CSE 233

Database System Overview

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
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.

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

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**

<table>
<thead>
<tr>
<th>customer_id</th>
<th>customer_name</th>
<th>customer_street</th>
<th>customer_city</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>12 Aloe St.</td>
<td>Palo Alto</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Main St.</td>
<td>Harrison</td>
</tr>
<tr>
<td></td>
<td></td>
<td>175 Park Ave.</td>
<td>Pittsfield</td>
</tr>
</tbody>
</table>

**Instance**

<table>
<thead>
<tr>
<th>account_number</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-01</td>
<td>300</td>
</tr>
<tr>
<td>A-02</td>
<td>400</td>
</tr>
<tr>
<td>A-05</td>
<td>350</td>
</tr>
<tr>
<td>A-201</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>

<table>
<thead>
<tr>
<th>customer_id</th>
<th>account_number</th>
</tr>
</thead>
<tbody>
<tr>
<td>792-83-7865</td>
<td>A-191</td>
</tr>
<tr>
<td>792-83-7865</td>
<td>A-201</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>A-217</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>A-222</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
    - keys
    - foreign keys (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
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. Arun Kumar
• Prof. Yannis Papakonstantinou
• Prof. Victor Vianu

Database group Web site: https://dbucsd.github.io
papers, seminars, bragging….

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

Database Theory

CSE 233
Relational db: theory → practice

Frege: FO logic

Tarski: algebra for FO

Codd: relational databases

Databases: implemented logic!

- FO lies at the core of modern database systems
  “Databases = FO on every desk!”

- Relational query languages are based on FO:
  SQL, QBE

- More powerful query languages are based on extensions of FO
Why is FO so successful as a query language?

- **easy to use** syntactic variants
  - SQL, QBE
- **efficient implementation** via relational algebra
  - amenable to analysis and simplification
- **potential for perfect scaling** to large databases
  - very fast response can be achieved
  - using parallel processing

---

**Journey of a Query**

<table>
<thead>
<tr>
<th>SQL ~ FO</th>
<th>select … from … where</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational Algebra</td>
<td>( \pi_{13}(P \bowtie Q) \bowtie \cdots )</td>
</tr>
<tr>
<td>Query Rewriting</td>
<td>( \pi_{14}(P \bowtie S) \bowtie Q \bowtie R )</td>
</tr>
<tr>
<td>Query Execution Plan</td>
<td>Execution</td>
</tr>
<tr>
<td>Physical Level</td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

```
Q \bowtie R \Rightarrow \pi_{14}

P \bowtie S
```
Journey of a Query

SQL $\sim$ FO

select … from … where

Relational Algebra

$\pi_{13}(P\bowtie Q) \bowtie Q R$

Query Rewriting

$\pi_{14}(P\bowtie S) \bowtie Q R$

Query Execution Plan

Execution

Physical Level

Most spectacular: theoretical potential for perfect scaling!

- perfect scaling: given sufficient resources, performance does not degrade as the database becomes larger
- key: parallel processing
- cost: number of processors polynomial in the size of the database ($FO \subseteq AC0$)
- role of algebra: operations highlight parallelism
Outline

- FO (aka CALC), relational algebra
- Static analysis for query processing
- Dependency theory
- Extending FO with recursion: Datalog and fixpoint logics
- Expressiveness and complexity
  - Ehrenfeucht-Fraisse games, 0/1 laws
  - The quest for a language for PTIME
- Highly expressive languages

Other topics (if time)

- Incomplete information
- Complex objects
- Selected research topics