CSIE30600/CSIEB0290 Database Systems

Lecture 1: Introduction

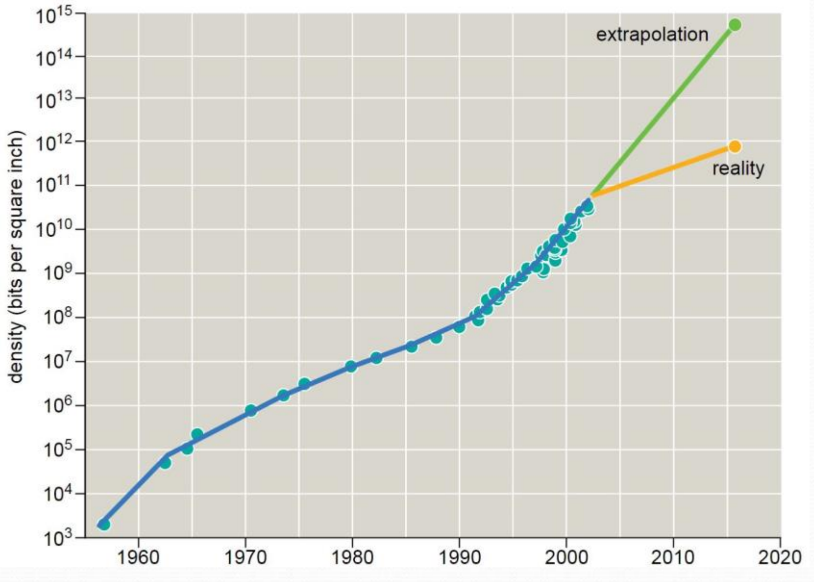
Outline

- Data is ubiquitous
- Basic Definitions
- Types of Databases and Database Applications
- Typical DBMS Functionalities
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Database Users
- Advantages of Using the Database Approach
- History of Database Systems
- Extending Database Capabilities
- When Not to Use Databases

Data & Technological Advances

- Five classes of technological advances are changing our relationship with data:
- More storage space
  - allows us to keep more data
- Faster processor (and memory) speeds
  - allows us to access and process more data
- Better networking
  - allows us to share data more efficiently
- Different "sensors"
  - allows us to access new kinds of data
- Better processing methods (AI & machine learning)
  - allows us to process data more intelligently

HDD Areal Density Growth

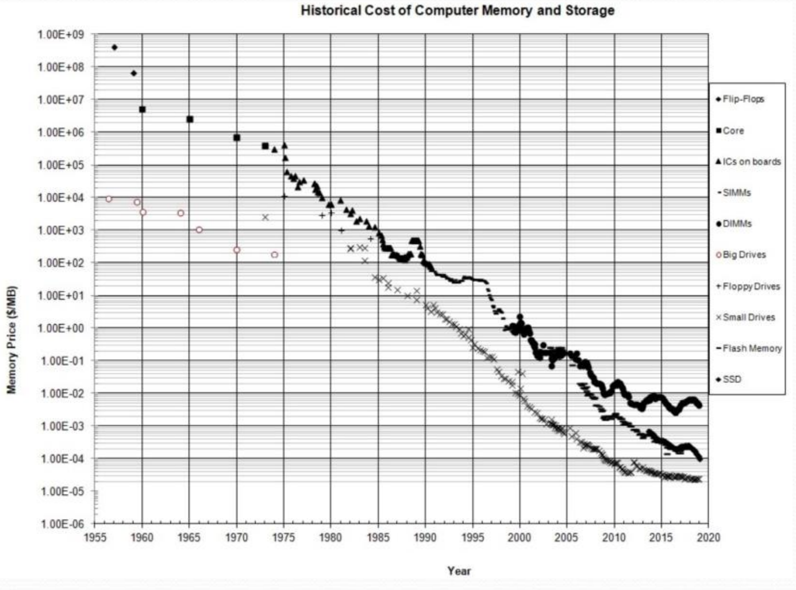


HDD Cost over Time

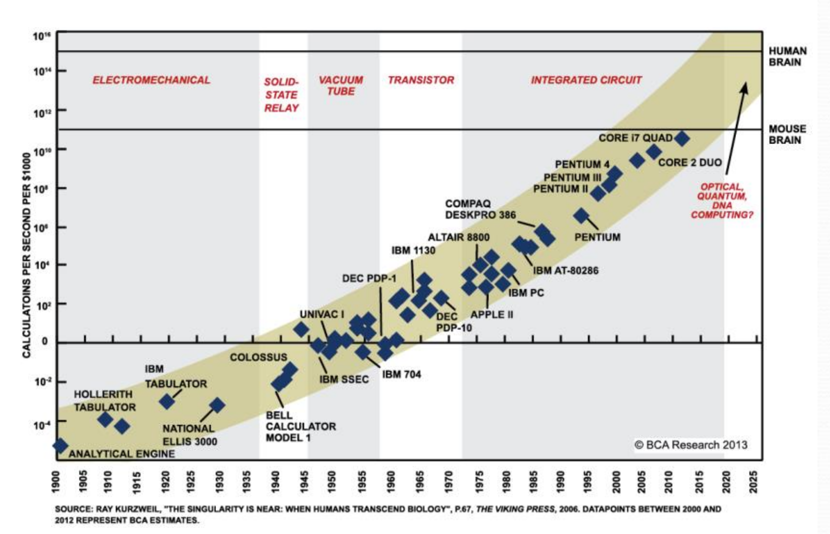
Hard drive cost per GB over time

date	capacity	cost	$/GB
1957	3.75 MB	$34,500	$9.2 million/GB
1989	40 MB	$1,200	$30,000/GB
1995	1 GB	$850	$850/GB
2004	250 GB	$250	$1/GB
2011	2 TB	$70	$0.035/GB
2018	4 TB	$75	$0.019/GB

HDD Cost Trends

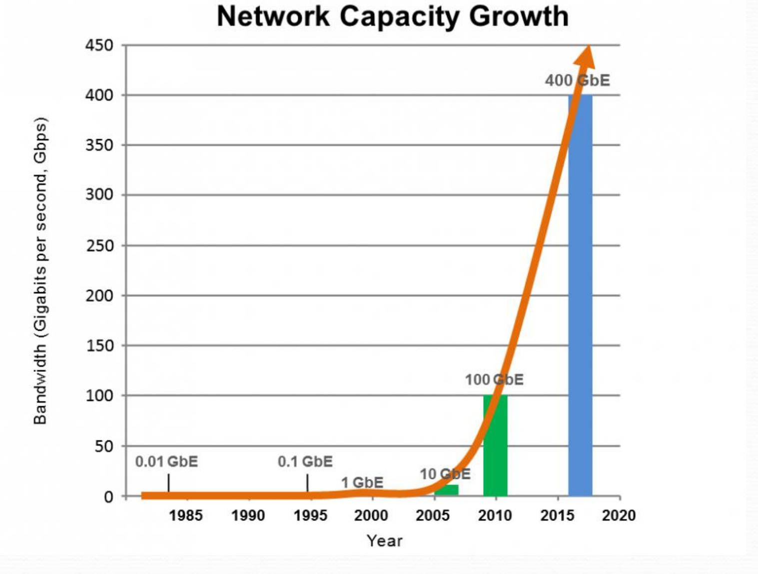


Processor Technology

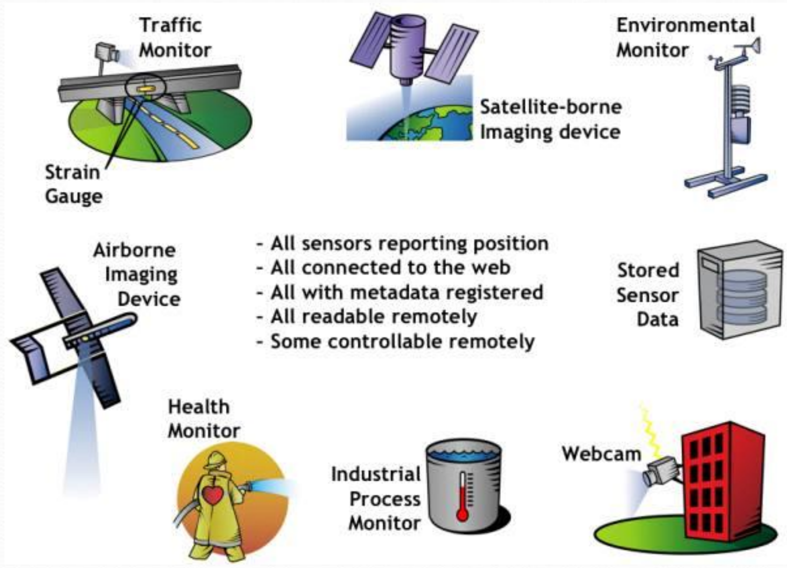


http://i1.wp.com/thetechguysblog.com/wp-content/uploads/2015/01/Chart-III-8-Moores-Law-Over-199-Years-And-Going-Strong.png

Network Capacity Growth



New Data from Sensor Web



Data Everywhere

- Airline flight management system
- Financial data
- Commercial store (eg, WalMart) data
- Department of Motor Vehicles
- Surveillance video
- University student records
- Baseball results
- Web sites
- Medical records
- …

Need Effective Data Management

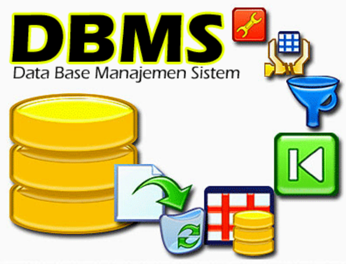
- Effective management can make an organization's data a valuable asset(資產).
- Ineffective policies can make an organization's data a liability(負債).
- Big data analytics is becoming the gold mine of the 21st century.
- The paradigm has been extended from database systems to data science.

Basic Concepts

- Data: Known facts that can be recorded and have an implicit meaning.
- Database: Collection of interrelated data
- Mini-World or Universe of Discourse (UoD): Some part of the real world about which data is stored in a database.
- Database Management System (DBMS): A collection of programs to facilitate the creation and maintenance of a database.
- Database System = DBMS + Database
- A database system contains information about a particular enterprise.
- A database system provides an environment that is both convenient and efficient to use.

Database Management System (DBMS)

- DBMS is:
  - A collection of software programs
  - General purpose
- DBMS enables users to:
  - Define DB
  - Construct DB
  - Change (or update) DB
  - Query the data in DB
  - Share DB
- DBMS maintains the integrity of DB



DBMS Popularity Ranking

☐ include secondary database models — 138 systems in ranking, February 2019

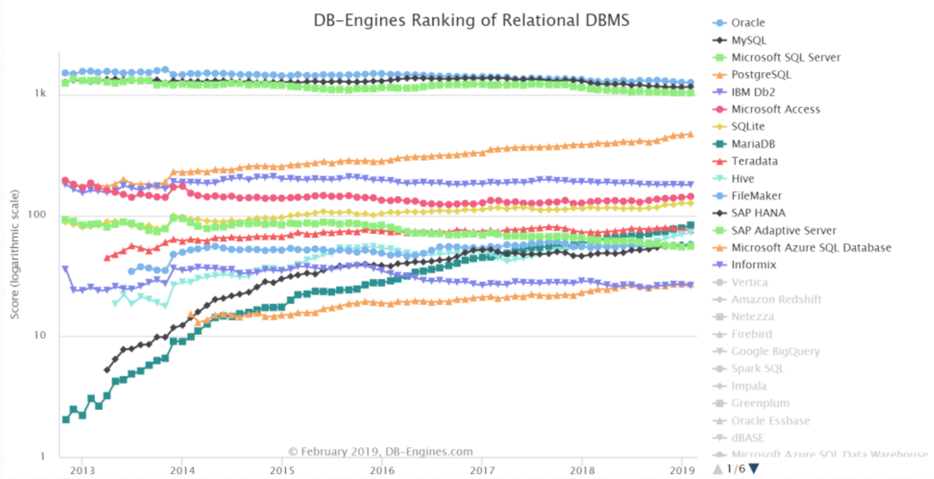
Rank Feb 2019	Rank Jan 2019	Rank Feb 2018	DBMS	Database Model	Score Feb 2019	Score Jan 2019	Score Feb 2018
1.	1.	1.	Oracle	Relational, Multi-model	1264.02	-4.82	-39.26
2.	2.	2.	MySQL	Relational, Multi-model	1167.29	+13.02	-85.18
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1040.05	-0.21	-81.98
4.	4.	4.	PostgreSQL	Relational, Multi-model	473.56	+7.45	+85.18
5.	5.	5.	IBM Db2	Relational, Multi-model	179.42	-0.43	-10.55
6.	6.	6.	Microsoft Access	Relational	144.02	+2.41	+13.95
7.	7.	7.	SQLite	Relational	126.17	-0.63	+8.89
8.	8.	↑ 10.	MariaDB	Relational, Multi-model	83.42	+4.60	+21.77
9.	9.	↓ 8.	Teradata	Relational	75.97	-0.22	+2.98
10.	10.	↑ 11.	Hive	Relational	72.29	+2.38	+17.23
11.	11.	↑ 12.	FileMaker	Relational	57.79	+0.64	+3.43
12.	12.	↑ 13.	SAP HANA	Relational, Multi-model	56.55	-0.09	+9.19
13.	13.	↓ 9.	SAP Adaptive Server	Relational	55.75	+0.71	-7.74
14.	14.	↑ 15.	Microsoft Azure SQL Database	Relational, Multi-model	27.12	-0.07	+3.34
15.	15.	↓ 14.	Informix	Relational, Multi-model	26.35	-0.41	-2.03
16.	16.	16.	Vertica	Relational, Multi-model	22.81	+1.03	+2.84
17.	↑ 18.	↑ 20.	Amazon Redshift	Relational	20.99	+0.94	+7.87
18.	↑ 19.	18.	Netezza	Relational	19.77	+0.12	+3.19
19.	↓ 17.	↓ 17.	Firebird	Relational	19.35	-0.83	+1.67
20.	20.	↑ 21.	Google BigQuery	Relational	18.75	+0.38	+6.40

http://db-engines.com/en/ranking/relational+dbms

DBMS Ranking Trend

February 2019

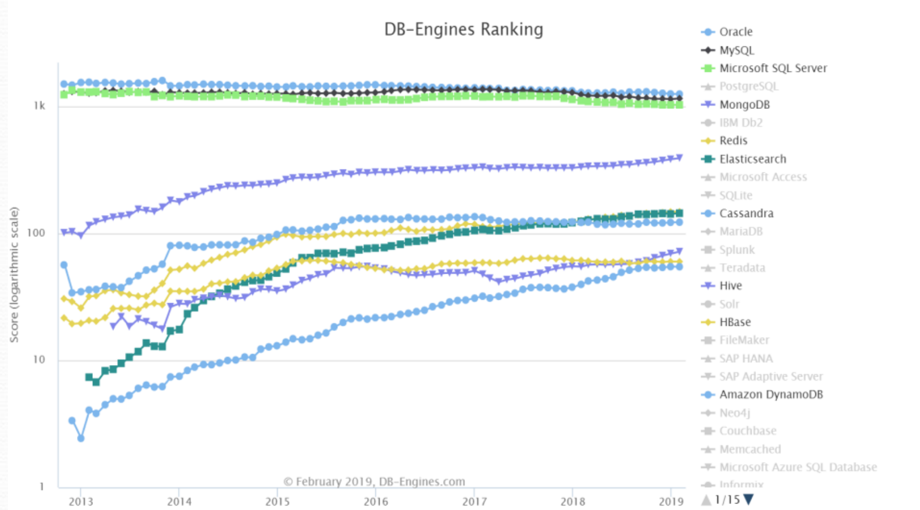
include secondary database models



http://db-engines.com/en/ranking_trend/relational+dbms

Growing Popularity of NoSQL

February 2019



Main Goals of DB Course

- To understand how to use a DBMS
  - How to design and create DB, data models, SQL, …
- To understand how a DBMS works
  - Physical properties of disks and files, software to manage reading and writing to disk, implementation of algorithms to answer user queries, …

Types of Databases and Applications

- Traditional Applications:
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases (images, audio, video, …)
  - Geographic Information Systems (GIS)
  - Data Warehouses
  - Real-time and Active Databases
  - Many other applications
- New Trends: big data analytics, IoT
- First part: focuses on traditional applications
- *A number of recent applications are described later in the class and book.*

Database System Environment

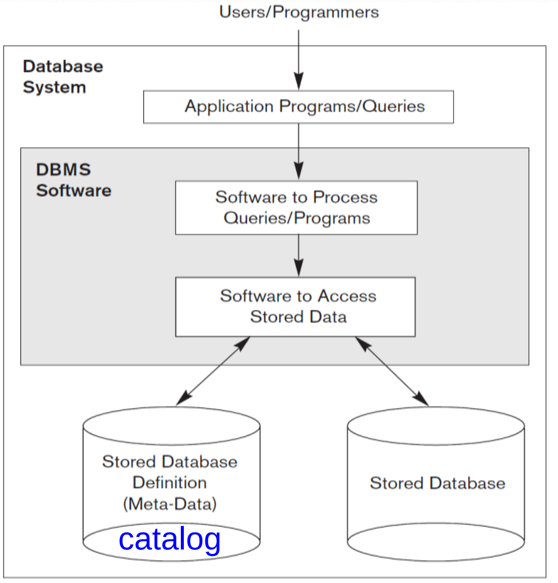


Figure 1.1
A simplified database system environment.

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

Typical DBMS Functionality

- Other features:
  - Protection or Security measures to prevent unauthorized access
  - "Actively" take internal actions on data
  - Presentation and Visualization of data
  - Maintaining the database and associated programs over the lifetime of the database application
    - Called database, software, and system maintenance

Application Activities Against a DB

- 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

Database Applications

- **Banking**: all transactions
- **Airlines**: reservations, schedules
- **Universities**: registration, grades
- **Sales**: customers, products, purchases
- **Online retailers**: order tracking, customized recommendations
- **Manufacturing**: production, inventory, orders, supply chain
- **Human resources**: employee records, salaries, tax deductions
- **Databases touch all aspects of our lives**

Purpose of Database Systems

- In the early days, database applications were built directly on top of **file systems**
- Drawbacks of using file systems :
  - **Data redundancy** and **inconsistency**
    - Multiple file formats, duplication in different files
  - **Difficulty in accessing** data
    - Need to write a new program for each new task
  - **Data isolation** — multiple files and formats
  - **Integrity** problems
    - Integrity constraints (e.g. account balance > 0) become "buried" in program code rather than being stated explicitly
    - Hard to add new constraints or change existing ones

Purpose of Database Systems

- Drawbacks of using file systems (cont.)
  - **Atomicity** of updates
    - Failures may leave database in an inconsistent state
    - Example: Transfer of funds from one account to another
  - **Concurrent access** by multiple users
    - Concurrent accessed needed for performance
    - Uncontrolled concurrent accesses can lead to inconsistencies
      - Example: Two people reading and updating at the same time
  - **Security problems**
    - Hard to provide user access to some, but not all, data
- Database systems offer solutions to **ALL** the above problems

Example of a Database

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

Example of a Database

- **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 (to be discussed later in class)

Example of a 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

Figure 1.2
A database that stores student and course information.

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

Main Characteristics

- Self-describing: A DBMS catalog (meta-data) stores the description of the database. (next slide)
- Program-data Independence: Allows changing storage structures w/o changing DBMS access programs.
- Data abstraction: Data models hide storage details and present the users with a conceptual view of the DB.
- Multiple views: Each user may see a different view of the database.
- Data sharing: among multiple users
- Transactions, concurrent access , recovery , OLTP

A Simplified Database Catalog

RELATIONS

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

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

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 enumerated type with all known majors.
XXXXNNNN is used to define a type with four alphabetic characters followed by four numeric digits.

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
  - 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 Administrator

- Coordinates all the activities of the database system; must have a good understanding of the enterprise's information resources and needs.
- Database administrator's duties:
  - Schema definition
  - Storage structure and access method definition
  - Schema and physical organization modification
  - Granting user authority to access the database
  - Specifying integrity constraints
  - Acting as liaison with users
  - Monitoring performance and responding to changes in requirements

End-users

- Actors on the scene (continued)
  - End-users: use the data for queries, reports and some of them update the database content.
- End-users can be categorized into:
  - Casual: access db 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.
    - Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.

End-users (cont.)

- Sophisticated:
  - 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 a tax program user that creates its own internal database.
  - Another example is a user that maintains an address book

System Analysts and Application Programmers

- System analysts: Analyze problem, determine the requirements of the users, develop specifications.
- Application programmers: Design and implement specification, testing, debugging, maintaining softwares. Also known as software developers or software engineers.
- Business analysts: There is an increasing need for people who can analyze vast amounts of business data and real-time data ("Big Data") for better decision making, planning, advertising, marketing etc.

Users behind the Scene

- DB designers – design the database systems for end users
- DBMS designers – design database management systems and tools for building databases
- Tool designers – Design and implement tools that facilitate building of applications and allow using database effectively (eg. modeling and designing databases, performance monitoring, prototyping, test data generation, user interface creation, simulation etc.)
- Operators and maintenance personnel.

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.
- Providing persistent storage for program objects
  - In object-oriented DBMS
- Providing storage structures (e.g. indexes) efficient data access
- Provide optimization of queries for efficient processing

Advantages of Using the Database Approach (cont.)

- 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

Additional Implications

- 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 (cont.)

- 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 airline, hotel, car reservations.
- Economies of scale:
  - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

History of Database Systems

- **Pre-1960s**
  - File processing systems
  - Redundancy and inconsistency between files
  - Incompatibility between access programs
  - Data isolation
  - Concurrent access anomalies
  - Security and integrity problems

DB History (cont.)

- **The '60s**
  - Carles Bachman designed the 1st DBMS Integrated Data Store (and received the 1st Turing Award in 1973)
  - Three-level architecture (more about this in next lecture)
  - CODASYL, DBTG, and the network model
  - Hierarchical model and the IMS system

DB History (cont.)

- **The '70s**
  - Edgar Codd (1970): The Relational model (Codd won the 1981 Turing Award)
  - Provide a sound theoretical base.
  - 1975, 1st ACM SIGMOD international conference
  - 1975, 1st VLDB international conference
  - Peter Chen 陳品山 (1976): The Entity-relationship model
  - System R (IBM), INGRES (UC-Berkely), System 2000 (UT-Austin)
  - SQL, QUEL

DB History (cont.)

- **The '80s**
  - Commertcial relational DBMS (DB2, ORACLE, SYBASE, INFORMIX, ...)
  - DBMS on PC's (DBASE, PARADOX, ...)
  - Transaction management (James Gray won the 1999 Turing Award)
  - Standards (SQL standardized in the late 1980s)

DB History (cont.)

- **The '90s**
  - New applications (Web, CAD/CAM, CASE, office automation, science and engineering, VLSI, ...)
  - Demand for new DBMS technologies
  - Object-oriented DBs, Parallel/Distributed DBs, Active/Deductive DBs, Multimedia DBs, Mobile DBs, Temporal/Real-time DBs, Spatial DBs(such as GIS), ...
  - The emergence of ERP (Enterprise Resource Planning) and MRP (Material Requirements Planning) packages
  - Data Warehousing and data mining
  - DBMS in the Internet/Web and E-commerce applications

DB History (cont.)

- **The 2000s and beyond**
  - XML, XQuery and the Semantic Web
  - Data Stream Management Systems (DSMS)
    - Sensor databases
    - Network traffic analysis
    - RFID data management
    - …
  - Mobile Data Management (MDM)

New Trend

- 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

Emergence of Big Data & NoSQL

- New data storage, management and analysis technology was necessary to deal with the huge volumn of data in petabytes a day (10^{15} bytes or 1000 terabytes) in some applications – "Big Data".
- Hadoop and Mapreduce programming approach to distributed data 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) systems have been designed for rapid search and retrieval from documents, processing of huge graphs, and other forms of unstructured data with flexible models of transaction processing

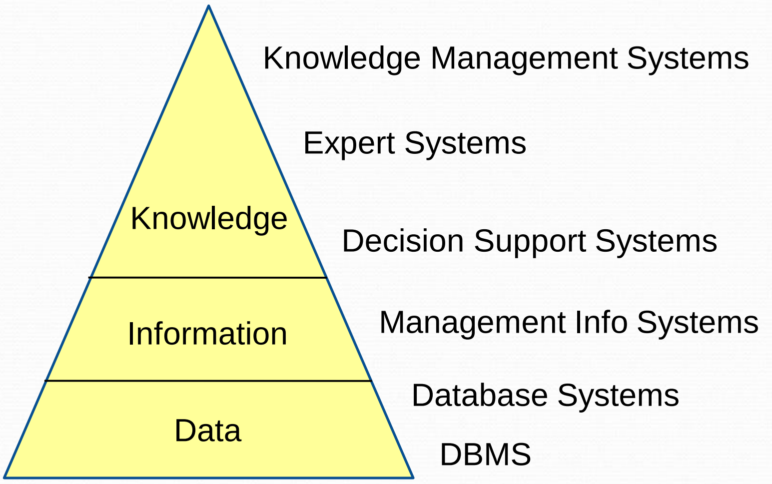
When NOT to Use a DBMS

- Main inhibitors (costs) of using a DBMS:
  - High initial investment and possible need for additional hardware. (No longer the case with cloud)
  - The generality that a DBMS provides for defining and processing data
  - Overhead for providing 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 NOT to use a DBMS

- When a DBMS may be infeasible:
  - In embedded systems where a general purpose DBMS may not fit in available storage
- When no DBMS may suffice:
  - If there are stringent real-time requirements that may not be met because of DBMS overhead
  - If the database system is not able to handle the complexity of data because of modeling limitations
  - If the database users need special operations not supported by the DBMS (e.g. GIS and location based services)

Related Systems



Summary

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Database Users
- Advantages of Using the Database Approach
- Database History and New Trend
- When NOT to Use Databases
- Related Systems