Lecture 1
Database Systems: An Introduction

February 26, 2014

Shuigeng Zhou
School of Computer Science
Fudan University

Outline

- Databases, Database Management Systems and Database Systems
- View of Data
- Database Languages
- Relational Databases
- Database Design
- Object-based and semi-structured databases
- Database Architecture
- Database Users and Administrators
- Overall Database System Structure
- Data Storage and Querying
- Transaction Management
- History of Database Systems
- Challenges & Opportunities

What is a database?

- A very large, integrated collection of data
  - The amount of data is very large
  - The data is structured and interrelated
  - The data is integrated
- Models real-world *enterprises* or *organizations*
  - Entities (e.g., students, courses)
  - Relationships (e.g., Li is taking *Database* course)
  - and active components (i.e. "business logic")
    - All courses are scored via A, A-, B+, B, B- etc.
- Databases touch all aspects of our lives

Database Applications

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades, students
- Sales: customers, products, purchases
- Online retailers: order tracking, customized recommendations
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions
- ……

Database Management System (DBMS)

- A *Database Management System (DBMS)* is a software package designed to store and manage databases
- Functions of DBMS
  - Manages very large amount of data
  - Supports efficient access to very large amount of data
  - Supports concurrent access to very large amount of data
    - Example: bank and its ATM machines
  - Supports secure, atomic access to very large amount of data
    - Contrast two people editing the same UNIX file – last to write "wins"

People working with DBMS

- End users: query/update databases through application user interfaces (e.g., Amazon.com etc.)
- Database designers: design database "schema" to model aspects of the real world
- Database application developers: build applications that interface with databases
- Database administrators (a.k.a. DBA's): load, back up, and restore data, fine-tune databases for performance
- DBMS implementors: develop the DBMS or specialized data management software, implement new techniques for query processing and optimization inside DBMS

Why use a DBMS?

- Data independence and efficient access
- Reduced application development time
- Data integrity and security
- Uniform data administration
- Concurrent access, recovery from crashes

Is a File System a DBMS?

- Thought Experiment 1:
  - You and your partner are editing the same file.
  - You both save it at the same time.
  - Whose changes survive?

A) Yours **B) Partner's** **C) Both** **D) Neither** **E) ???**

- Thought Experiment 2:
  - -You're updating a file.
  - -The power goes out.
  - -Which of your changes survive?

A) All **B) None** **C) All since last save** **D) All up to last save**

Is an Traditional IR System a DBMS?

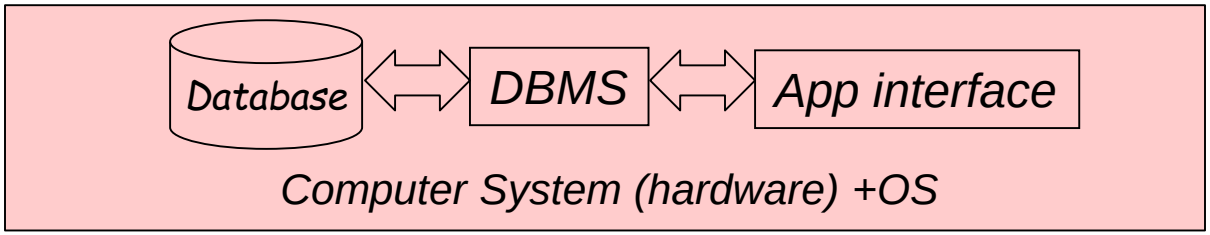
- Traditional IR system
  - Mainly for data retrieval, not data update
  - Store and manage free structured or semi-structured document
  - Support semantic-oriented matching
  - Consider no data consistency, and concurrency is easy to be implemented
- DBMS
  - Support data access and update
  - Store and manage structured data
  - Require declarative query language support
  - Support data consistency, atomic transaction, concurrency control and failure recovery

Is Web Search Engine a DBMS?

- Fairly sophisticated search available
  - crawler *indexes* pages; Keyword-based search for pages
  - data is mostly unstructured and untyped; search only:
    - can't modify the data
    - can't get summaries, complex combinations of data
  - few guarantees provided for freshness of data, consistency across data items, fault tolerance, …
  - Web sites (e.g. e-commerce) typically have a DBMS in the backend to provide these functions

Database Systems

- A Database System (DBS) contains the following components
  - Hardware platform (PC/Workstation/Cluster/Mainframe etc.)
  - DBMS
  - A (number of) database(s)
  - HI that is both *convenient* and *efficient* to use



DBS

Why Study Databases?

- Datasets increasing in diversity and volume
  - Digital libraries, interactive video, Human Genome project, EOS project
  - Applications' need for DBMS exploding
- Shift from *computation* to *management*
- DBMS encompasses most of CS
  - OS, languages, theory, AI, multimedia, logic

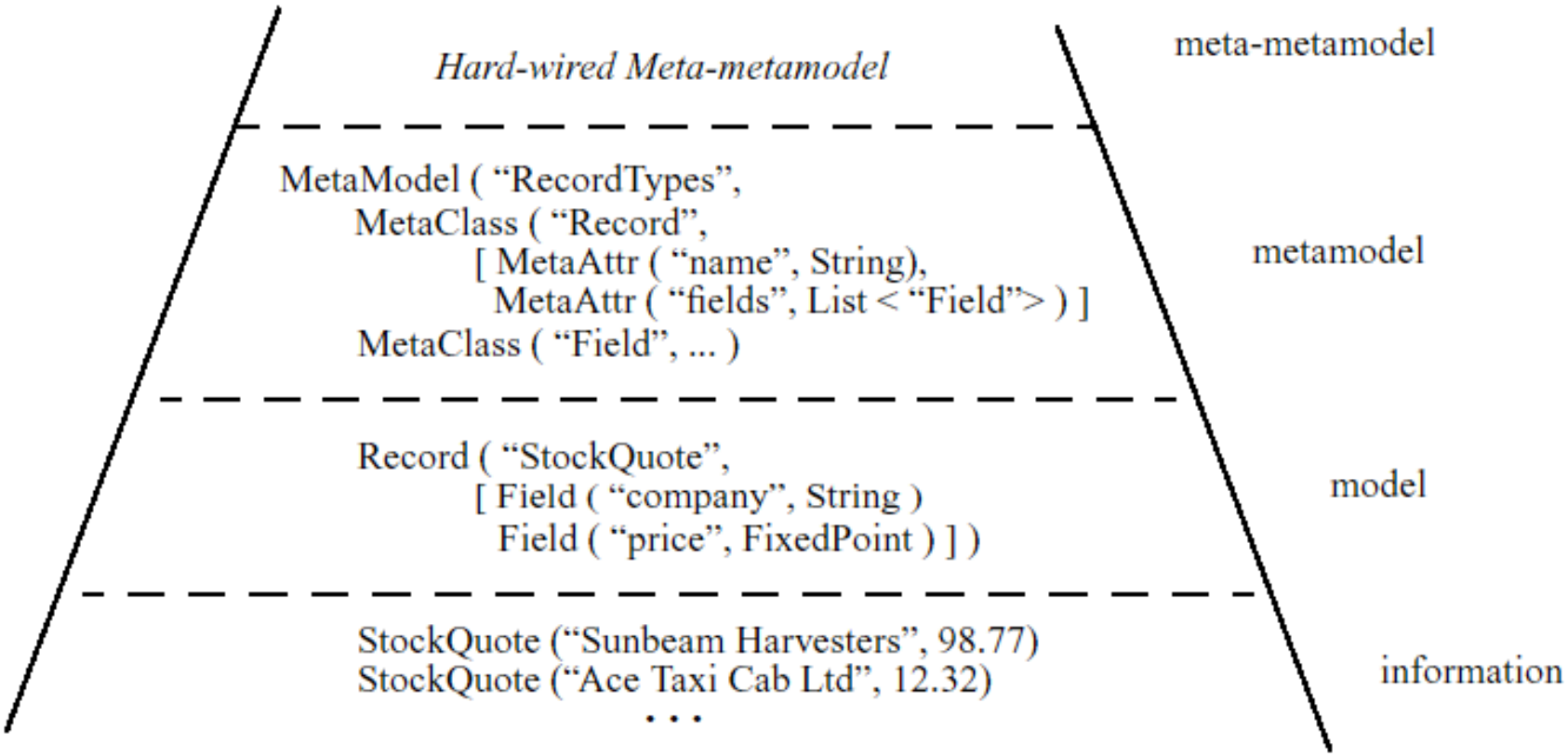
What's the Intellectual Content?

- Representing information
  - data modeling
- Languages and systems for querying data
  - complex queries with real semantics over massive data sets
- Concurrency control for data manipulation
  - controlling concurrent access
  - ensuring transactional semantics
- Reliable data storage
  - maintain data semantics even if you pull the plug

Data Models

- A collection of tools for describing
  - Data structures; Data relationships; Data semantics; Data constraints
- Entity-Relationship data model (mainly for database design)
- Relational model
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data model (XML)
- Other older models:
  - Network model
    - Example: Integrated Data Store (IDS), was designed by Charles Bachman at General Electric in the 1960s
  - Hierarchical model
    - Example: IBM Information Management System (IMS)

Data Abstraction



Levels of Datbase Abstraction

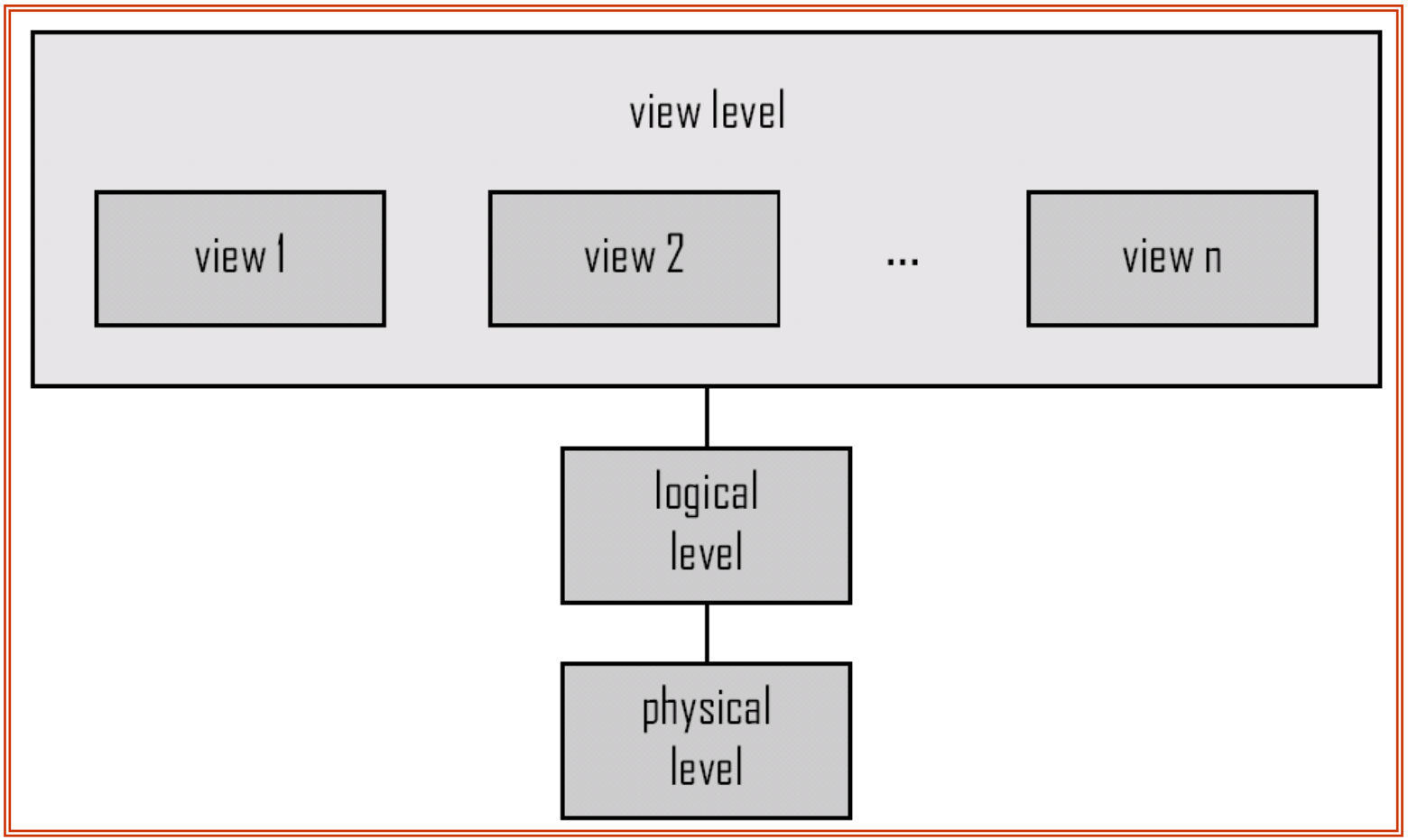
- **Physical level:** describes how a record (e.g., customer) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

```
type customer = record
        customer_id : string;
        customer_name : string;
        customer_street : string;
        customer_city : integer;
    end;
```

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - Example: The database consists of information about a set of customers and accounts and the relationship between them)
  - Analogous to type information of a variable in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point of time
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - **Procedural** – user specifies what data is required and how to get those data
  - **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data
- SQL is the most widely used query language

Data Definition Language (DDL)

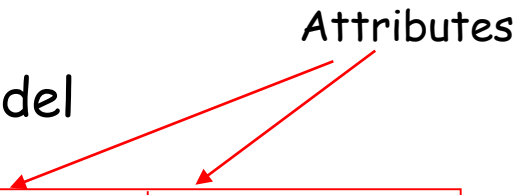
- Specification notation for defining the database schema

  Example: **create table** *account* (
  account-number **char**(10),
  balance **integer**)

- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Data *storage and definition* language
    - Specifies the storage structure and access methods used
  - Integrity constraints
    - Domain constraints
    - Referential integrity (**references** constraint in SQL)
    - Assertions
  - Authorization

Relational Model

- Example of tabular data in the relational model



Customer-id	*customer-name*	*customer-street*	*customer-city*	*account-number*
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

A Sample Relational Database

customer-id	*customer-name*	*customer-street*	*customer-city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	*account-number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

SQL

- **SQL**: widely used non-procedural language
  - Example: Find the name of the customer with customer-id 192-83-7465

```
select   customer.customer_name
from     customer
where    customer.customer_id = '192-83-7465'
```

  - Example: Find the balances of all accounts held by the customer with customer-id 192-83-7465

```
select   account.balance
from     depositor, account
where    depositor.customer_id = '192-83-7465' and
         depositor.account_number = account.account_number
```

- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

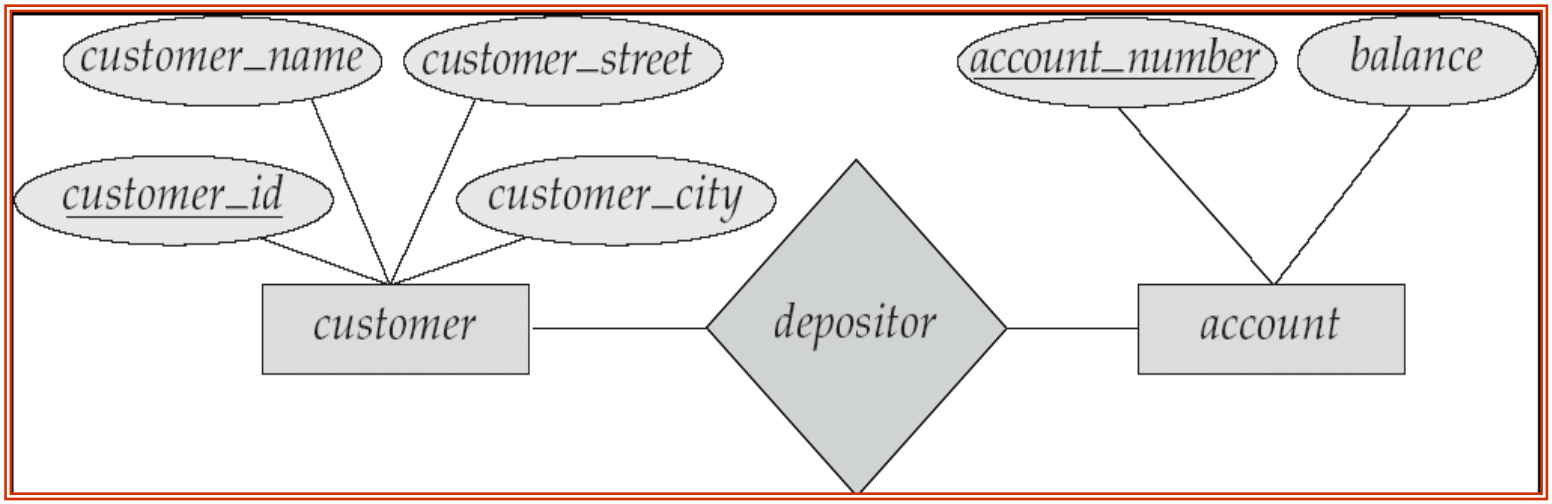
Database Design

The process of designing the general structure of the database:

- Conceptual Design - Deciding on the database conceptual schema via ER model
  - Business decision – What attributes should we record in the database?
- Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

The Entity-Relationship Model

- Models an enterprise as a collection of *entities* and *relationships*
  - Entity: a "thing" or "object" in the enterprise that is distinguishable from other objects
    - Described by a set of *attributes*
  - Relationship: an association among several entities
- Represented diagrammatically by an *entity-relationship diagram:*



Object-Relational Data Models

- Extend the relational data model by including object orientation and constructs to deal with added data types.
- Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
- Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
- Provide upward compatibility with existing relational languages.

XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
  - A subset of SGML (Standard Generalized Markup Language)
- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data

Database System Architecture

The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

- Centralized
- Parallel (multi-processor)
- Distributed
- Client-server/Browser-server

Database Users

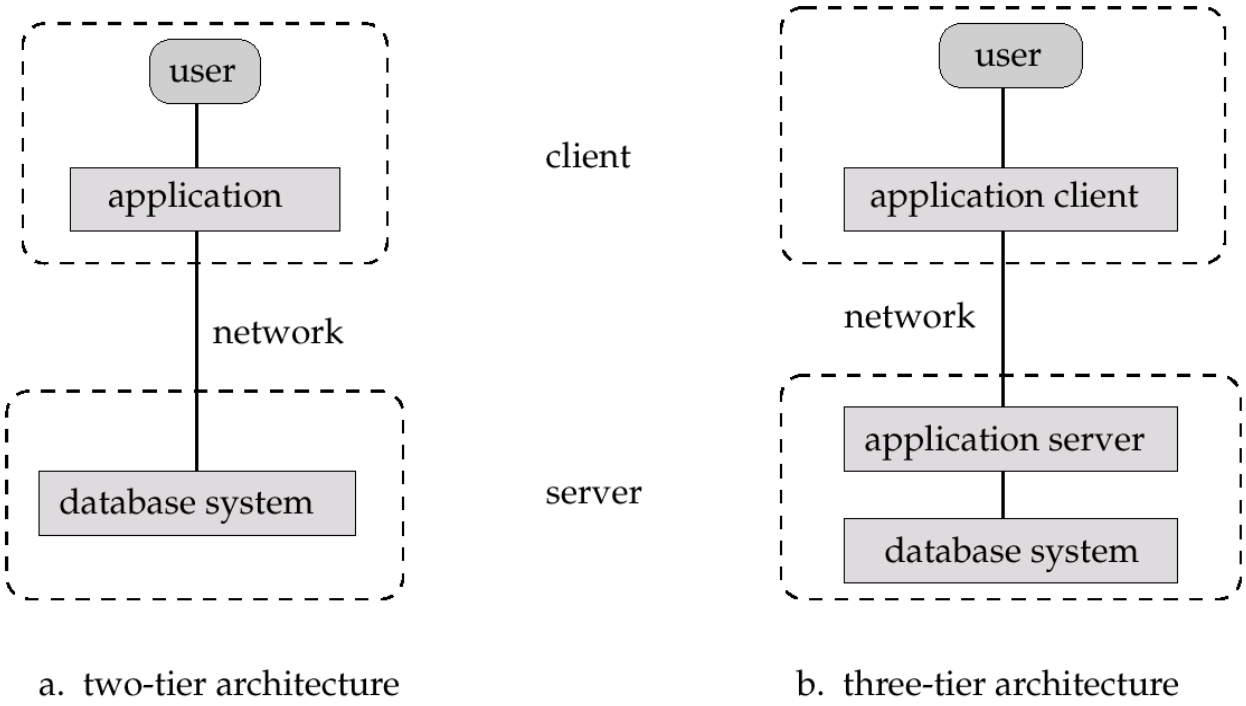
Users are differentiated by the way they expect to interact with the system

- **Naïve users** – invoke one of the permanent application programs that have been written previously
  - Examples, people accessing database over the web, bank tellers, clerical staff
- **Application programmers** – interact with system through DML calls
- **Sophisticated users** – form requests in a database query language
- **Specialized users** – write specialized database applications that do not fit into the traditional data processing framework

Database Administrator (DBA)

- Coordinates all the activities of the database system; the DBA should have a good understanding of the enterprise's information resources and needs, as well as DBMS
- Database administrator's duties include:
  - 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

Application Architectures

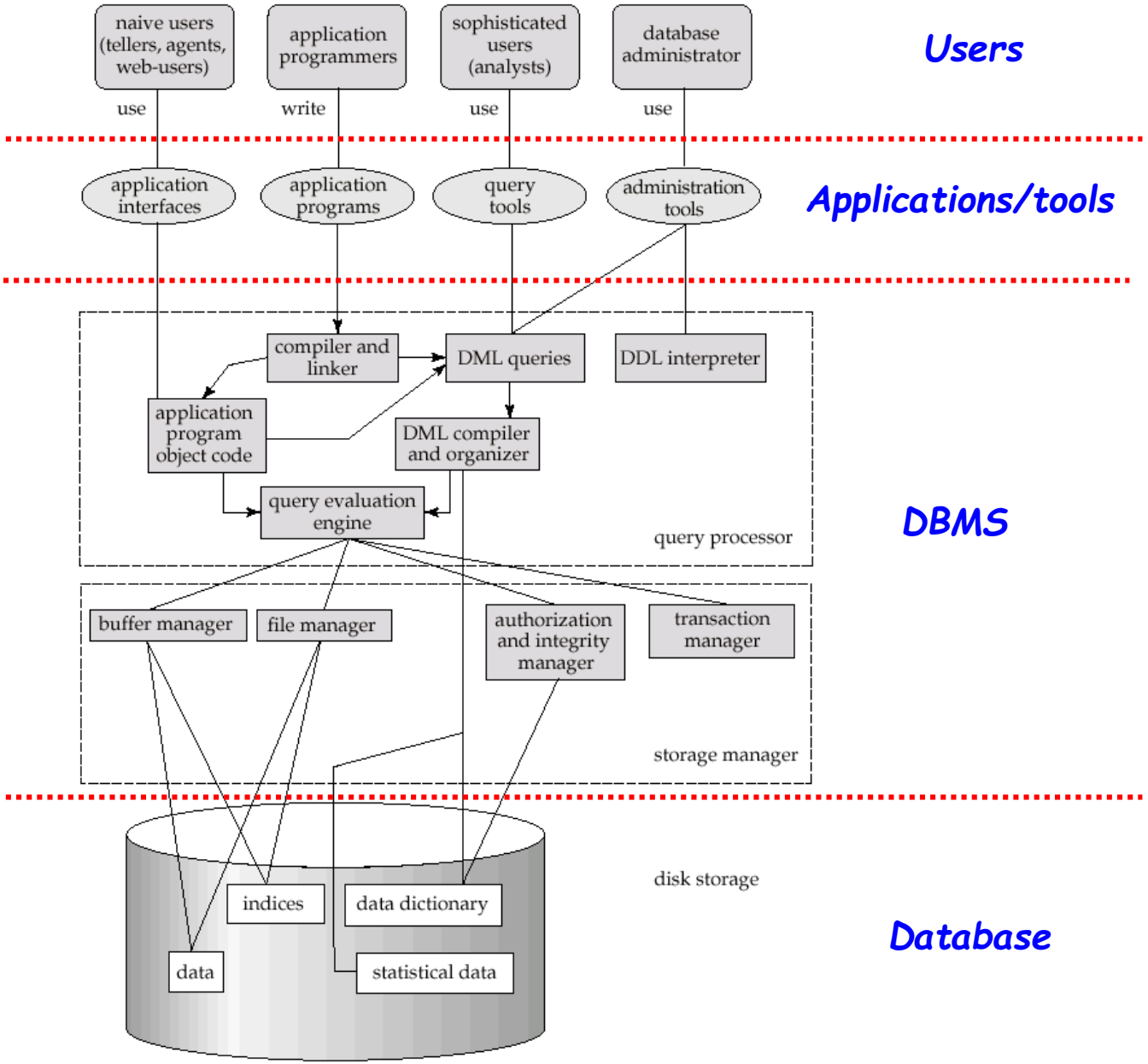


a. two-tier architecture

b. three-tier architecture

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

Database System Structure

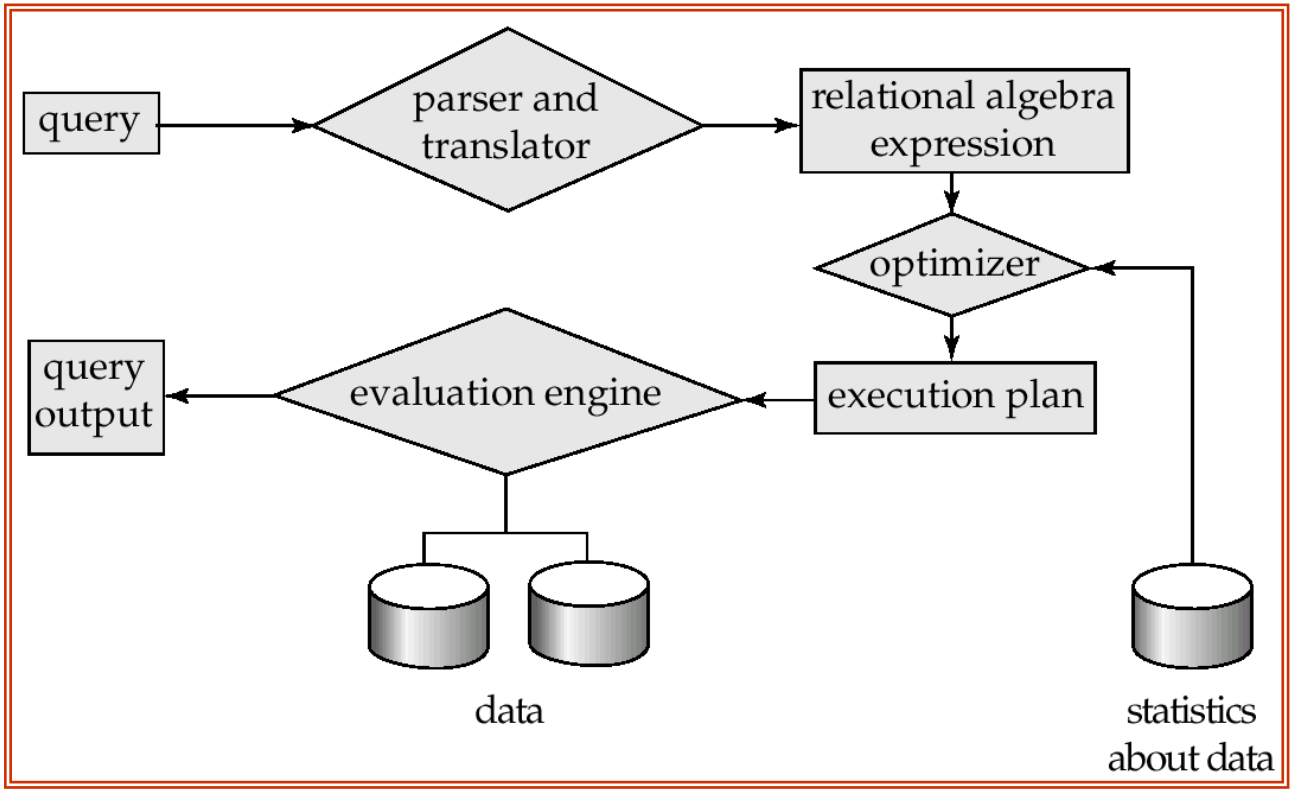


Storage Management

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

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Query Processing (Cont.)

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions

Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Major DBMS Today

- Major commercial DBMS products
  - Oracle
  - IBM DB2 (from System R, System R*, Starburst)
  - Microsoft SQL Server
  - Sybase
  - Informix (acquired by IBM)
  - NCR Teradata
  - Tandem NonStop (acquired by Compaq, now HP)
  - ……
- Open source DBMS
  - PostgreSQL (from UC Berkeley's Ingres, Postgres)
  - MySQL
  - ……

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provide only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allow direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Won the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley begins Ingres prototype
  - High-performance (for the era) transaction processing
- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems

History (cont.)

- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- 2000s:
  - XML and XQuery standards
  - Automated database administration
  - Streaming, mobile data management
  - P2P and sensor networks data management
  - Trusted data management (data privacy)
  - Data management over new platforms (e.g. Memory/flash; multi-cores)
  - Uncertain/probabilistic data management
  - Data-space
  - Cloud data management
  - Big data management
  - ……

Milestones in DBMS History (1)

- "Factoring out" data management functionalities from applications and standardizing these functionalities is an important first step
  - CODASYL (Conference on Data Systems Language) standard (circa 1960's)
  - Charles Bachman got a Turing award for this in 1973
    - Developed the first database management system IDS (Integrated Data Store)

Charles W. Bachman

(Dec. 11, 1924 – Now)

He received the ACM Turing Award in 1973 for *"his outstanding contributions to database technology"*

Milestones in DBMS History (2)

- The relational revolution (1970')
  - A simple data model: data is stored in relations (tables)
  - A declarative query language: SQL
  - Provides physical data independence
    - The single most important reason behind the success of DBMS today
    - And a Turing Award for E. F. Codd in 1981

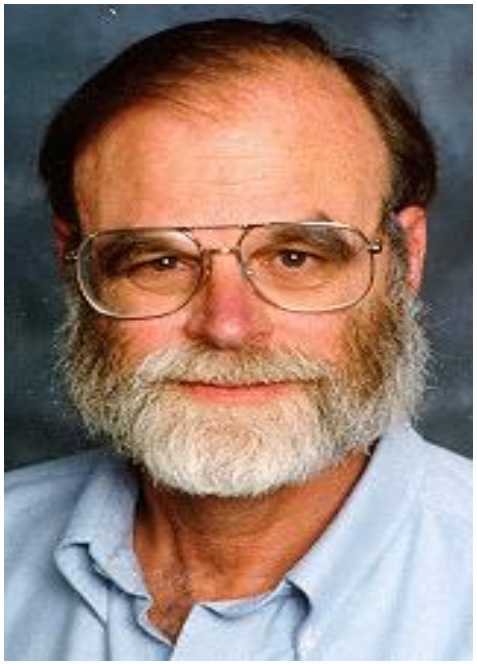
E. F. Codd, A Relational Model of Data for Large Shared Data Banks, CACM 13, No. 6, June 1970.

Edgar F. Codd

(August 23rd, 1923 - April 18th, 2003)

Milestones in DBMS History (3)

- Transaction processing in 1980s
- Jim Gray got a Turing Award for this work in 1998



(1944-2007)

Received the Turing Award in 1998 "for seminal contributions to database and transaction processing research and technical leadership in system implementation."

Jim Gray was lost at sea on 28 Jan., 2007.

In 2008, Microsoft announced the opening of **Microsoft Jim Gray Systems Lab** in Madison, Wisconsin.

Who Will Be the Next Turing Award Winner in Database Field? (1)

- Some information about the three Turing Award winners in database field
  - Graduate from famous universities
    - Charles Bachman (Master of Upen)
    - E. F. Codd (Bachelor of Oxford, PhD of Univ. of Michigan)
    - Jim Gray (the first CS PhD of Berkeley)
  - All worked for IT industries for a long time
  - Active in academy and contributed a lot to database development

Who Will Be the Next Turing Award Winner in Database Field? (2)

- My guess: Michael Stonebraker (1943-)
  - Bachelor (Princeton); Master/PhD (U. of Michigan)
  - Prof. (Berkeley); Adjunct Prof. (MIT)
  - His career covers, and helped create, the majority of the existing relational database market today. He is also the founder of Ingres, Illustra, StreamBase Systems, Vertica, VoltDB, SciDB and was previously the CTO of Informix
  - He has received several awards, including the IEEE John von Neumann Medal and the first SIGMOD Edgar F. Codd Innovations Award. In 1994 he was inducted as a Fellow of ACM

An overview on the 2008 Claremont Report

Database Research: Challenges and Opportunities

A Great Tradition

- Over the last twenty years, small groups of database researchers, practitioners and opinionated professionals have periodically gathered to assess the state of the field and propose directions for future research
- A final report will come out after each meeting, which aims to serve various functions: to foster debate within the database research community, to explain research directions to external organizations, and to help focus community efforts on timely challenges
- The latest meeting was held in late May, 2008, at the Claremont Resort in Berkeley, California, which is the seventh meeting of this sort in 20 years

the Claremont Report

- The participants of Claremont meeting
  - Eric A. Brewer, Michael Stonebraker, Joseph M. Hellerstein, Michael J. Franklin (Berkeley)
  - Rakesh Agrawal (Yahoo!)
  - Philip A. Bernstein, Surajit Chaudhuri (Microsoft)
  - Michael J. Carey (UC Irvine); AnHai Doan (UWM)
  - Hector Garcia - Molina (Stanford)
  - Johannes Gehrke (Cornell), Le Gruenwald (OU)
  - Laura M. Haas (IBM)
  - Raghu Ramakrishnan (Google); Alon Y. Halevy (Washington U./Google!)
  - Samuel Madden (MIT); Hank F. Korth (Lehigh U.); Alexander S. Szalay (HJU)
  - Roger Magoulas, Tim O'Reilly (O'Reilly Media)
  - Donald Kossmann (ETH); Anastasia Ailamaki (EPFL)
  - Gerhard Weikum (MPI for CS); Yannis E. Ioannidis (UOA); Daniela Florescu (INRIA)
  - **Beng Chin Ooi (NUS)**, Sunita Sarawagi (IIT Bombay)

Challenges

- **Big Data**
  - the number of communities working with large volumes of data has grown considerably, to include not only traditional enterprise applications and Web search, but also "e-science" efforts (in astronomy, biology, earth science, etc.), digital entertainment, natural language processing, social network analysis, and more
- **Data analysis as a profit center (Data center / Cloud Computing)**
- **Ubiquity of structured and unstructured data**
- **Expanded developer demands**
  - Programmer adoption of relational DBMSs and query languages has grown significantly in recent years
- **Architectural shifts in computing**
  - At the macro scale, the rise of "cloud" computing services suggests fundamental changes in software architecture
  - At a micro scale, computer architectures have shifted the focus of Moore's Law from increasing clock speed per chip to increasing the number of processor cores and threads per chip
  - In storage technologies, major changes are underway in the memory hierarchy, due to the availability of more and larger on-chip caches, large inexpensive RAM, and flash memory
  - Power consumption has become an increasingly important aspect of the price/performance metric of large systems

Opportunities (1)

- **Revisiting Database Engines**
  - designing systems for clusters of many-core processors
  - exploiting remote RAM and Flash as persistent media, rather than relying solely on magnetic disk
  - treating query optimization and physical data layout as a unified, adaptive, self-tuning task to be carried out continuously
  - compressing and encrypting data at the storage layer, integrated with data layout and query optimization;
  - designing systems that embrace non-relational data models, rather than "shoehorning" them into tables
  - trading off consistency and availability for better performance and scaleout to thousands of machines
  - designing power-aware DBMSs that limit energy costs without sacrificing scalability (Green DBMS)

Opportunities (2)

- **Declarative Programming for Emerging Platforms**
- **The Interplay of Structured and Unstructured Data**
  - A significant long-term goal for our community is to transition from managing traditional databases consisting of well-defined schemata for structured business data, to the much more challenging task of managing a rich collection of structured, semi-structured and unstructured data, spread over many repositories in the enterprise and on the Web. This has sometimes been referred to as the challenge of managing dataspaces

Opportunities (3)

- **Cloud Data Services**
- **Mobile Applications and Virtual Worlds**
  - In the mobile space, 1) the platforms on which to build mobile applications (i.e., the hardware, software and network) are maturing to the point that they have attracted large user bases, and can ubiquitously support very powerful interactions "on the go". 2) emergence of mobile search and social networks suggests an exciting new set of mobile applications
  - Virtual worlds like Second Life are growing quickly in popularity

Conclusion

- DBMS used to maintain, query large datasets
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security
- Levels of abstraction give data independence
- A DBMS typically has a layered architecture
- DBAs hold responsible jobs and are well-paid! ☺
- DBMS R&D is still one of the broadest, most exciting areas in CS

Some useful terms

- Byte
- Kilobyte – 2^{10} bytes (i.e. 1024 bytes)
- Megabyte – 2^{20} bytes (i.e. 1024 kilobytes)
- Gigabyte – 2^{30} bytes (i.e. 1024 megabytes)
- Terabyte – 2^{40} bytes (i.e. 1024 gigabytes)
  - A handful of these for files in EECS
  - Biggest single online DB is Wal-Mart, >100TB
- Petabyte – 2^{50} bytes (i.e. 1024 terabytes)
- Exabyte – 2^{60} bytes (i.e. 1024 petabytes)
- Zettabyte – 2^{70} bytes (i.e. 1024 exabytes)
- Yottabyte – 2^{80} bytes (i.e. 1024 zettabytes)

What is Big Data?

- It's said that the term "**Big data**" in its current use was coined by *Roger Magoulas* @ O'Reilly Media
- There is not a consensus as to how to define big data

"Big data exceeds the reach of commonly used hardware environments and software tools to capture, manage, and process it within a tolerable elapsed time for its user population."
- Teradata Magazine article, 2011

"Big data refers to data sets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze."
- The McKinsey Global Institute, 2011

The Vs of Big Data

- 3Vs model
  - high-volume, high-velocity, and/or high-variety
    - Gartner (2012)
- 4Vs models
  - Volume, velocity, variety and **virtual**
    - Courtney Lambert (2012)
  - Volume, velocity, variety and **veracity**
    - IBM (2012)
  - Volume, velocity, variety and **value**
    - DataStax (2012)
- **5Vs model**
  - *Volume, velocity, variety, veracity and value*

Why is Big Data Hot?

- Applications
  - Sensor networks, social networks, Internet search indexing, astronomy, atmospheric science, genomics, military surveillance, medical records, video archives, and large-scale e-commerce
- Market & industry
- Government
  - In 2012, the Obama administration announced the *Big Data Research and Development Initiative*
- Academia
  - Deal with Data, *Science*, Feb. 2011 issue
  - Big data, *Nature*, vol. 455, no. 7209, 2008

What's Big Data for?

- Gartner's big data definition

*"Big Data are high-volume, high-velocity, and/or high-variety information assets that require new forms of processing to enable enhanced **decision making**, **insight discovery** and **process optimization**."*

Data -> Knowledge -> Business Intelligence

Big data -> "Big" Knowledge -> "Big" Intelligence

Technologies Make Big Data

- Internet/Web
- High-resolution sensors
- Mobile devices
- RFID
- High-throughput sequencing
- ……

Technologies of Big Data

- Scalable storage systems
- Distributed file systems
- Parallel and distributed databases,
- Machine learning
- Data mining
- Natural language processing
- Visualization
- ……

End of Lecture 1