Overview of Database Systems

CPSC 315 – Programming Studio
Fall 2009
Project 1, Lecture 1

Project

- Your first project (next week) will involve putting together a very basic database system
- There will be a few lectures to give you an overview of database systems
- This is nowhere close to what you would get in a full database course
- *Slides adapted from Jennifer Welch (some of hers were from Jeffrey Ullman)*

Database Systems

- Systems designed to manage very large amounts of data, and to query that data to pull out useful information
- Often, key considerations include:
  - Efficiency
  - Reliability
  - Ease of access (querying, distributed)

Creating a Database

- A database *schema* determines what will be represented in the database
- This should be tightly controlled by a database manager
- Specified through a data definition language
Querying Databases

- Once database has been populated, users can query the data
- A data manipulation language controls how the user can specify queries, (and thus what types of queries are allowed)
  - SQL is probably the most well-known

Other Database Topics

- “Real” database courses include lots of other things that we’ll be ignoring here
  - More complete theory behind design
  - Query optimization
  - Efficient storage
  - Processing Transactions – grouped queries that provide atomic operations
    - Scheduling, logging, recovery

Entity-Relationship Model

- Way of expressing (in diagrammatic form) a database design
  - Kinds of data and how they connect
- Easy first way to think about databases
- Later, relational model described

Entities and Attributes

- Entities are things
- Entity sets are collections of those things
- Attributes are properties of entity sets
**Entity Sets and Attributes**

- **Senator**
  - Name
  - Party
  - State
  - Years

- **Bill**
  - Name
  - Text

**Relationships**

- Connect two or more entity sets

**Values of Relationships**

- The “value” of an entity set is the entities it contains
- The “value” of a relationship is a list of currently related entities (one from each entity set)

<table>
<thead>
<tr>
<th>Senator</th>
<th>Bill</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>Tax Bill</td>
</tr>
<tr>
<td>Smith</td>
<td>Defense Bill</td>
</tr>
<tr>
<td>Jones</td>
<td>Tax Bill</td>
</tr>
</tbody>
</table>

**Multi-Way Relationships**

- E.g. Lobbyist lobbied Senator about Bill
Relationship Types

- Consider binary relationships (two entity groups in a relationship)
- One-to-one
  - Each entity can have at most one in the other category
  - e.g. entity groups: Baseball player, Team
    - relationship: Team MVP
  - A team can only have one MVP, and a player can only be MVP for one team.

- Many-to-one
  - Each entity of first set can go to at most one of the second set
  - e.g. entity groups: Person, Town
    - relationship: BornIn
  - A person can is born in only one town, but a town can have many people born there

Relationship Types

- Consider binary relationships (two entity groups in a relationship)
- One-to-one
- Many-to-one
- Many-to-many
  - Any number from one set to the other
  - e.g. Senators can sponsor many bills, and each bill can be sponsored by many Senators

Diagrams of Relationships

- Arrow shows “to one”

[Diagram showing relationships between Person, Born In, Town, Baseball Player, MVP, Team]
Attributes on Relationships

- Can be converted to multi-way diagrams

\[ \text{Person} \xrightarrow{\text{Born In}} \text{Hospital} \xrightarrow{} \text{Town} \]

Attributes on Relationships

- Note arrows

\[ \text{Person} \xrightarrow{\text{Injured}} \text{Hospitals} \xrightarrow{} \text{Hospital} \]

Programmer Roles

- If multiple references to same entity set, label edges by roles

\[ \text{Students} \rightarrow \text{Team} \rightarrow \text{Tester} \rightarrow \text{Team Lead} \]
**Subclass**

- Fewer entities, more properties

**Keys**

- A *key* is a set of attributes for an entity set such that no two entities agree on all the attributes.
- We must have a key for every entity set.

**Key for multiple attributes**

- Must choose *one* set of attributes
Key for multiple attributes

- Must choose *one* set of attributes

Weak entity sets

- Need “help” to determine key

Design Techniques

- Avoid redundancy
  - Say the same thing two ways
Design Techniques

- Avoid redundancy
  - Say the same thing two ways

- Don’t use entity set if attribute will do
- Entity lists should either
  - Have some non-key attribute
  - Be the “many” in a many-one/many-many relationship

- Don’t overuse weak entity sets
- Usually use unique key for each entity set (e.g. UIN, SSN, VIN)
- Not always possible, though