Data Management

An evolving, expanding field:

- Classical stand-alone databases (Oracle, DB2, SQL Server)
- Computer science is becoming data-centric:
  - web knowledge harvesting, crowd sourcing, cloud computing, scientific databases, networks, data mining, streaming sensor monitoring, social networks, bioinformatics, geographic information systems, digital libraries, data-driven business processes

- Classical database concepts and algorithms continue to provide the core technology → this course
CSE132A: Database Systems Principles

- Core concepts and techniques in database systems
- Databases from the viewpoint of user and designer
- A lot of SQL, but also peeks under the hood: query processing, indexing, schema design, transactions and concurrency control
- Some basic theory: formal languages underlying SQL relational algebra and relational calculus
- Basic background for follow-up courses
  - 132B: Database System Applications
  - 135: Server-Side Web Applications

Resources

- Slides, recommended texts, podcast
- Practice problem sets with solutions (ungraded)
  - Gradiance online practice homeworks and labs
  - Posted practice problems
- Weekly discussion sections (choose one of the two)
- TA/tutor daily office hours and instructor office hours
- Discussion board (Piazza)
- Everything will be posted on the class website (check often!)
  - http://cseweb.ucsd.edu/classes/wi16/cse132A-a/
Requirements

- Two Gradiance SQL Labs and two written homeworks (14%)
- Two programming assignments (SQL and JDBC) (33%)
- Midterm (25%)
- Final (25%)
- Class participation via clickers (3%)

What is a database?

- Persistent data
- Query and update language for accessing and modifying data
- Query optimization
- Transactions and concurrency control

What kind of data?

Emphasis: many instances of similarly structured data

Examples:

- Airline reservations: database (large set of similar records)
- Computerized library: information retrieval
- Medication advisor: expert system
Top Level Goals of a Database System

- Provide users with a meaning-based view of data
  - shield from irrelevant detail → abstract view

- Support operations on data
  - queries, updates

- Provide data control
  - integrity, protection
  - concurrency, recovery

Database System

- Tailored to specific application

Database Management System

- Generalized DB system
  - used in variety of application environments
  - common approach to
    - data organization
    - data storage
    - data access
    - data control
  - e.g. Ingres/Postgres, DB2, Oracle, SQL Server, MySQL, etc.
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
    • 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
• 2000s:
  – XML and XQuery standards
  – Automated database administration
  – Today: tremendous expansion across computer science
Purpose of Database Systems

“database: leap of abstraction from file systems”

• 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
  – 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 (Cont.)

• Drawbacks of using file systems (cont.)
  – No guaranteed 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 and updating it at the same time
  – Security problems
    • Hard to provide user access to some, but not all, data
Levels of Abstraction

• **Logical level**: describes data stored in database in terms close to the application
  
  ```
  type customer = record
      customer_id : string;
      customer_name : string;
      customer_street : string;
      customer_city : integer;
  end;
  ```

• **Physical level**: describes how the data is stored.

• **View level**: customized, restructured information. Views can also hide information (such as an employee’s salary) for security purposes.

**Basic Architecture of a Database System**

![Diagram of database system architecture](image)

**Data Independence** – logical and physical levels are independent
Data Models

- A collection of concepts and tools for describing the data relationships, semantics, constraints...
- A language for querying and modifying the data

- Relational model
- Entity-Relationship data model (mainly for database design, no query language)
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Schemas and Instances

Similar to types and values of 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 of a variable in a program

- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
Example: Entity-Relationship Model

- Models an application 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:*

Example: Relational Model

- *Schema*

(a) The customer table

(b) The account table

(c) The depositor table
Example: Relational Model

<table>
<thead>
<tr>
<th>customer_id</th>
<th>customer_name</th>
<th>customer_street</th>
<th>customer_city</th>
</tr>
</thead>
<tbody>
<tr>
<td>765-44-0005</td>
<td>Johnson</td>
<td>12 Ains St.</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>678-44-0004</td>
<td>Kaye</td>
<td>3 Main St.</td>
<td>Harrison</td>
</tr>
<tr>
<td>321-22-1234</td>
<td>Turner</td>
<td>123 Putnam Ave.</td>
<td>Stambord</td>
</tr>
<tr>
<td>345-66-9990</td>
<td>Jones</td>
<td>100 Main St.</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>011-28-3746</td>
<td>Lindsey</td>
<td>175 Park Ave.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Smith</td>
<td>72 North St.</td>
<td>Rye</td>
</tr>
</tbody>
</table>

(a) The customer table

<table>
<thead>
<tr>
<th>account_number</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>A-215</td>
<td>700</td>
</tr>
<tr>
<td>A-102</td>
<td>400</td>
</tr>
<tr>
<td>A-203</td>
<td>350</td>
</tr>
<tr>
<td>A-204</td>
<td>900</td>
</tr>
<tr>
<td>A-217</td>
<td>750</td>
</tr>
<tr>
<td>A-222</td>
<td>700</td>
</tr>
</tbody>
</table>

(b) The account table

<table>
<thead>
<tr>
<th>customer_id</th>
<th>account_number</th>
</tr>
</thead>
<tbody>
<tr>
<td>765-44-0005</td>
<td>A-101</td>
</tr>
<tr>
<td>678-44-0004</td>
<td>A-215</td>
</tr>
<tr>
<td>321-22-1234</td>
<td>A-102</td>
</tr>
<tr>
<td>345-66-9990</td>
<td>A-203</td>
</tr>
<tr>
<td>011-28-3746</td>
<td>A-204</td>
</tr>
</tbody>
</table>

Data Definition Language (DDL)

- Specification language 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
  - Integrity constraints
    - Domain constraints
    - Referential integrity (references constraint in SQL)
    - Assertions
  - Authorization information
Data Manipulation Language (DML)

- Language for accessing and modifying data
  DML also known as query/update language
- Two classes of languages
  - Procedural – user specifies what data is required and how to get that data
  - Declarative (nonprocedural) – user specifies what data is required without specifying how to get it
- SQL is the most widely used query language
  primarily declarative

Database Architecture

Different architectures for different settings:
- Centralized
- Parallel (multi-processor) cloud computing/map-reduce
- Client-server
- Distributed
Client/Server DBMS

SQL requests

Answers

Client Server

Map-reduce

25

26
Core database issues

- Data models, query languages
- Database design
- Query processing
- Storage management
- Transaction management
- Concurrency control

Beyond the Core

- Deductive databases
- Temporal databases
- Multimedia databases
- Geographic information systems
- Data warehouses
- Real-time and active databases
- XML databases
- Database-driven Web applications/services
- Data analytics (aka Big Data)
Databases at UCSD

• Prof. Alin Deutsch
• Prof. Yannis Papakonstantinou
• Prof. Victor Vianu

Database group Web site: http://db.ucsd.edu/
  papers, seminars, bragging….

• Intersections with other CSE groups
  – storage
  – multimedia
  – machine learning
  – IR/ data mining
  – networks

This course

1. Database Management Systems overview
2. Relational databases
   --The relational model
   --Commercial query languages: SQL (and some QBE)
   --Formal query languages: relational algebra and calculus
   --Query processing
   --Indexing (sequential files, B-trees)
   --Schema design: normal forms and the ER model
3. Concurrency control
4. Other topics, as time allows: object databases, XML