The Relational Model

Chapter 3

ECS 165A – Winter 2022

Mohammad Sadoghi

Exploratory Systems Lab
Department of Computer Science

UCDAVIS
UNIVERSITY OF CALIFORNIA

ExpoLab
Creativity Unfolded

ResilientDB

Why Study the Relational Model?

❖ Most widely used model.
- ▪ Vendors: IBM, Microsoft, Oracle, etc.

❖ "Legacy systems" in older models
- ▪ E.G., IBM's IMS

❖ Old competitors:
- ▪ Hierarchical Model
- ▪ Network Model

❖ Competitors: object-oriented model

❖ *Object-relational model*

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 Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

❖ Cardinality = 3, degree = 5, all rows distinct

❖ Do all columns in a relation instance have to be distinct?

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

❖ Developed by IBM (system R) in the 1970s

❖ Need for a standard since it is used by many vendors

❖ Standards:

- SQL-86

- SQL-89 (minor revision)

- SQL-92 (major revision)

- SQL-99 (major extensions)

- SQL:2003, SQL:2006, SQL:2008, SQL:2011

Example: University Database

❖ 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)*

The SQL Query Language

❖ To find all 18 year old students, we can write:

SELECT *
  FROM Students S
WHERE S.age=18

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

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

SELECT S.name, S.login

Querying Multiple Relations

❖ What does the following query compute?

Given the following instances of Enrolled and Students:

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

SELECT S.name, E.cid

FROM Students S, Enrolled E

WHERE S.sid=E.sid AND E.grade="A"

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

we get:

S.name	E.cid
Smith	Topology112

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.

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

CREATE TABLE Students
        (sid: CHAR(20),
        name: CHAR(20),
        login: CHAR(10),
        age: INTEGER,
        gpa: REAL)

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

Removing and Altering Relations

DROP TABLE Students

❖ Removes the relation Students. The schema information *and* the tuples are deleted.

ALTER TABLE Students
        ADD COLUMN firstYear: integer

❖ The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a *null* value in the new field.

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'

Integrity Constraints (ICs)

- ❖ IC: condition that must be true for *any* instance of the database; e.g., *domain constraints.*
  - ▪ ICs are <u>specified</u> when schema is defined.
  - ▪ ICs are <u>checked</u> 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!

Primary Key Constraints

❖ A set of fields is a *<u>key</u>* for a relation if

no two distinct tuples (records) can have same values in all key fields, and

Primary and Candidate Keys in SQL

❖ Possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*.

❖ "For a given student and course, there is a single grade."

❖ "Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade."

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

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

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'.

❖ E.g. *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.
- Can you name a data model w/o referential integrity?
  - Links in HTML!

Foreign Keys in SQL

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

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

Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8
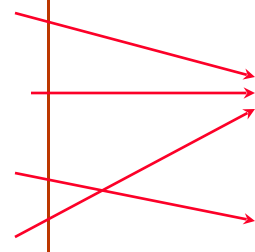
Enforcing Referential Integrity

❖ 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 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)

Views

❖ A *view* is just a relation, but we store a *definition*, rather than a set of tuples.

> CREATE VIEW YoungActiveStudents (name, grade)
> AS SELECT S.name, E.grade
> FROM Students S, Enrolled E
> WHERE S.sid = E.sid and S.age<21

❖ Views can be dropped using the DROP VIEW command.
- How to handle DROP TABLE if there's a view on the table?
  - DROP TABLE command has options to let the user specify this.

Views and Security

❖ Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).

- Given YoungStudents, but not Students or Enrolled, we can find students s who have are enrolled, but not the *cid's* of the courses they are enrolled in.

Relational Model: Summary

❖ A tabular representation of data.

❖ Simple and intuitive, currently the most widely used.

❖ 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 and natural query languages exist.