

The Relational Model

Zaki Malik

August 28, 2008

Course Outline

- Weeks 1–5, 13: Query/Manipulation Languages
 - The Relational Model
 - Relational Algebra
 - SQL Basics
 - Data Definition
 - Programming with SQL
- Weeks 6–8: Data Modeling
- Weeks 9–13: Relational Design
- Week 14–15: Students' choice

Data Model

- A Data Model is a notation for describing data or information. Consists of three parts:
 - Structure of data (e.g. arrays, structs)
 - Conceptual model: In databases, structures are at a higher level.
 - Operations on data (Modifications and Queries)
 - Limited Operations: Ease of programmers and efficiency of database.
 - Constraints on data (what the data can be)
- Two Important Data Models Today
 - The Relational Model
 - The Semistructured-Data Model
 - XML and related standards

The Relational Model

- Simple: Built around a single concept for modeling data: the relation or table.
 - A relational database is a collection of **relations**.
 - Each relation is a **table** with rows and columns.
- Supports high-level programming language (SQL).
 - Limited but very useful set of operations
 - Enables programmers to express their wishes at a very high level
- Has an elegant mathematical design theory.
- Most current DBMS are relational.

Relations

- A relation is a two-dimensional table:
 - Relation \approx table.
 - Attribute \approx column name.
 - Tuple \approx row (not the header row).
- Database \approx collection of relations.
- A relation has two parts:
 - **Schema** defines column heads of the table (attributes).
 - **Instance** contains the data rows (tuples, rows, or records) of the table.

Student	Course	Grade
Hermione Grainger	Potions	A-
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

Schema

CoursesTaken :

Student	Course	Grade
Hermione Grainger	Potions	A-
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

- The schema of a relation is the name of the relation followed by a parenthesized list of attributes.

`CoursesTaken(Student, Course, Grade)`

- A design in a relational model consists of a set of schemas.
- Such a set of schemas is called a relational database schema.

Equivalent Representations of a Relation

CoursesTaken :

Student	Course	Grade
Hermione Grainger	Potions	A-
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

CoursesTaken(Student, Course, Grade)

- Relation is a set of tuples and not a list of tuples.
 - Order in which we present the tuples does not matter.

- The attributes in a schema are also a set (not a list).
 - Schema is the same irrespective of order of attributes.

CoursesTaken(Student, Grade, Course)

- We specify a “standard” order when we introduce a schema.
- How many equivalent representations are there for a relation with m attributes and n tuples? $m! n!$

Degree and Cardinality

CoursesTaken :

Student	Course	Grade
Hermione Grainger	Potions	A-
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

- **Degree** is the number of fields/attributes in schema (=3 in the table above)
- **Cardinality** is the number of tuples in relation (=4 in the table above)

Example

- Create a database for managing class enrollments in a single semester. You should keep track of all students (their names, Ids, and addresses) and professors (name, Id, department). Do not record the address of professors but keep track of their ages. Maintain records of courses also. Like what classroom is assigned to a course, what is the current enrollment, and which department offers it. At most one professor teaches each course. Each student evaluates the professor teaching the course. Note that all course offerings in the semester are unique, i.e. course names and numbers do not overlap. A course can have ≥ 0 pre-requisites, excluding itself. A student enrolled in a course must have enrolled in all its pre-requisites. Each student receives a grade in each course. The departments are also unique, and can have at most one chairperson (or dept. head). A chairperson is not allowed to head two or more departments.

Example

- Create a database for managing class enrollments in a single semester. You should keep track of all **students** (their names, Ids, and addresses) and **professors** (name, Id, department). Do not record the address of professors but keep track of their ages. Maintain records of **courses** also. Like what classroom is assigned to a course, what is the current enrollment, and which department offers it. At most one professor **teaches** each course. Each student **evaluates** the professor teaching the course. Note that all course offerings in the semester are unique, i.e. course names and numbers do not overlap. A course can have ≥ 0 **pre-requisites**, excluding itself. A student enrolled in a course must have enrolled in all its pre-requisites. Each student receives a **grade** in each course. The **departments** are also unique, and can have at most one chairperson (or dept. head). A chairperson is not allowed to head two or more departments.

Relational Design for the Example

- Students (PID: *string*, Name: *string*, Address: *string*)
- Professors (PID: *string*, Name: *string*, Office: *string*, Age: *integer*, DepartmentName: *string*)
- Courses (Number: *integer*, DeptName: *string*, CourseName: *string*, Classroom: *string*, Enrollment: *integer*)
- Teach (ProfessorPID: *string*, Number: *integer*, DeptName: *string*)
- Take (StudentPID: *string*, Number: *integer*, DeptName: *string*, Grade: *string*, ProfessorEvaluation: *integer*)
- Departments (Name: *string*, ChairmanPID: *string*)
- PreReq (Number: *integer*, DeptName: *string*, PreReqNumber: *integer*, PreReqDeptName: *string*)

Issues to Consider in the Design

- Can we merge Courses and Teach since each professor teaches at most one course?
- Do we need a separate relation to store evaluations?
- How can we handle pre-requisites that are “or”s, e.g., you can take CS 4604 if you have taken either CS 2604 or CS 2606?
- How do we generalize this schema to handle data over more than one semester?
- What modifications does the schema need if more than one professor can teach a course?

SQL and Relational Algebra

What is SQL

- SQL = Structured Query Language (pronounced “sequel”).
- Language for defining as well as querying data in an RDBMS.
- Primary mechanism for querying and modifying the data in an RDBMS.
- SQL is declarative:
 - Say what you want to accomplish, without specifying how.
 - One of the main reasons for the commercial success of RDBMSs.
- SQL has many standards and implementations:
 - ANSI SQL
 - SQL-92/SQL2 (null operations, outerjoins)
 - SQL-99/SQL3 (recursion, triggers, objects)
 - Vendor-specific variations.

Relational Algebra

- Relational algebra is a notation for specifying queries about the contents of relations.
- Notation of relational algebra eases the task of reasoning about queries.
- Operations in relational algebra have counterparts in SQL.

What is an Algebra?

- An algebra is a set of operators and operands.
 - Arithmetic: operands are variables and constants, operators are $+, -, \times, \div, /$, etc.
 - Set algebra: operands are sets and operators are $\cap, \cup, -$
- An algebra allows us to construct expressions by combining operands and expression using operators and has rules for reasoning about expressions.
 - $a^2 + 2 \times a \times b + 2b, (a + b)^2$.
 - $R - (R - S), R \cap S$.

Basics of Relational Algebra

- Operands are relations, thought of as sets of tuples.
- Think of operands as variables, whose tuples are unknown.
- Results of operations are also sets of tuples.
- Think of expressions in relational algebra as queries, which construct new relations from given relations.
- Four types of operators:
 - **Select/Show parts of a single relation**: projection and selection.
 - Usual **set operations** (union, intersection, difference).
 - **Combine the tuples of two relations**, such as cartesian product and joins.
 - **Renaming**.

Projection

- The projection operator produces from a relation R a new relation containing only some of R's columns.
- “Delete” (i.e. not show) attributes not in projection list.
- Duplicates eliminated
- To obtain a relation containing only the columns A_1, A_2, \dots, A_n of R

RA: $\pi_{A_1, A_2, \dots, A_n}(R)$

SQL: `SELECT A1, A2, ... An FROM R;`

Projection Example

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$\pi_{sname, rating}(S2)$

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$\pi_{age}(S2)$

age
35.0
55.5

Selection

- The selection operator applied to a relation R produces a new relation with a subset of R's tuples.
- The tuples in the resulting relation satisfy some condition C that involves the attributes of R.
 - with duplicate removal

RA: $\sigma_C(R)$

SQL: `SELECT *FROM R WHERE C;`

- The WHERE clause of a SQL command corresponds to $\sigma()$.

Selection: Syntax of Conditional

- Syntax of conditional (C): similar to conditionals in programming languages.
 - Values compared are constants and attributes of the relations mentioned in the FROM clause.
 - We may apply usual arithmetic operators to numeric values before comparing them.
- RA** Compare values using standard arithmetic operators.
- SQL** Compare values using =, <>, <, >, <=, >=.

Selection Example

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$\sigma_{rating > 8}(S2)$

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

$\pi_{sname, rating}(\sigma_{rating > 8}(S2))$

sname	rating
yuppy	9
rusty	10

Combining Operators