A software package/ system to facilitate the creation and maintenance of a computerized database
- Database system
  - The DBMS software together with the data itself. Sometimes, the applications are also included

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

Simplified Database System Environment

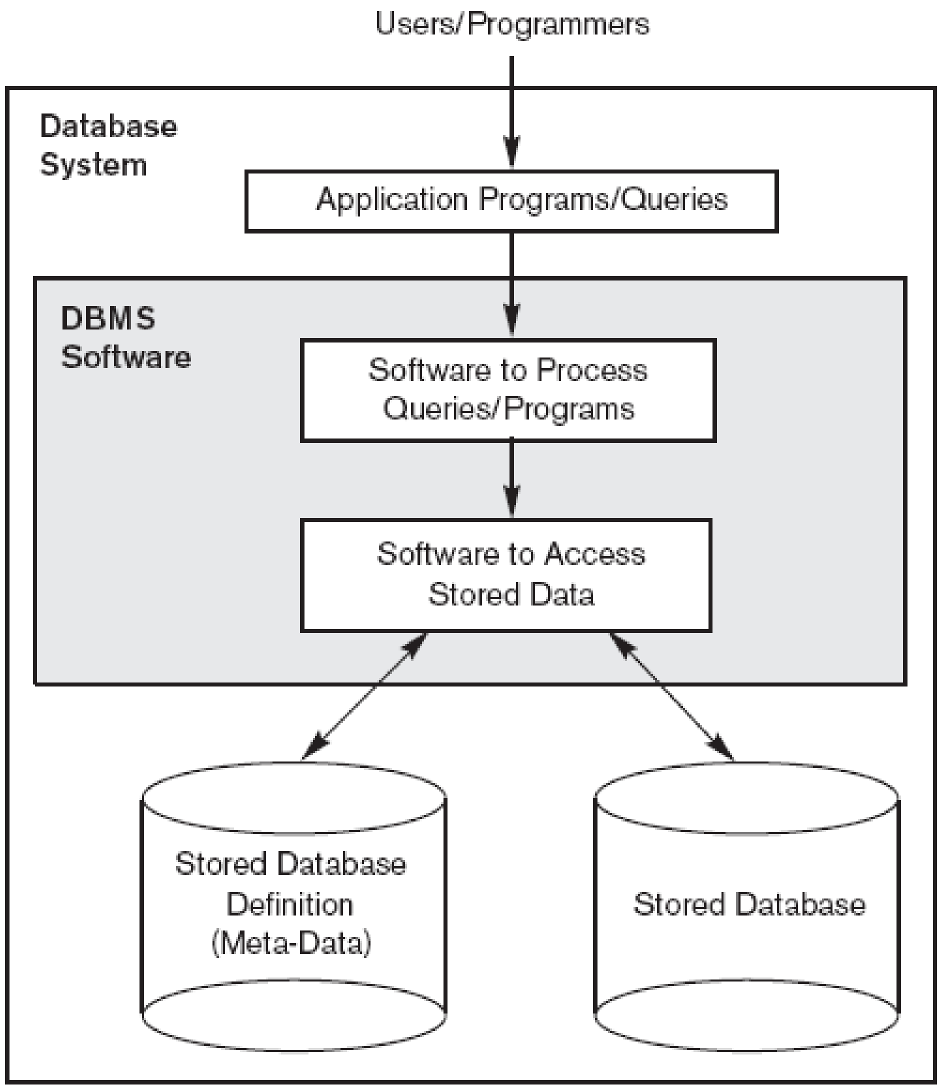

Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

Figure 1.1
A simplified database system environment.

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

Application Activities Against a Database

- Applications interact with a database by generating
  - Queries: that access different parts of data and formulate the result of a request
  - Transactions: that may read some data and “update” certain values or generate new data and store that in the database
- Applications must not allow unauthorized users to access data
- Applications must keep up with changing user requirements against the database

Additional DBMS Functionality

- DBMS may additionally provide:
  - Protection or Security measures to prevent unauthorized access
  - “Active” processing to take internal actions on data
  - Presentation and Visualization of data
  - Maintenance of the database and associated programs over the lifetime of the database application
    - Called database, software, and system maintenance

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

Example of a Simple Database

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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

- **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.
- **Isolation 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 (continued)

- **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 (continued)

- **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.

Database Users

- Users may be divided into
  - » Those who actually use and control the database content, and those who design, develop and maintain database applications (called “Actors on the Scene”), and
  - » Those who design and develop the DBMS software and related tools, and the computer systems operators (called “Workers Behind the Scene”).

Database Users – Actors on the Scene

- Actors on the scene
  - **Database administrators:**
    - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.
  - **Database Designers:**
    - Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

Database End Users

- Actors on the scene (continued)
  - » **End-users:** They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
    - **Casual**: access database occasionally when needed
    - **Naïve** or Parametric: they make up a large section of the end-user population.
      - They use previously well-defined functions in the form of “canned transactions” against the database.
        - » Users of Mobile Apps mostly fall in this category
      - Bank-tellers or reservation clerks are parametric users who do this activity for an entire shift of operations.
      - Social Media Users post and read information from websites

Database End Users (continued)

- **Sophisticated:**
  - These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
  - Many use tools in the form of software packages that work closely with the stored database.
- **Stand-alone:**
  - Mostly maintain personal databases using ready-to-use packaged applications.
  - An example is the user of a tax program that creates its own internal database.
  - Another example is a user that maintains a database of personal photos and videos.

Database Users – Actors on the Scene (continued)

- **System Analysts and Application Developers**

  This category currently accounts for a very large proportion of the IT work force.

  - **System Analysts**: They understand the user requirements of naïve and sophisticated users and design applications including canned transactions to meet those requirements.
  - **Application Programmers:** Implement the specifications developed by analysts and test and debug them before deployment.
  - **Business Analysts**: There is an increasing need for such people who can analyze vast amounts of business data and real-time data ("Big Data") for better decision making related to planning, advertising, marketing etc.

Database Users – Actors Behind the Scene

- **System Designers and Implementors:** Design and implement DBMS packages in the form of modules and interfaces and test and debug them. The DBMS must interface with applications, language compilers, operating system components, etc.
- **Tool Developers**: Design and implement software systems called tools for modeling and designing databases, performance monitoring, prototyping, test data generation, user interface creation, simulation etc. that facilitate building of applications and allow using database effectively.
- **Operators and Maintenance Personnel:** They manage the actual running and maintenance of the database system hardware and software environment

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
- Providing Storage Structures (e.g., indexes) for efficient Query Processing

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.

Additional Implications of Using the Database Approach

- Potential for enforcing standards:
  - This is very crucial for the success of database applications in large organizations.
  - **Standards** refer to data item names, display formats, screens, report structures, meta-data (description of data), Web page layouts, etc.
- Reduced application development time:
  - Incremental time to add each new application is reduced.

Additional Implications of Using the Database Approach (continued)

- Flexibility to change data structures:
  - » Database structure may evolve as new requirements are defined.
- Availability of current information:
  - » Extremely important for on-line transaction systems such as shopping, airline, hotel, car reservations.
- Economies of scale:
  - » Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

Historical Development of Database Technology

- Early Database Applications:
  - » The Hierarchical and Network Models were introduced in mid 1960s and dominated during the seventies.
  - » A bulk of the worldwide database processing still occurs using these models, particularly, the hierarchical model using IBM's IMS system.
- Relational Model based Systems:
  - » Relational model was originally introduced in 1970, was heavily researched and experimented within IBM Research and several universities.
  - » Relational DBMS Products emerged in the early 1980s

Historical Development of Database Technology (continued)

- Object-oriented and emerging applications:
  - » Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.
    - Their use did not take off broadly for various reasons.
  - » Many relational DBMSs have incorporated object database concepts, leading to a new category called *object-relational* DBMSs (ORDBMSs)
  - » *Extended relational* systems add further capabilities (e.g. for multimedia data, text, XML, and other data types)

Historical Development of Database Technology (continued)

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

Extending Database Capabilities

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

Extending Database Capabilities (continued)

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

Extending Database Capabilities (continued)

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

When Not to Use a DBMS

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

When Not to Use a DBMS (continued)

- 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).

Summary

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Types of Database Users
- Advantages of Using the Database Approach
- Historical Development of Database Technology
- Extending Database Capabilities
- When Not to Use Databases

Agenda

Section Outline

- Data Models and Their Categories
- History of Data Models
- Schemas, Instances, and States
- Three-Schema Architecture
- Data Independence
- DBMS Languages and Interfaces
- Database System Utilities and Tools
- Centralized and Client-Server Architectures
- Classification of DBMSs

Data Models

- **Data Model:**
  - » A set of concepts to describe the ***structure*** of a database, the ***operations*** for manipulating these structures, and certain ***constraints*** that the database should obey.
- **Data Model Structure and Constraints:**
  - » Constructs are used to define the database structure
  - » Constructs typically include ***elements*** (and their ***data types***) as well as groups of elements (e.g. ***entity, record, table***), and ***relationships*** among such groups
  - » Constraints specify some restrictions on valid data; these constraints must be enforced at all times
- **Data Model Operations:**
  - » These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.
  - » Operations on the data model may include ***basic model operations*** (e.g. generic insert, delete, update) and ***user-defined operations*** (e.g. compute_student_gpa, update_inventory)

Categories of Data Models

- **Conceptual (high-level, semantic) data models:**
  - » Provide concepts that are close to the way many users perceive data.
    - (Also called ***entity-based*** or ***object-based*** data models.)
- **Physical (low-level, internal) data models:**
  - » Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals
- **Implementation (representational) data models:**
  - » Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).
- **Self-Describing Data Models:**
  - » Combine the description of data with the data values. Examples include XML, key-value stores and some NoSQL systems.

Schemas vs. Instances

- Database Schema:
  - The ***description*** of a database.
  - Includes descriptions of the database structure, data types, and the constraints on the database.
- Schema Diagram:
  - An ***illustrative*** display of (most aspects of) a database schema.
- Schema Construct:
  - A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.
- Database State:
  - The actual data stored in a database at a ***particular moment in time***. This includes the collection of all the data in the database.
  - Also called database instance (or occurrence or snapshot).
    - The term *instance* is also applied to individual database components, e.g. *record instance, table instance, entity instance*

Database Schema vs. Database State

- Database State:
  - » Refers to the ***content*** of a database at a moment in time.
- Initial Database State:
  - » Refers to the database state when it is initially loaded into the system.
- Valid State:
  - » A state that satisfies the structure and constraints of the database
- Distinction
  - » The ***database schema*** changes very infrequently.
  - » The ***database state*** changes every time the database is updated.

- **Schema** is also called **intension**.
- **State** is also called **extension**.

Example of a Database Schema

Figure 2.1
Schema diagram for the database in Figure 1.2.

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Example of a Database State

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

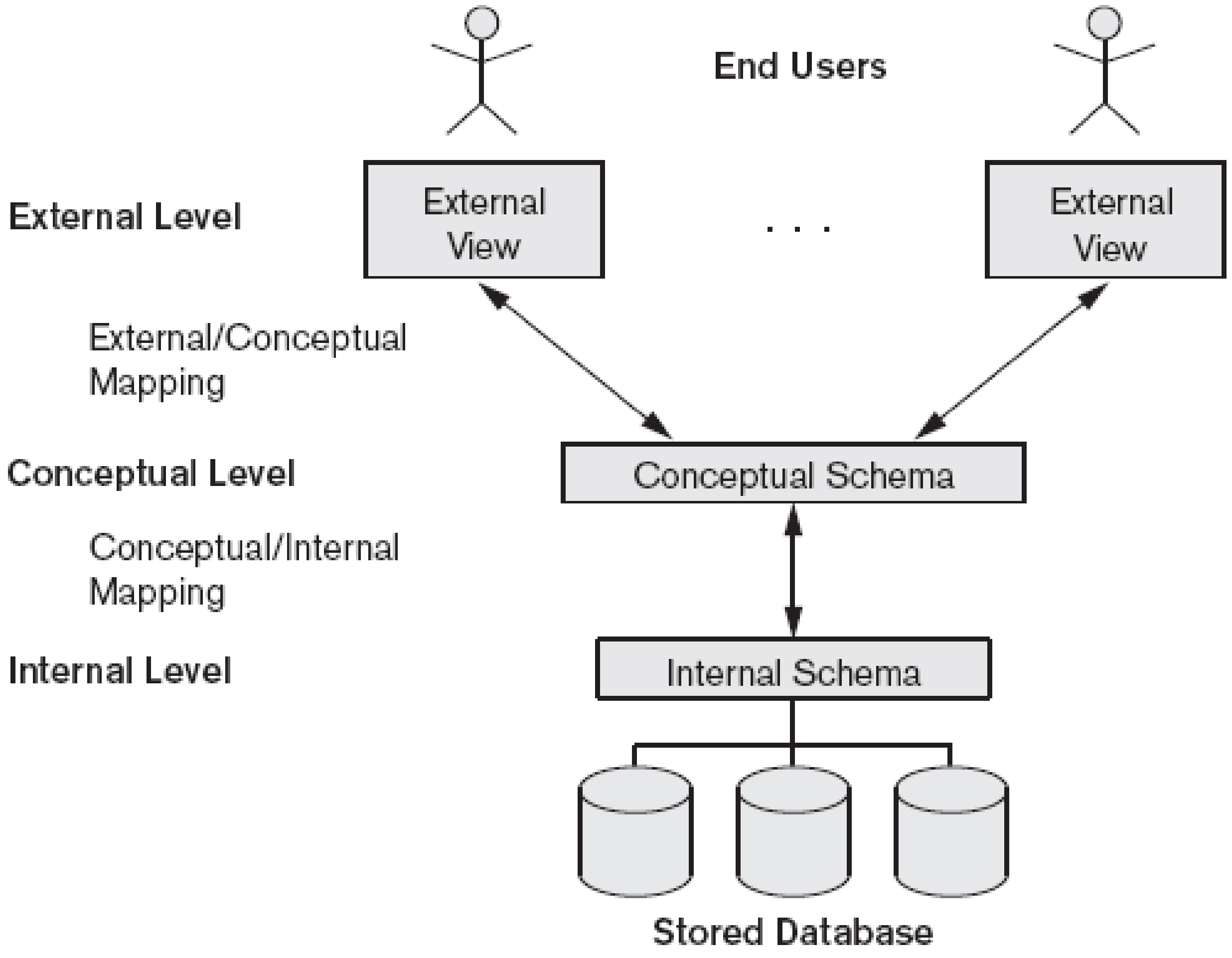
Figure 1.2
A database that stores student and course information.

Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - **Program-data independence.**
  - Support of **multiple views** of the data.
- Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization
- Defines DBMS schemas at ***three*** levels:
  - **Internal schema** at the internal level to describe physical storage structures and access paths (e.g indexes).
    - Typically uses a **physical** data model.
  - **Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users.
    - Uses a **conceptual** or an **implementation** data model.
  - **External schemas** at the external level to describe the various user views.
    - Usually uses the same data model as the conceptual schema.

The Three-Schema Architecture

Figure 2.2
The three-schema architecture.



Three-Schema Architecture

- Mappings among schema levels are needed to transform requests and data.
  - » Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
  - » Data extracted from the internal DBMS level is reformatted to match the user’s external view (e.g. formatting the results of an SQL query for display in a Web page)

Data Independence

- **Logical Data Independence:**
  - » The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.
- **Physical Data Independence:**
  - » The capacity to change the internal schema without having to change the conceptual schema.
  - » For example, the internal schema may be changed when certain file structures are re-organized or new indexes are created to improve database performance
- When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence.
- The higher-level schemas themselves are **unchanged**.
  - » Hence, the application programs need not be changed since they refer to the external schemas.

DBMS Languages

- Data Definition Language (DDL)
  - Used by the DBA and database designers to specify the conceptual schema of a database.
  - In many DBMSs, the DDL is also used to define internal and external schemas (views).
  - In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas.
    - SDL is typically realized via DBMS commands provided to the DBA and database designers

DBMS Languages (continued)

- Data Manipulation Language (DML)
  - High-Level or Non-procedural Languages: These include the relational language SQL
    - May be used in a standalone way or may be embedded in a programming language
  - Low Level or Procedural Languages:
    - These must be embedded in a programming language
  - Used to specify database retrievals and updates
  - DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
    - A library of functions can also be provided to access the DBMS from a programming language
  - Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

Types of DML

- **High Level or Non-procedural Language:**
  - For example, the SQL relational language
  - Are “set”-oriented and specify what data to retrieve rather than how to retrieve it.
  - Also called **declarative** languages.
- **Low Level or Procedural Language:**
  - Retrieve data one record-at-a-time;
  - Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

DBMS Interfaces

- Stand-alone query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)
- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
  - Menu-based, forms-based, graphics-based, etc.
- Mobile Interfaces are 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 Database 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

Database Systems Utilities

- To perform certain functions such as:
  - » Loading data stored in files into a database. Includes data conversion tools.
  - » Backing up the database periodically on tape.
  - » Reorganizing database file structures.
  - » Performance monitoring utilities.
  - » Report generation utilities.
  - » Other functions, such as sorting, user monitoring, data compression, etc.

Other Tools

- Data dictionary / repository:
  - Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.
  - **Active data dictionary** is accessed by DBMS software and users/DBA.
  - **Passive data dictionary** is accessed by users/DBA only.
- Application Development Environments and CASE (computer-aided software engineering) tools:
- Examples:
  - PowerBuilder (Sybase originally)
  - JBuilder (Borland and now MicroFocus)
  - JDeveloper 1xG (Oracle)

The Database System Environment

- DBMS component modules
  - Buffer management
  - Stored data manager
  - DDL compiler
  - Interactive query interface
    - Query compiler
    - Query optimizer
  - Pre-compiler
  - Runtime database processor
  - System catalog
  - Concurrency control system
  - Backup and recovery system

Typical DBMS Component Modules

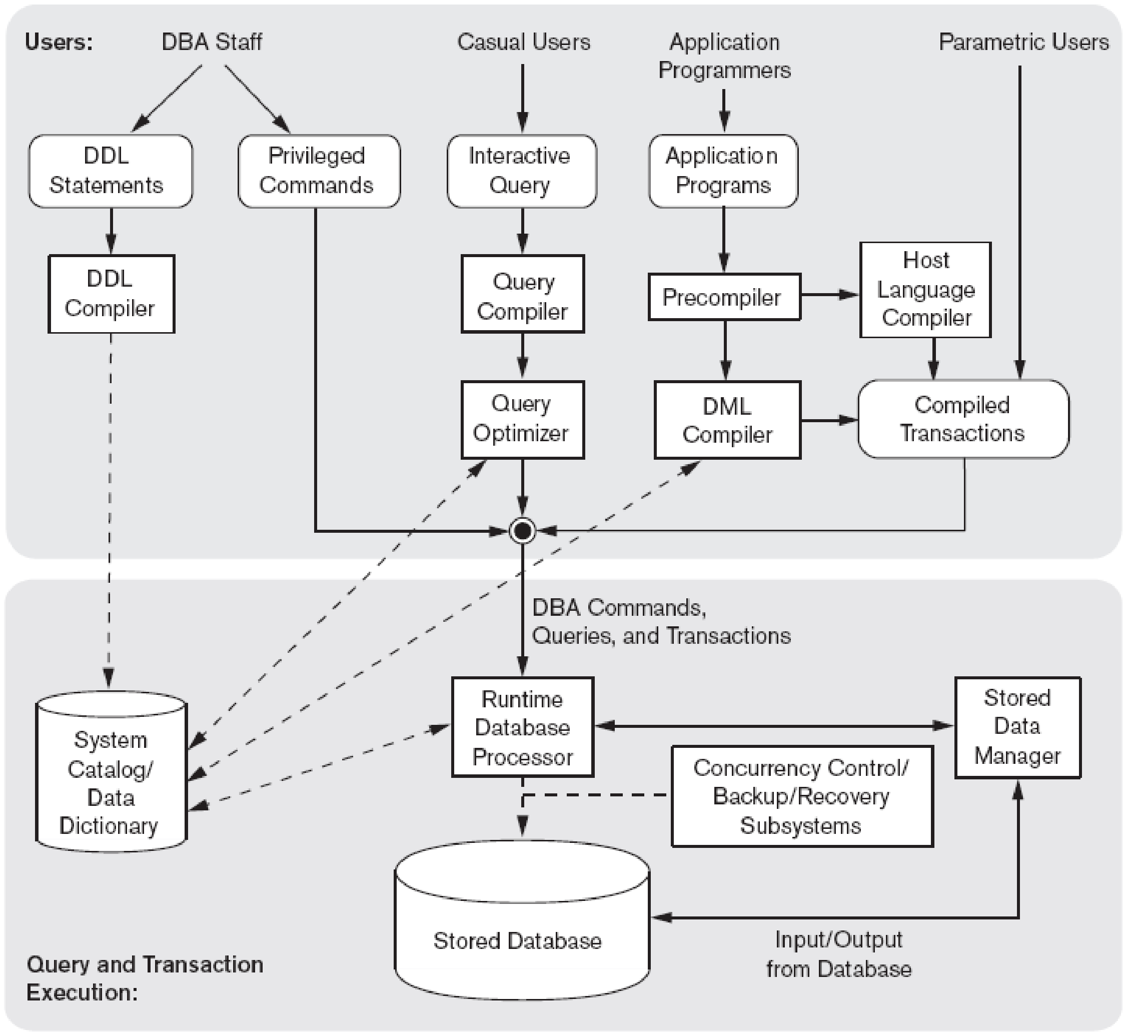


Figure 2.3
Component modules of a DBMS and their interactions.

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.

A Physical Centralized DBMS Architecture

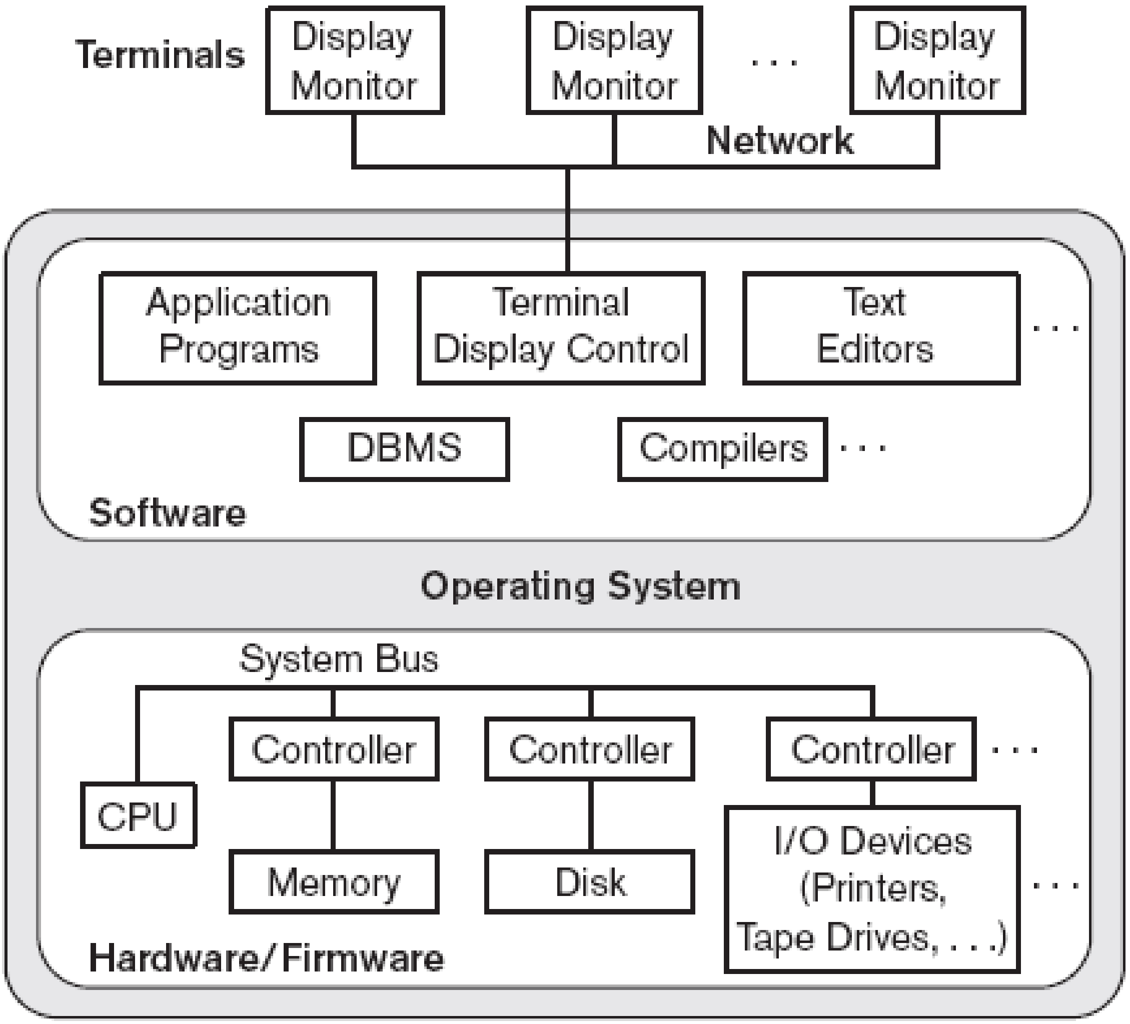


Figure 2.4
A physical centralized architecture.

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

Logical and Physical Two-Tier Client / Server Architecture

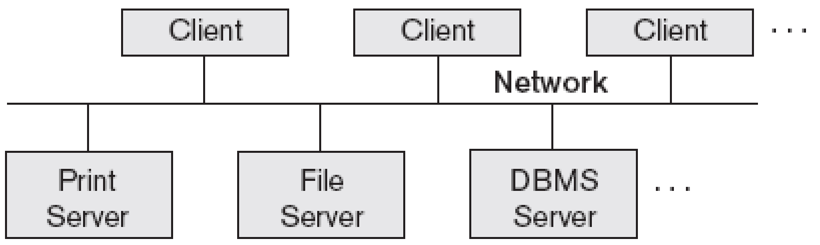


Figure 2.5
Logical two-tier client/server architecture.

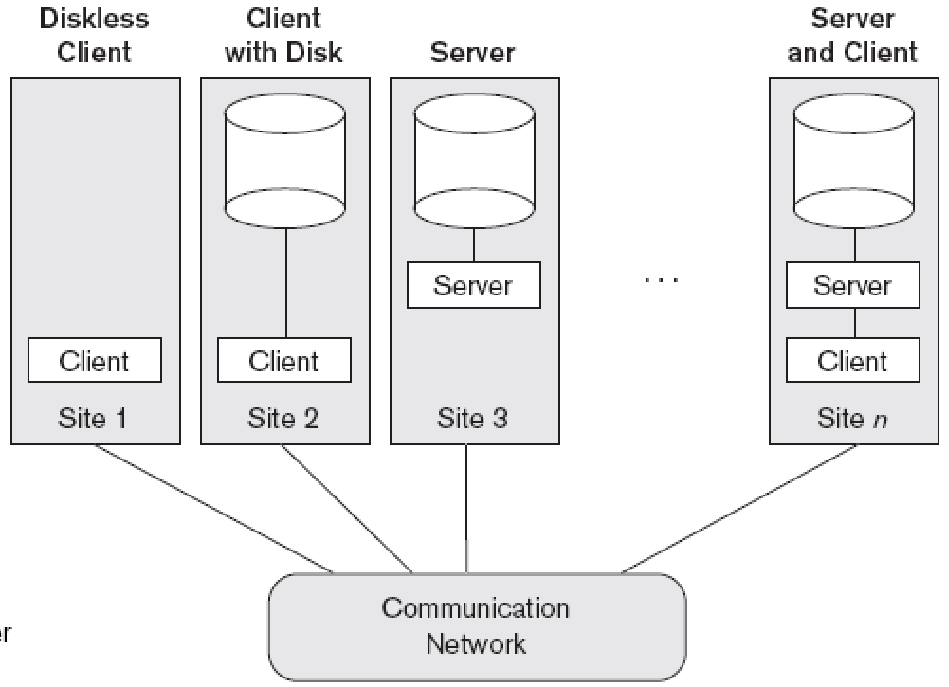


Figure 2.6
Physical two-tier client/server architecture.

DBMS Clients

- 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

Two Tier Client-Server Architecture

- Client and server must install appropriate client module and server module software for ODBC or JDBC
- A client program may connect to several DBMSs, sometimes called the data sources.
- In general, data sources can be files or other non-DBMS software that manages data.

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 part of the application used to access the corresponding data from the database server
  - » Acts like a conduit for sending partially processed data between the database server and the client.
- Three-tier Architecture Can Enhance Security:
  - » Database server only accessible via middle tier
  - » Clients cannot directly access database server
  - » Clients contain user interfaces and Web browsers
  - » The client is typically a laptop or a mobile device connected to the Web

Three-Tier Client/Server Architecture

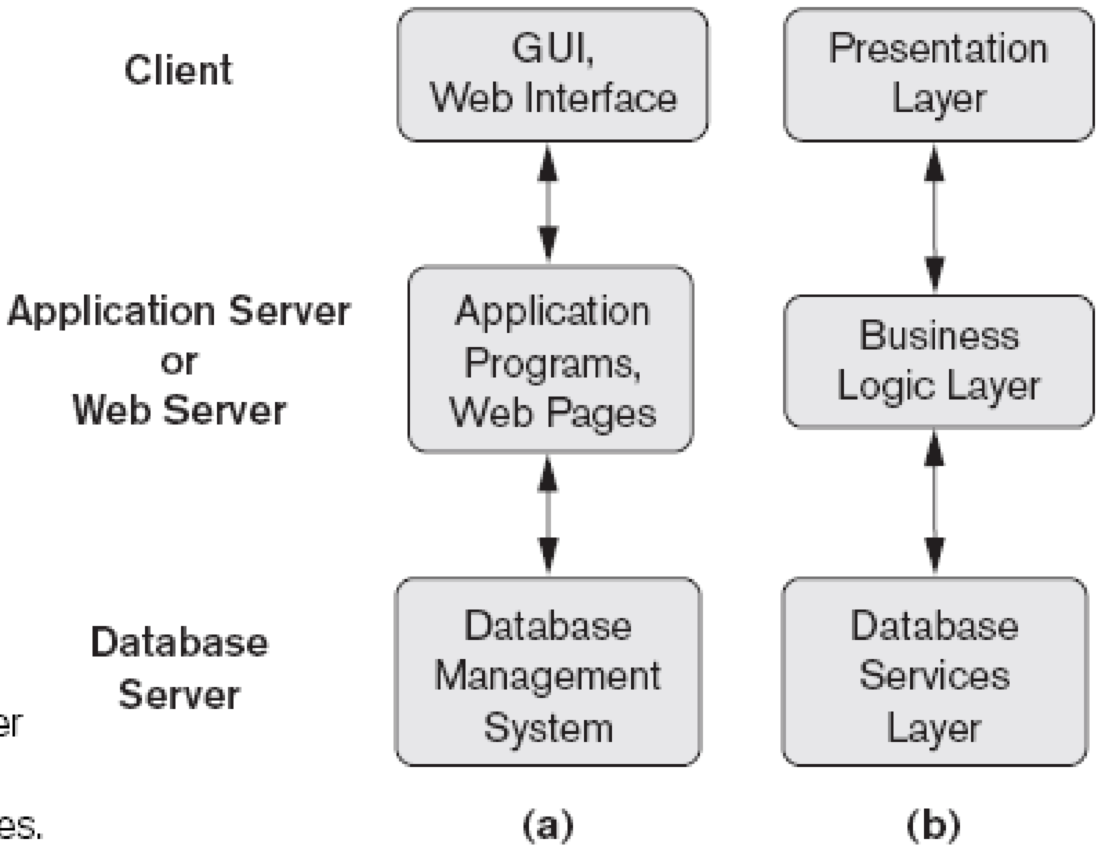


Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

Classification of Database Management Systems

- 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/JSON DBMSs, NewSQL 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)

Variations of Distributed DBMSs (DDBMSs)

- Homogeneous DDBMS
- Heterogeneous DDBMS
- Federated or Multi-database 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.

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/JSON 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, subscription, 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)

History of Data Models (Additional Material)

- Network Model
- Hierarchical Model
- Relational Model
- Object-oriented Data Models
- Object-Relational Models

History of Data Models – Network Model

- **Network Model:**
  - The first network DBMS was implemented by Honeywell in 1964-65 (IDS System).
  - Adopted heavily due to the support by CODASYL (Conference on Data Systems Languages) (CODASYL - DBTG report of 1971).
  - Later implemented in a large variety of systems - IDMS (Cullinet - now Computer Associates), DMS 1100 (Unisys), IMAGE (H.P. (Hewlett-Packard)), VAX -DBMS (Digital Equipment Corp., next COMPAQ, now H.P.).

History of Data Models – Network Model

- Advantages:
  - Network Model is able to model complex relationships and represents semantics of add/delete on the relationships.
  - Can handle most situations for modeling using record types and relationship types.
  - Language is navigational; uses constructs like FIND, FIND member, FIND owner, FIND NEXT within set, GET, etc.
    - Programmers can do optimal navigation through the database.
- Disadvantages:
  - Navigational and procedural nature of processing
  - Database contains a complex array of pointers that thread through a set of records.
    - Little scope for automated “query optimization”

History of Data Models – Network Model

- Types of access path options
- General or special-purpose

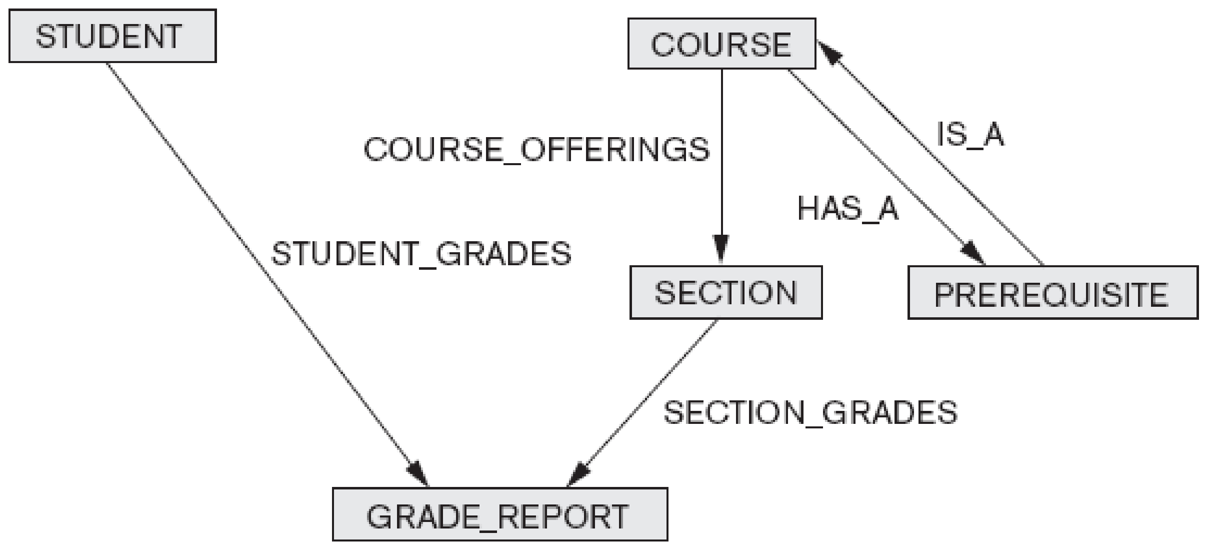


Figure 2.8
The schema of Figure 2.1 in network model notation.

[14]CODASYL DBTG stands for Conference on Data Systems Languages Database Task Group, which is the committee that specified the network model and its language.

History of Data Models – Hierarchical Model

- **Hierarchical Data Model:**
  - » Initially implemented in a joint effort by IBM and North American Rockwell around 1965. Resulted in the IMS family of systems.
  - » IBM's IMS product had (and still has) a very large customer base worldwide
  - » Hierarchical model was formalized based on the IMS system
  - » Other systems based on this model: System 2k (SAS inc.)

History of Data Models – Hierarchical Model

- Advantages:
  - Simple to construct and operate
  - Corresponds to a number of natural hierarchically organized domains, e.g., organization (“org”) chart
  - Language is simple:
    - Uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT, etc.
- Disadvantages:
  - Navigational and procedural nature of processing
  - Database is visualized as a linear arrangement of records
  - Little scope for "query optimization"

History of Data Models – Relational Model

- **Relational Model:**
  - » Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82.
  - » Now in several commercial products (e.g. DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX).
  - » Several free open source implementations, e.g. MySQL, PostgreSQL
  - » Currently most dominant for developing database applications.
  - » SQL relational standards: SQL-89 (SQL1), SQL-92 (SQL2), SQL-99, SQL3, …

History of Data Models – Object-Oriented Data Models

- **Object-oriented Data Models:**
  - » Several models have been proposed for implementing in a database system.
  - » One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE).
  - » Additionally, systems like O2, ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).
  - » Object Database Standard: ODMG-93, ODMG-version 2.0, ODMG-version 3.0.

History of Data Models – Object-Relational Models

- **Object-Relational Models:**
  - The trend to mix object models with relational was started with Informix Universal Server.
  - Relational systems incorporated concepts from object databases leading to object-relational.
  - Exemplified in the versions of Oracle, DB2, and SQL Server and other DBMSs.
  - Current trend by Relational DBMS vendors is to extend relational DBMSs with capability to process XML, Text and other data types.
  - The term “object-relational” is receding in the marketplace.

Section Summary

- Data Models and Their Categories
- Schemas, Instances, and States
- Three-Schema Architecture
- Data Independence
- DBMS Languages and Interfaces
- Database System Utilities and Tools
- Database System Environment
- Centralized and Client-Server Architectures
- Classification of DBMSs
- History of Data Models

Agenda

1 **Instructor and Course Introduction**

2 **Databases and Database Users**

3 **Database System Concepts and Architecture**

➡ 4 **Summary and Conclusion**

Course Assignments

- Individual Assignments
    - Reports based on case studies / class presentations
    - Textbook problem sets
- Project-Related Assignments
    - All assignments (other than the individual assessments) will correspond to milestones in the course project

Assignments & Readings

- Readings

  - Slides and Handouts posted on the course web site
  - Textbook: Chapters 1 & 2

- Assignment #1 – Database R&D Exercise (Report)
  - (a) select a product from the following types of database systems (at least two):
    - XML Database, ODBMs, NoSQL Databases, Cloud Databases, etc.
  - (b) write a short report to explain the capabilities and inner-workings
  - (c) demonstrate the use of the database systems of your choice on small example(s) of your choice

Next Session: Relational Data Model and Relational Database Constraints

Any Questions?