The Relational Model

Nosayba El-Sayed (based on slides from Prof. Diane Horton)
Fall 2015

UNIVERSITY OF TORONTO

DCS50

Recap

- The relational model is based on the concept of a relation or table.
- Two example relations:

Teams

Name	Home Field	Coach
Rangers	Runnymede CI	Tarvo Sinervo
Ducks	Humber Public	Maeve Mahar
Choppers	High Park	Tom Cole

Games

Home team	Away team	Home goals	Away goals
Rangers	Ducks	3	0
Ducks	Choppers	1	1
Rangers	Choppers	4	2
Choppers	Ducks	0	5

Relations in Math

- A domain is a set of values.
- Suppose D_1, D_2, ... D_n are domains.
  - The Cartesian product $D_1 \times D_2 \times \ldots \times D_n$ is the set of all tuples $\langle d_1, d_2, \ldots, d_n \rangle$ such that $d_1 \in D_1$, $d_2 \in D_2$, ..., $d_n \in D_n$.
  - I.e., every combination of a value from D_1, a value from D_2 etc.
- A (mathematical) relation on D_1, D_2, ... D_n is a subset of the Cartesian product.

Example!

- Example of a mathematical relation
  - Let A = {p, q, r, s}, B = {1, 2, 3} and C = {100, 200}.
  - R = {<q, 2, 100>, <s, 3, 200>, <p, 1, 200>} is a relation on A, B, C.
- Our database tables are *relations* too.
- Example

Games

Home team	Away team	Home goals	Away goals
Rangers	Ducks	3	0
Ducks	Choppers	1	1
Rangers	Choppers	4	2
Choppers	Ducks	0	5

Example!

- Example of a mathematical relation
  - Let A = {p, q, r, s}, B = {1, 2, 3} and C = {100, 200}.
  - R = {<q, 2, 100>, <s, 3, 200>, <p, 1, 200>}
    is a relation on A, B, C.
- Our database tables are *relations* too.
- Example

  {<Rangers, Ducks, 3, 0>, <Ducks, Choppers, 1, 1>,

  <Rangers, Choppers, 4, 2>, <Choppers, Ducks, 0, 5>}

Relation schemas vs instances

- Schema: definition of the *structure* of the relation.
- Example:
  - -Teams have 3 *attributes*: name, home field, coach.
  - -No two teams can have the same name.
- Notation for expressing a relation's schema

  Teams(Name, HomeField, Coach)

- Instance: particular data in the relation.
- Instances *change* constantly; schemas rarely.
- Conventional databases store the current version of the data. Databases that record the history are called *temporal* databases.

Terminology

Teams

Name	Home Field	Coach
Rangers	Runnymede CI	Tarvo Sinervo
Ducks	Humber Public	Maeve Mahar
Choppers	High Park	Tom Cole
Crullers	WTCS	Anna Liu

- relation (table)
- attribute (column)
  Optionally, we can specify that attributes have domains; like types in a programming language
- tuple (row)
- arity of a relation: number of attributes (columns)
- cardinality of a relation: number of tuples (rows)

Relations are sets

- A relation is a *set* of tuples, which means:
  - there can be no duplicate tuples
  - order of the tuples doesn't matter

- In another model, relations are bags — a generalization of sets that allows duplicates.
- Commercial DBMSs use this model.
- But for now, we will stick with relations as sets.

Database schemas and instances

- Database schema: a set of relation schemas
- Database instance: a set of relation instances

Superkeys

- Superkey: a set of one or more attributes whose *combined* values are unique:
  - I.e., no two tuples can have the same values on all of these attributes.
- Example:
  - A relation called Course, with attributes: department code, course number, and course name.
  - One tuple might be <"csc", "343", "Introduction to Databases">
  - What is a superkey for this relation?
- Questions..
  - Does every relation have a superkey? (Yes)
  - Can a relation have more than one superkey? (Yes)

Keys

- Key: a *minimal* superkey.
  - I.e., you may not remove an attribute from a key, and still have a set of attributes whose combined values are unique.

- What is a key for the Course relation?

  Course(deptCode, cNumber, cName)

- Can a relation have more than one key?

  Student(student#, UTORid, surname, firstname, gpa)

- We underline attributes in the schema to indicate that they form a key.

  Teams(Name, HomeField, Coach)

- Aside: Called "superkey" because it is a *superset* of some key. (Not necessarily a proper superset.)

Coincidence vs key

- If a set of attributes is a key for a relation:
  - It does not mean merely that "there are no duplicates" in a particular instance of the relation
  - It means that in principle there *cannot* be any.
- Often we have to invent an artificial new attribute to ensure all tuples will be unique.
  - This predates databases. E.g., SIN, ISBN number.
- A key is a kind of integrity constraint.

Example: Movies schema

References between relations

- Relations often *refer* to each other.
- Example:
  In the Roles relation, the tuple about Han Solo needs to say he is played by Ford.
- Rather than repeat information already in the Artists table, we store Ford's key.

Artists

aID	aName	nat
1	Nicholson	American
2	Ford	American
3	Stone	British
4	Fisher	American

Roles

mID	aID	character
1	1	Jack Torrance
3	1	Jake 'J.J.' Gittes
1	3	Delbert Grady
5	2	Han Solo
6	2	Bob Falfa
5	4	Princess Leia Organa

Q! If aID is a key for Artists, does that mean a particular aID can appear only once in Roles?? (No)

Foreign keys

- The *referring* attribute is called a foreign key because it refers to an attribute that is a key in another table.
- This gives us a way to refer to a single tuple in that relation.
- A foreign key may need to have *several* attributes.

Declaring foreign keys

- We use this notation to express relationships between attribute values in two relations:

  $R_1[X] \subseteq R_2[Y]$

- Example: Roles[aID] ⊆ Artists[aID]
- R[A] notation:
  - R is a relation and
    A is a list of attributes in R.
  - R[A] is the set of all tuples from R,
    but with only the attributes in list A.

Foreign keys in the Movies schema

Referential integrity constraints

- These $R_1[X] \subseteq R_2[Y]$ relationships are called referential integrity constraints or inclusion dependencies.
- Not all referential integrity constraints are *foreign key* constraints.
- For example, we could say

  Artists[aID] ⊆ Roles[aID]
  Movies[length] ⊆ Roles[mID]

- In these cases, we are not referring to a unique tuple.
- $R_1[X] \subseteq R_2[Y]$ is a foreign key constraint iff Y is a key for relation R2.

Designing a schema

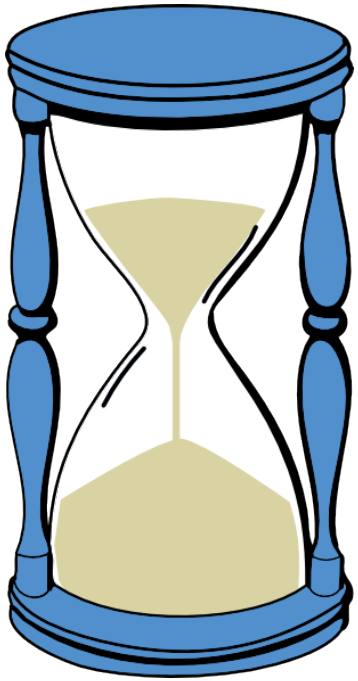
- Mapping from the real world to a relational schema is surprisingly challenging and interesting.
- There are always many possible schemas.
- Two important goals:

  1- Represent the data well.
  For example, avoid constraints that prevent expressing things that occur in the domain.

  2- Avoid redundancy.

- Later, we'll learn some elegant theory that provides sound principles for good design.

Solve the questions on the back of Movies handout..



What's next...

- We will learn how to use SQL to
  - define a database's structure,
  - put data in it, and
  - write queries on it.

- First we'll learn how to write queries in relational algebra.
  - Relational algebra is the foundation for SQL.
  - Other important concepts, like query optimization, are defined in terms of RA.

Don't forget your To-Dos ;-)

- Anyone new to the cdf labs:
  - Find out your account on our cdf machines. See the course website for details.
  - Try logging in.
- Read the course syllabus.
- Bookmark the course website.
- Do the class prep due Sunday night (due 11pm).