The Relational Model

csc343, Introduction to Databases
Diane Horton
Winter 2016

UNIVERSITY OF
TORONTO

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 $<d_1, d_2, \ldots, d_n>$ 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:

{<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.
- Formally:
  If attributes $a_1, a_2, \ldots, a_n$ form a superkey for relation R, $\nexists$ tuples t_1 and t_2 such that

$$(t_1.a_1 = t_2.a_1) \wedge (t_1.a_2 = t_2.a_2) \wedge \ldots \wedge (t_1.a_n = t_2.a_n)$$

Superkeys

- Example:
  - A relation called Course, with attributes department code, course number, and course name.
  - One tuple might be <“csc”, “343”, “Introduction to Databases”>
  - If department code + course number is a superkey for this relation, what is not allowed?
- Does every relation have a superkey?
- Can a relation have more than one superkey?

Keys

- A superkey may not be minimal.
  - I.e., you may be able to remove an attribute, and still have a set of attributes whose combined values are unique.
- key: a minimal superkey.
- We underline attributes in the schema to indicate that they form a key.

  Teams(<u>Name, HomeField</u>, Coach)
- Aside: Called "superkey" because it is a superset of some key.
  (Not necessarily a proper superset.)
- Can a relation have more than one key?

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.
  - Only a domain expert can determine that.
- Often we invent an 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.
- If aID is a key for Artists, does that mean a particular aID can appear only once in Roles??

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

- A bit of notation: R[A]
  - 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.
- We declare foreign key constraints this way:
  $R_1[X] \subseteq R_2[Y]$
- Example: Roles[aID] ⊆ Movies[mID]

UNIVERSITY OF TORONTO

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

  $$\text{Artists[aID]} \subseteq \text{Roles[aID]}$$

- 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:
  - Represent the data well.
    For example, avoid constraints that prevent expressing things that occur in the domain.
  - Avoid redundancy.
- Later, we'll learn some elegant theory that provides sound principles for good design.

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.