CSEN 501 – CSEN501 - Databases I

Lecture 3:
The Relational Model

Prof. Dr. Slim Abdennadher
Dr. Nada Sharaf

German University Cairo, Faculty of Media Engineering and Technology

Design Steps

Ideas $\Rightarrow$ E/R Design $\Rightarrow$ Relational Schema $\Rightarrow$ Relational DBMS

The Relational Data Model (1970)

Lessons from the Codd paper: A Relational Model for Data for Large Shared Data Banks

- Let us separate physical level from conceptual level
- Model the data independently from how it will be used (accessed, printed, etc.)
  - Describe the data mathematically
    - A relation describes an association between data items - tuples with attributes
    - We generally think of tables and rows, but that's somewhat imprecise
  - Use standard mathematical (logical) operations over the data - these are the relational algebra or relational calculus

Outline

- Relational Model
- Constraints
- Mapping ERD to RM Schema
- Mapping EERD to RM Schema

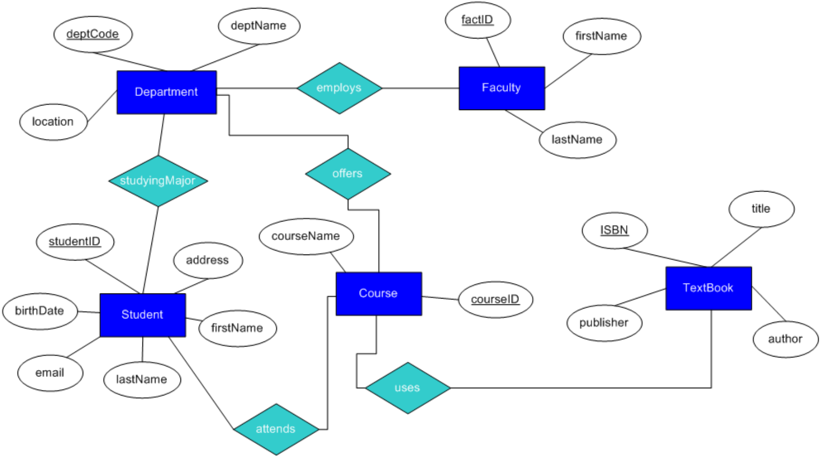
Why Did It Take So Many Years to Implement RDs?

- Codd's original work: 1969-70
- Earliest relational database research: 1976
- **System R** developed by IBM researchers during the late 1970s.
- **Oracle** developed in 1979 using many of the System R results
- **Why the gap?**
  - "You could do the same thing in other ways"
  - "Nobody wants to write math formulas"
  - "Why would I turn my data into tables?"
  - "It won't perform well"

Getting More Concrete: Building a Database and Application

- Start with a conceptual model
  - “On paper” using certain techniques, e.g. E/R diagrams
  - We ignore **low-level details** - focus on logical representation
- Design and implement schema
  - Design and codify (in SQL) the relations
  - Think about the physical level - indexes, etc.
- Import the data
- Write applications using DBMS and other tools
  - Many of the hard problems are taken care of by other people (DBMS, API writers, library authors, web server, etc.)

Conceptual Design for a University Database



What is a Relational Database?

- The **relational model** is based on the concept of a **relation**.
- A relation is physically represented as a **table** or two-dimensional array.
- Tables are used to hold information about objects to be represented in the database.
- Using the terms of EER model, both entity sets and relationships sets are shown using tables.

Student:

sid	firstName	gpa
1	Dina	1.0
2	Ahmed	2.0
3	Maria	0.7

Enroll:

sid	grade	cid
1	A	501
1	A	502
3	C	401

Course:

cid	subj	sem
501	DB	W05
502	TC	W05
401	CP	S05

- Our focus now is the relational schema: set of **tables**
- Can have other kinds of schemas - XML, object, . . .
- **Question**: Can a table have duplicate rows?

Some Terminology I

- A **relation** is represented as a two-dimensional table with columns and rows.
- Columns are called **attributes** or **fields**. The number of these columns is the **arity** (**degree**) of the relation
- The rows are called **tuples**. The number of rows is called **cardinality**.
- Each attribute has values taken from a **domain**, e.g., subj has domain string
  Domains: string, integer, real, date, . . . - atomic types
- Theoretically: a relation is a **set of tuples**; no tuple can occur more than once
- Mathematicians define a relation to be a subset of a **Cartesian product** of a list of domains.
- **Example** $D_1 = \{1, 2\}$ and $D_2 = \{a, b, c\}$
  $D_1 \times D_2 = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$

Some Terminology II

- What are the cardinality and arity of relation Student?

Student

sid	firstName	gpa
1	Dina	1.0
2	Ahmed	2.0
3	Maria	0.7
4	Ali	1.3

- Can we put a Student tuple and a Course tuple in the same relation? If not how to prevent this?
- In other words, how to **describe** relations?

Describing Relations

- A relation is defined by a **schema**, which specifies domains of each field in the relation.
- In DBMS, **data definition language** (DDL) is used - like programming language type definitions
- In relational DBs, we use **relation(attribute:domain)**

```
Student(sid:int, name:string, gpa:real)
Enroll(sid:int, grade:string, cid:string)
Course(cid:string, subj:string, sem:string)
Teaches(fid:int, cid:string)
Professor(fid:int, name:string)
```

- Database people, when they are discussing design, often assume domains are evident to the reader:

```
Student(sid, name, gpa)
```

Definitions: Schema, Relational database

- A relation (instance) is a table (a set of tuples).
- A schema is the structure of the table together with a specification of the domains and any other restrictions on possible values.
- A (relational) database schema is a collection of schemas.
- A relational database (instance) is a collection of tables, each has a distinct name.

Integrity Constraints (ICs)

- **Integrity Constraint** (IC): condition that must be true for any instance of the database.
- Example: a schema specifies domains of the fields in the relation, and is called domain constraints.
- ICs are specified when schema is defined.
  Note: this might not be true in Web databases.
- ICs are checked when relations are modified.
- A legal instance of relation is one that satisfies all specific ICs.
- DBMS does not allow illegal instances.
  - If the DBMS checks ICs, stored data is more faithful to real world meaning, that is, for consistency and accuracy.
  - When ICs are enforced, DBMS also avoids data entry errors.

Key Constraints: Entity Integrity

- A set of fields is a (candidate) key for a relation if
  - no two distinct tuples can have the same values in all key fields,
  - any proper subset of the key does not satisfy this condition.
- If a set of fields satisfies condition 1 but does not satisfy condition 2, then it is called a super key for the relation.
- If there are more than one candidate keys for a relation, one is chosen to be the primary key.
- Given a relation schema, the primary keys of the schema is underlined .

  `Student(sid:int, name:string, gpa:real)`
- **Entity Integrity**: in a relation no attribute of a primary key can have a null value.

Key Constraints: Foreign Key Constraint

- **Foreign key**: A set of fields in one relation that is used to refer to a tuple in another relation. This set must correspond to the primary key of the second relation. Example: sid is a foreign key in Enroll referring to Student

```
Student(sid:int, name:string, gpa:real)
Enroll(sid:int, grade:string, cid:string)
       -------
```

- Foreign key constraints: Let S_1 be a foreign key in relation R_1 referring to R_2. Then the values of S_1 in R_1 must match (be a subset of) the values of S_1 in R_2.

Foreign Keys - Referential Integrity

- Question: Does the following satisfy the foreign key constraint?

Student:

sid	firstName	gpa
1	Dina	1.0
2	Ahmed	2.3
3	Maria	0.7

Enroll:

sid	grade	cid
1	A	501
1	A	502
5	C	202

- **Referential Integrity**: the database does not include any invalid foreign key value. That is, all foreign key constraints are enforced.
- Suppose Table B has a foreign key that points to a field in Table A. Referential integrity would prevent you from adding a record to Table B that does not exist in Table A.
- The referential integrity rules might also specify that whenever you delete a record from Table A, any records in Table B that are linked to the deleted record will also be deleted. This is called **cascading delete**.

Representing Relational Database Schemas

- A relational database schema can have any number of relations.
- Relation schemas can be represented by giving the name of each relation followed by the attribute names in parentheses with the primary key underlined.
- The clearest way of showing foreign keys is by drawing arrows from the foreign keys to the primary keys they refer to.
- Some attributes appear in more than one relation.

```
Student(studID, lastName, firstname, major)
Enroll(studID,classNo,grade)
```

- To distinguish between the two appearances of attributes, we use the relation name followed by a period followed by the attribute name.

```
Student.studID                Enroll.studID
```

- When an attribute appears in more than one relation, its appearance usually represents a relationship or interaction between tuples of the two relations.
- Common attributes play important role in data manipulation.

Relational Data Manipulation Languages

- Variety languages used by relational database management systems
- Procedural languages: The user tells the system how to manipulate the data, e.g. Relational Algebra
- Declarative languages: the user states what data is needed but not exactly how it is to be located, e.g. Relational Calculus and SQL
- Graphical languages: allowing the user to give an example or an illustration of what data should be found, e.g. QBE

Summary - What you should remember!

- What is a relational database? Tuple? Attribute? Field? Domain? Schema? Database schema? Instance?
- What are integrity constraints? Why study them? How to decide whether an instance is legal? Where do ICs come from? What integrity constraints are a must for relational databases?
- Questions: Are the following claims correct?
  - Every relation has at least one key.
  - Every relation has one and only one primary key.
  - Every relation has one and only one superkey.