The Relational Model
CS 186, Spring 2006, Lecture 2
R & G, Chap. 1 & 3

Administrivia I
• CS 186 IS MOVING!!!!
• Starting TUES 1/24 (next week)
  we will be in 105 NORTHGATE

Administrivia II
• Recall: Discussion Sections
  – W11-12 70 Evans
  – W 2-3 70 Evans
  – W 3-4 241 Cory
• Section on Tuesdays is Cancelled.
• Still working on approval for 3rd TA.
• Web site is getting there.
• Details on Projects, Grading, TA office
  hours, etc. available by Tuesday.
• I *will* be holding office hours today as
  scheduled: 1-2pm 687 Soda Hall

Administrivia III - Don’t Forget
• CS 186 IS MOVING!!!!
• Starting TUES 1/24 (next week)
  we will be in 105 NORTHGATE

Data Models
• A Database models some portion of the real world.
• A Data Model is link between user’s view of the world
  and bits stored in computer.
• Many models have been proposed.
• We will concentrate on the Relational Model.

Describing Data: Data Models
• A data model is a collection of concepts for describing data.
• A database schema is a description of a particular collection of data, using a given data model.
• The relational model of data is the most widely used model today.
  – Main concept: relation, basically a table with rows and columns.
  – Every relation has a schema, which describes the columns, or fields.
Levels of Abstraction

- **Views** describe how users see the data.
- **Conceptual schema** defines logical structure
- **Physical schema** describes the files and indexes used.
  - (sometimes called the ANSI/SPARC model)

Data Independence: The Big Breakthrough of the Relational Model

- A Simple Idea: Applications should be insulated from how data is structured and stored.
- **Logical data independence**: Protection from changes in logical structure of data.
- **Physical data independence**: Protection from changes in physical structure of data.
- Q: Why are these particularly important for DBMS?

Why Study the Relational Model?

- **Most widely used model currently.**
  - DB2, MySQL, Oracle, PostgreSQL, SQLServer, ...
  - Note: some “Legacy systems” use older models
    - e.g., IBM’s IMS
- **Object-oriented concepts have recently merged in**
  - *object-relational model*
    - Informix, IBM DB2, Oracle 8i
    - Early work done in POSTGRES research project at Berkeley
- **XML (semi-structured) models emerging?**

Relational Database: Definitions

- **Relational database**: a set of relations.
- **Relation**: made up of 2 parts:
  - *Schema*: specifies name of relation, plus name and type of each column.
    - E.g., Students(sid: string, name: string, login: string, age: integer, gpa: real)
  - *Instance*: a table, with rows and columns.
    - #rows = cardinality
    - #fields = degree / arity
- Can think of a relation as a set of rows or tuples.
  - i.e., all rows are distinct

Example: University Database

- **Conceptual schema**:
  - **Students**: sid: string, name: string, login: string, age: integer, gpa: real
  - **Courses**: cid: string, cname: string, credits: integer
  - **Enrolled**: sid: string, cid: string, grade: string
- **External Schema (View)**:
  - **Course_info**: cid: string, enrollment: integer
- **One possible Physical schema**:
  - Relations stored as unordered files.
  - Index on first column of Students.

Ex: An Instance of Students Relation

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

- Q: Do all values in each column of a relation instance have to be Unique?
- Q: Is “Cardinality” a schema property?
- Q: Is “Arity” a schema property?
SQL - A language for Relational DBs

- **SQL (a.k.a. “Sequel”)**, “Intergalactic Standard for Data”
  - Stands for Structured Query Language
  - Two sub-languages:
    - **Data Definition Language (DDL)**
      - create, modify, delete relations
      - specify constraints
      - administer users, security, etc.
    - **Data Manipulation Language (DML)**
      - Specify queries to find tuples that satisfy criteria
      - add, modify, remove tuples

SQL Overview

- CREATE TABLE <name> { <field> <domain>, ...
- INSERT INTO <name> {<field names>}
  VALUES {<field values>}
- DELETE FROM <name>
  WHERE <condition>
- UPDATE <name>
  SET <field name> = <value>
  WHERE <condition>
- SELECT <fields>
  FROM <name>
  WHERE <condition>

Creating Relations in SQL

- **Creates the Students relation.**
  - Note: the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

```
CREATE TABLE Students
(sid CHAR(20),
 name CHAR(20),
 login CHAR(10),
 age INTEGER,
 gpa FLOAT)
```

Table Creation (continued)

- Another example: the Enrolled table holds information about courses students take.

```
CREATE TABLE Enrolled
(sid CHAR(20),
 cid CHAR(20),
 grade CHAR(2))
```

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES ('53688', 'Smith', 'smith@ee', 18, 3.2)
```

- Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE FROM Students S
WHERE S.name = 'Smith'
```

Powerful variants of these commands are available; more later!

Keys

- Keys are a way to associate tuples in different relations
- Keys are one form of integrity constraint (IC)

```
Enrolled | Students
---------+---------
53660   | Carnatic101 | C
53666   | Reggae203   | B
53650   | Topology112 | A
53660   | History105  | B
```

- **Primary Key** ( PK )
- **Foreign Key** ( FK )

PK: sid

FK: sid
Primary Keys

- A set of fields is a superkey if:
  - No two distinct tuples can have same values in all key fields
- A set of fields is a candidate key for a relation if:
  - It is a superkey
  - No subset of the fields is a superkey
- what if >1 key for a relation?
  - one of the candidate keys is chosen (by DBA) to be the primary key.
E.g.:
  - sid is a key for Students.
  - What about name?
  - The set (sid, gpa) is a superkey.

Primary and Candidate Keys in SQL

- Possibly many candidate keys (specified using UNIQUE), one of which is chosen as the primary key.
- Keys must be used carefully!
  - "For a given student and course, there is a single grade."
- One of the candidate keys is chosen (by DBA) to be the primary key.

CREATE TABLE Enrolled

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>5366</td>
<td>Carnatic101</td>
<td>C</td>
</tr>
<tr>
<td>5366</td>
<td>Reggae203</td>
<td>B</td>
</tr>
<tr>
<td>5360</td>
<td>Topology112</td>
<td>A</td>
</tr>
<tr>
<td>5366</td>
<td>History105</td>
<td>B</td>
</tr>
<tr>
<td>53650</td>
<td>English102</td>
<td>A</td>
</tr>
</tbody>
</table>

Students

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>5366</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Foreign Keys, Referential Integrity

- Foreign key: Set of fields in one relation that is used to refer to a tuple in another relation.
  - Must correspond to the primary key of the other relation.
  - Like a logical pointer.
- If all foreign key constraints are enforced, referential integrity is achieved (i.e., no dangling references.)

Foreign Keys in SQL

- E.g. Only students listed in the Students relation should be allowed to enroll for courses.
  - sid is a foreign key referring to Students:

CREATE TABLE Enrolled

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>5366</td>
<td>Carnatic101</td>
<td>C</td>
</tr>
<tr>
<td>5366</td>
<td>Reggae203</td>
<td>B</td>
</tr>
<tr>
<td>5360</td>
<td>Topology112</td>
<td>A</td>
</tr>
<tr>
<td>5366</td>
<td>History105</td>
<td>B</td>
</tr>
<tr>
<td>53650</td>
<td>English102</td>
<td>A</td>
</tr>
</tbody>
</table>

Students

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>5366</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Enforcing Referential Integrity

- Consider Students and Enrolled; sid in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student sid is inserted? (Reject it)
- What should be done if a Students tuple is deleted?
  - Also delete all Enrolled tuples that refer to it?
  - Disallow deletion of a Students tuple that is referred to?
  - Set sid in Enrolled tuples that refer to it to a default sid?
  - (In SQL, also: Set sid in Enrolled tuples that refer to it to a special value null, denoting ‘unknown’ or ‘inapplicable’.)
- Similar issues arise if primary key of Students tuple is updated.

Integrity Constraints (ICs)

- IC: condition that must be true for any instance of the database; e.g., domain constraints.
  - ICs are specified when schema is defined.
  - ICs are checked when relations are modified.
- A legal instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances.
- If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors, too!
Where do ICs Come From?

- ICs are based upon the semantics of the real-world that is being described in the database relations.
- We can check a database instance to see if an IC is violated, but we can NEVER infer that an IC is true by looking at an instance.
  - An IC is a statement about all possible instances!
  - From example, we know name is not a key, but the assertion that sid is a key is given to us.
- Key and foreign key ICs are the most common; more general ICs supported too.

Relational Query Languages

- A major strength of the relational model: supports simple, powerful querying of data.
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
  - The key: precise semantics for relational queries.
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

The SQL Query Language

- The most widely used relational query language.
  - Current std is SQL-2003; SQL92 is a basic subset that we focus on in this class.
- To find all 18 year old students, we can write:

  ```sql
  SELECT * FROM Students S WHERE S.age=18
  ```

  To find just names and logins, replace the first line:

  ```sql
  SELECT S.name, S.login
  ```

Querying Multiple Relations

- What does the following query compute?

  ```sql
  SELECT S.name, E.cid
  FROM Students S, Enrolled E
  WHERE S.sid=E.sid AND E.grade='A'
  ```

  Given the following instance of Enrolled:

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>53831</td>
<td>Carnatic101</td>
<td>C</td>
</tr>
<tr>
<td>53831</td>
<td>Reggae203</td>
<td>B</td>
</tr>
<tr>
<td>53650</td>
<td>Topology112</td>
<td>A</td>
</tr>
<tr>
<td>53666</td>
<td>History105</td>
<td>B</td>
</tr>
</tbody>
</table>

  We get:

  ```sql
  S.name  E.cid
  -------- --------
  Smith    Topology112
  ```

Semantics of a Query

- A conceptual evaluation method for the previous query:
  1. do FROM clause: compute cross-product of Students and Enrolled
  2. do WHERE clause: Check conditions, discard tuples that fail
  3. do SELECT clause: Delete unwanted fields
- Remember, this is conceptual. Actual evaluation will be much more efficient, but must produce the same answers.

Cross-product of Students and Enrolled Instances

<table>
<thead>
<tr>
<th>sid</th>
<th>S.name</th>
<th>S.login</th>
<th>S.age</th>
<th>S.gpa</th>
<th>E.cid</th>
<th>E.grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@ics</td>
<td>18</td>
<td>3.4</td>
<td>53831 Carnatic101</td>
<td>C</td>
</tr>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@ics</td>
<td>18</td>
<td>3.4</td>
<td>53831 Reggae203</td>
<td>B</td>
</tr>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@ics</td>
<td>18</td>
<td>3.4</td>
<td>53650 Topology112</td>
<td>A</td>
</tr>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@ics</td>
<td>18</td>
<td>3.4</td>
<td>53666 History105</td>
<td>B</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@math</td>
<td>18</td>
<td>3.2</td>
<td>53831 Carnatic101</td>
<td>C</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@math</td>
<td>18</td>
<td>3.2</td>
<td>53831 Reggae203</td>
<td>B</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@math</td>
<td>18</td>
<td>3.2</td>
<td>53650 Topology112</td>
<td>A</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@math</td>
<td>18</td>
<td>3.2</td>
<td>53666 History105</td>
<td>B</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
<td>53831 Carnatic101</td>
<td>C</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
<td>53831 Reggae203</td>
<td>B</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
<td>53650 Topology112</td>
<td>A</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
<td>53666 History105</td>
<td>B</td>
</tr>
</tbody>
</table>
Queries, Query Plans, and Operators

- System handles query plan generation & optimization; ensures correct execution.

- Issues: view reconciliation, operator ordering, physical operator choice, memory management, access path (index) use, ...

```
SELECT COUNT DISTINCT (E.eid)
FROM Emp E, Proj P, Asgn A
WHERE E.eid = A.eid
AND P.pid = A.pid
AND E.loc <> P.loc
```

Structure of a DBMS

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- Each system has its own variations.
- The book shows a somewhat more detailed version.
- You will see the “real deal” in PostgreSQL.
  - It’s a pretty full-featured example
- Next class: we will start on this stack, bottom up.

Relational Model: Summary

- A tabular representation of data.
- Simple and intuitive, currently the most widely used
  - Object-relational variant gaining ground
- Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
  - Two important ICs: primary and foreign keys
  - In addition, we always have domain constraints.

- Powerful query languages exist.
  - SQL is the standard commercial one
    - DDL - Data Definition Language
    - DML - Data Manipulation Language

Administrivia IV - Don’t Forget

- CS 186 IS MOVING!!!!

- Starting TUES 1/24 (next week)
  we will be in 105 NORTHGATE