CS 2441 Database Systems

Intro to Relational Model

Summary of Conceptual Design using ER Model

- Conceptual design follows requirements analysis,
  - Yields a high-level description of data to be stored
  - Visual language – the diagram is the syntax!
- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: constraints, type, participation
- Can automate mapping of ER model to relational tables!
Getting More Concrete: Building a Database and Application

1. Start with a conceptual model
   - “On paper” using ER
   - We ignore low-level details – focus on logical representation

2. Design & implement schema
   - Design and codify (in SQL) the relations/tables
   - Do physical layout – indexes, etc.

The Relational Data Model (1970)

Originally proposed by E.F. Codd (IBM):
- Separates physical implementation from logical
- Models the data independently from how it will be used (accessed, printed, etc.)
  - Describes the data minimally and mathematically
    - A relation describes an association between data items – tuples with attributes
    - Uses standard mathematical (logical) operations over the data – relational algebra or relational calculus
Relational Model

- Vendors: IBM, Informix, Microsoft, Oracle, Sybase, etc.
- Shift from networked “records” of data to a set of tables
  - Why is this a good thing for SW developers?
- Row corresponds to relationship among values
- Table corresponds to a relation
  - mathematical definition of relation over K sets

Why Did It Take So Many Years to Implement Relational Databases?

- Codd’s original work: 1969-70
- Earliest relational database research: ~1976
- Commercial Relational DBMSs: ~mid 1980s
- Why the gap? Top 10 reasons…
  1. “You could do the same thing in other ways”
  2. “Nobody wants to write math formulas”
  3. “Why would I turn my data into tables?”
  4. “It won’t perform well”
  5. …
- What do you think?
Relational Data Model: Definitions

- **Relational database**: a set of relations or tables.
  - **Columns** of a relation are called attributes or fields
  - The number of these columns is the **arity** of the relation
  - The **rows** of a relation are called tuples
  - The number of rows is the **cardinality/size**
  - Each attribute has values taken from a **domain**, e.g., name has domain **string**

Relational Model: Definition

- Formally, a table is a relation over K sets (domains)
  - R is a subset of \( D_1 \times D_2 \ldots \times D_K \)
  - Tuple= \((t_1,t_2,\ldots,t_k)\), where \(t_i\) is an element from domain/set \(D_i\)

- A database is a collection of relations
- Theoretically: a relation is a set of tuples; no tuple can occur more than once
  - **Real systems may allow duplicates for efficiency or other reasons – we’ll ignore this for now**
Conceptual Design for mini-banner:

“Who’s taking what, and what grade do they expect?”

![Diagram of entity-relationship model for mini-banner]

Example Instance – Schema

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>Takes</th>
<th>COURSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>name</td>
<td>sid</td>
</tr>
<tr>
<td>1</td>
<td>Ross</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Lee</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Emily</td>
<td>3</td>
</tr>
</tbody>
</table>

- Our focus now: relational schema – set of tables
- Can have other kinds of schemas – XML, object, …

<table>
<thead>
<tr>
<th>PROFESSOR</th>
<th>Teaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>fid</td>
<td>name</td>
</tr>
<tr>
<td>1</td>
<td>Wood</td>
</tr>
<tr>
<td>2</td>
<td>Heller</td>
</tr>
<tr>
<td>8</td>
<td>Narahari</td>
</tr>
</tbody>
</table>
Describing Relations: Schema

- A schema can be represented many ways
  - a schema is like a type definition in a program
- DBMS’ use data definition language (DDL) – programming language like “type definitions”
- In relational DBs, we use `relation(attribute domain,…)`
  - Relational DBMSs have very limited “built-in” domains: either tables or scalar attributes – int, string, byte sequence, date, etc.

```
STUDENT (sid int, name string)
Takes (sid int, exp-grade char[2], cid string)
COURSE (cid string, subj string, sem char[3])
Teaches (fid int, cid string)
PROFESSOR (fid int, name string)
```
Domains and schemas are one form of constraint on a valid data instance. Need to capture other business rules:
- Only “valid” students can take a course
- Student must have a name – cannot be a null string
- Only a professor must teach a course
- Student ID must be unique – only one student per ID
- If student ID appears in the Takes table, then that student must exist in the Students table
- ……

Business rules ensure integrity of the data.

Integrity Constraints

Key constraints:
- Subset of fields that uniquely identifies a tuple, and for which no subset of the key has this property
- May have several candidate keys; one is chosen as the primary key
- A superkey is a subset of fields that includes a key

Inclusion dependencies (referential integrity constraints):
- A field in one relation may refer to a tuple in another relation by including its key
- The referenced tuple must exist in the other relation for the database instance to be valid
  - Student enrolled in a course MUST appear in the students table

IMPORTANT!!!! Once specified in the schema, the DBMS enforces the constraints.
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.

Relational Query Languages

- **Query languages**: Allow manipulation and *retrieval of data* from a database.
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic.
  - Allows for much optimization.
- Query Languages ≠ programming languages!
  - QLs not expected to be “Turing complete”.
  - QLs not intended to be used for complex calculations.
  - QLs support easy, efficient access to large data sets.
Formal Query Languages

- Formal query languages are defined as mathematical operators over the set
  - What is the advantage of a formal language?
  - Relational algebra, Relational calculus are examples
- Procedural vs Non-procedural languages
  - Can have a mix in practice
  - Relational algebra: procedural language
  - Relational calculus: non-procedural (declarative)

SQL: Structured Query Language

The standard language for relational data
  - Invented by folks at IBM, esp. Don Chamberlin
  - Actually not a great language…
  - Beat a more elegant competing standard, QUEL, from Berkeley

Separated into a DML & DDL

SQL DML component based on relational algebra & calculus
SQL

- components
  - Data definition (DDL) – to define schema/tables
  - Manipulation/query (DML) – for queries
  - Transaction control – to specify a transaction
  - Index – to specify storage and indexing schemes
  - Authorization- for access control/security
    - We will cover the DDL and query part of SQL first
    - Shall return to the other components after we cover those topics

Glimpse into SQL Query Language: DML

- The most widely used relational query language. Current standard is SQL-92.
- SELECT clause
  - What attributes you want
  - What relations/tables to search
  - What condition/predicate to apply
The SQL Query Language

- 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
```

<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@ee</td>
<td>18</td>
<td>3.2</td>
</tr>
</tbody>
</table>

DDL and Constraint specifications

- Specifying schema/table
- Specifying constraints
Creating Relations in SQL

- Creates the Students relation. Observe that 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: REAL)

As another example, the Enrolled table holds information about courses that students take.

- Is sid same field in the two tables??

CREATE TABLE Enrolled
(sid: CHAR(20),
cid: CHAR(20),
grade: CHAR(2))
Domain Constraints SQL

- Name should not be NULL
- Other constraints…later
  - Age > 10...

CREATE TABLE Students
  (sid CHAR(20),
   name: CHAR(20) NOT NULL,
   login CHAR(10),
   age INTEGER,
   gpa: REAL,
   CHECK (age > 10) )

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.
- Why is this useful
- 
  - If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors, too!
- Think of the constraints as the business rules derived from the application
Where do ICs Come From?

- ICs are based upon the semantics of the real-world enterprise that is being described in the database relations.
  - Need to carefully analyze the application before reaching a conclusion on the Integrity Constraints!
- 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.
  - Shall return to these after we cover DML aspect of SQL

Primary Key Constraints

- Every relation must have a key
- A set of fields is a key for a relation if:
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key.
    - Part 2 false? A superkey.
    - If there’s >1 key for a relation, one of the keys is chosen (by DBA) to be the primary key.
- E.g., what is a key for Students relation?
- 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.  

    CREATE TABLE Enrolled1 
    (sid CHAR(20), 
    cid CHAR(20), 
    grade CHAR(2), 
    PRIMARY KEY (sid,cid) ) 

    CREATE TABLE Enrolled2 
    (sid CHAR(20), 
    cid CHAR(20), 
    grade CHAR(2), 
    PRIMARY KEY (sid), 
    UNIQUE (cid, grade) )

Any difference between these two tables?

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 primary key of the second relation.) Like a `logical pointer'.
- In Enrolled table – sid is a student, what can we say about the students table?
- sid is a foreign key referring to Students:
  - Enrolled(sid: string, cid: string, grade: string)
  - If all foreign key constraints are enforced, referential integrity is achieved, i.e., no dangling references.
**Foreign Keys in SQL**

- Only students listed in the Students relation should be allowed to enroll for courses.

```sql
CREATE TABLE Enrolled
    (sid CHAR(20), cid CHAR(20), grade CHAR(2),
     PRIMARY KEY (sid,cid),
     FOREIGN KEY (sid) REFERENCES Students )
```

- **Students**
<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>

- **Enrolled**
<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jazz101</td>
<td>C</td>
</tr>
<tr>
<td>53666</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>

---

**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 nonexistent student id 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 if primary key of Students tuple is updated.
Referential Integrity in SQL

- SQL/92 and SQL:1999 support all 4 options on deletes and updates.
  - Default is NO ACTION (delete/update is rejected)
  - CASCADE (also delete all tuples that refer to deleted tuple)
  - SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid)
REFERENCES Students
  ON DELETE CASCADE
  ON UPDATE SET DEFAULT)

Referential Integrity: cyclical references?

- Can have a cyclical reference pattern
  - Table A references table B, and table B references table A
  - How do we deal with foreign key constraint?
- Person(ssn, name, spouse)
  - Spouse is also a person
    - Foreign key referencing Person
- Allow NULL in Foreign key attribute
  - Foreign key constraint actually says that if a non-NULL value appears in the attribute of A then it must appear in table B.
How to design a good schema?

- What do we mean by a “good design”?  
  - Can we quantify it?
- Shall revisit this problem after finishing SQL  
  - Theory of Normalization and Normal forms

Summary - Relational Model:  
Advantages

- Data driven not design driven  
  - designed once; data changes over time without affecting applications
- data stored, read, modified from “one” location
- rules/constraints control how data defined and enforced
- changes to database scheme without affecting application?
Next: Formal Query Languages

- Formal query languages
  - We focus mainly on relational algebra
    - This is what we will need when we discuss query optimization
  - Will glance at relational calculus
    - A declarative language based on calculus…Datalog