Database Principle

Zhuo Wang

zhuowang@sjtu.edu.cn

2017 Spring
Overview

- Data
- Database
- Database Management System
- Database System
References

- **Database System Concepts**

- **数据库系统概念**
  杨冬青，李红燕，唐世渭. 机械工业出版社

- **A First Course in Database Systems**

- **数据库系统概论**
  王珊 萨师煊 著 高等教育出版社
Software

- Oracle
- MySQL
- SQL Server
- PostgreSQL
- Kingbase
Organization of course

- Ch1. Introduction

- **Section 1: Relational database**
  - Ch2. Relational model
  - Ch3. SQL
  - Ch4. Intermediate SQL
  - Ch5. Advanced SQL

- **Section 2: Database design**
  - Ch6. Database Design: Entity-Relationship Model
  - Ch7. Relational Database Design
Section 3: Data storage, query
Ch8. Storage and File Structure
Ch9. Indexing and Hashing
Ch10. Query Processing
Ch11. Query Optimization

Section 4: Transaction Management
Ch12. Transaction
Ch13. Concurrency Control
Ch14. Recovery system

Section 5: Expand
Ch15. NoSQL
Ch16. Data mining
Teaching activity

- Lecture (assignment) 2/3
- Lab exercise- Computer experiment 1/3
- Group project
- Final exam
Evaluation

- Absence and assignment 10%
- Lab exercise 15%
- Database design in group 15%
- Final exam 60%
Chapter 1: Introduction
Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both convenient and efficient to use
- Database Applications:
  - Banking: 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 can be very large.
- Databases touch all aspects of our lives
University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems
Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation — multiple files and formats
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones
Drawbacks of using file systems to store data (Cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all

- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) 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
Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model
Relational Model

- Relational model (Chapter 2)
- Example of tabular data in the relational model

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>History</td>
<td>60000</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
<td>Elec. Eng.</td>
<td>80000</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
</tr>
<tr>
<td>58583</td>
<td>Califieri</td>
<td>History</td>
<td>62000</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
</tr>
<tr>
<td>33456</td>
<td>Gold</td>
<td>Physics</td>
<td>87000</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
<td>Finance</td>
<td>80000</td>
</tr>
</tbody>
</table>

(a) The instructor table
A Sample Relational Database

(a) The instructor table

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>History</td>
<td>60000</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
<td>Elec. Eng.</td>
<td>80000</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
</tr>
<tr>
<td>58583</td>
<td>Calieri</td>
<td>History</td>
<td>62000</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
</tr>
<tr>
<td>33456</td>
<td>Gold</td>
<td>Physics</td>
<td>87000</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
<td>Finance</td>
<td>80000</td>
</tr>
</tbody>
</table>

(b) The department table

<table>
<thead>
<tr>
<th>dept_name</th>
<th>building</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp. Sci.</td>
<td>Taylor</td>
<td>100000</td>
</tr>
<tr>
<td>Biology</td>
<td>Watson</td>
<td>90000</td>
</tr>
<tr>
<td>Elec. Eng.</td>
<td>Taylor</td>
<td>85000</td>
</tr>
<tr>
<td>Music</td>
<td>Packard</td>
<td>80000</td>
</tr>
<tr>
<td>Finance</td>
<td>Painter</td>
<td>120000</td>
</tr>
<tr>
<td>History</td>
<td>Painter</td>
<td>50000</td>
</tr>
<tr>
<td>Physics</td>
<td>Watson</td>
<td>70000</td>
</tr>
</tbody>
</table>
Data Definition Language (DDL)

- Specification notation for defining the database schema
  Example: `create table instructor (``ID`` char(5), ``name`` varchar(20), ``dept_name`` varchar(20), ``salary`` numeric(8,2))`

- DDL compiler generates a set of table templates stored in a `data dictionary`
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
    - Referential integrity (`references` constraint in SQL)
      - e.g. `dept_name` value in any `instructor` tuple must appear in `department` relation
  - Authorization
SQL

- **SQL**: widely used non-procedural language
  - Example: Find the name of the instructor with ID 22222
    ```sql
    select name
    from instructor
    where instructor.ID = '22222'
    ```
  - Example: Find the ID and building of instructors in the Physics dept.
    ```sql
    select instructor.ID, department.building
    from instructor, department
    where instructor.dept_name = department.dept_name and
    department.dept_name = 'Physics'
    ```

- 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

- Chapters 3, 4 and 5
Database Design?

- Is there any problem with this design?

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>salary</th>
<th>dept_name</th>
<th>building</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>95000</td>
<td>Physics</td>
<td>Watson</td>
<td>70000</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>90000</td>
<td>Finance</td>
<td>Painter</td>
<td>120000</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>60000</td>
<td>History</td>
<td>Painter</td>
<td>50000</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>75000</td>
<td>Comp. Sci.</td>
<td>Taylor</td>
<td>100000</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
<td>80000</td>
<td>Elec. Eng.</td>
<td>Taylor</td>
<td>85000</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>72000</td>
<td>Biology</td>
<td>Watson</td>
<td>90000</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>65000</td>
<td>Comp. Sci.</td>
<td>Taylor</td>
<td>100000</td>
</tr>
<tr>
<td>58583</td>
<td>Califieri</td>
<td>62000</td>
<td>History</td>
<td>Painter</td>
<td>50000</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>92000</td>
<td>Comp. Sci</td>
<td>Taylor</td>
<td>100000</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>40000</td>
<td>Music</td>
<td>Packard</td>
<td>80000</td>
</tr>
<tr>
<td>33456</td>
<td>Gold</td>
<td>87000</td>
<td>Physics</td>
<td>Watson</td>
<td>70000</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
<td>80000</td>
<td>Finance</td>
<td>Painter</td>
<td>120000</td>
</tr>
</tbody>
</table>
Design Approaches

- **Normalization Theory (Chapter 7)**
  - Formalize what designs are bad, and test for them

- **Entity Relationship Model (Chapter 6)**
  - 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*:
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:

What happened to dept_name of instructor and student?
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 for 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
Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- 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.
Database Users and Administrators

- naive users (tellers, agents, web users) use application interfaces
- application programmers write application programs
- sophisticated users (analysts) use query tools
- database administrators use administration tools

Database
Database Architecture

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

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed
History of Database Systems

• 1950s and early 1960s:
  • Data processing using magnetic tapes for storage
    • Tapes provided only sequential access
  • Punched cards for input

• Late 1960s and 1970s:
  • Hard disks allowed direct access to data
  • Network and hierarchical data models in widespread use
  • Ted Codd defines the relational data model
    • Would win the ACM Turing Award for this work
    • IBM Research begins System R prototype
    • UC Berkeley begins Ingres prototype
  • High-performance (for the era) transaction processing
History (cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..
End of Chapter 1