

The Relational Model

csc343, Introduction to Databases
Diane Horton, Michelle Craig, and Sina Meraji
Fall 2017

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 | Tracy Zheng |
| | Choppers | High Park | Ammar Jalali |

| 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, \dots, D_n are domains.
 - The **Cartesian product** $D_1 \times D_2 \times \dots \times D_n$ is the set of all tuples $\langle d_1, d_2, \dots, d_n \rangle$ such that $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$.
 - I.e., every combination of a value from D_1 , a value from D_2 etc.
- A **(mathematical) relation** on D_1, D_2, \dots, 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 $C = \{100, 200\}$.
 - $R = \{ \langle q, 2, 100 \rangle, \langle s, 3, 200 \rangle, \langle p, 1, 200 \rangle \}$ is a relation on A, B, C .
- Our database tables are relations too.
- Example:
 $\{ \langle \text{Rangers}, \text{Ducks}, 3, 0 \rangle, \langle \text{Ducks}, \text{Choppers}, 1, 1 \rangle, \langle \text{Rangers}, \text{Choppers}, 4, 2 \rangle, \langle \text{Choppers}, \text{Ducks}, 0, 5 \rangle \}$

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 | Tracy Zheng |
| | Choppers | High Park | Ammar Jalali |
| | Crullers | WTCS | Anna Liu |

- **relation** (table)
- **attribute** (column)
- **tuple** (row)
- **arity** of a relation: number of attributes
- **cardinality** of a relation: number of tuples

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

A schema for our league data

Teams(name, homefield, coach)

Games(hometeam, awayteam, homegoals, awaygoals)

- Can there be >1 team with the same name?
 - Didn't happen in our dataset, but could it?
 - That's up to the league, not us!
- Suppose the league wants to allow
 - multiple teams with same name
 - multiple teams with same home field
- The schema allows it: nothing says otherwise.
- But what if the league wants to disallow
 - multiple teams with the same name *and* home field?

Constraining the data

- More formally: \nexists tuples t_1 and t_2 such that
 $(t_1.name = t_2.name) \wedge$
 $(t_1.homefield = t_2.homefield)$
- Implication: If we know the values for {name, homefield}, we can look up *exactly* one team in the relation.
- I.e., attributes {name, homefield} uniquely identify tuples in this relation.
- We say that {name, homefield} is a “superkey” for relation Teams.

Superkeys

- Informally:

A superkey is 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, \dots, a_n form a **superkey** for relation R , \nexists tuples t_1 and t_2 such that

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

Example

- Course(dept, number, name, breadth)
 - One tuple might be
 <“csc”, “343”, “Introduction to Databases”, True>
- Suppose our knowledge of the domain tells us that no two tuples can have the same value for dept and number.
- This means that {dept, number} is a superkey.
- This is a constraint on what can go in the relation.
- Create an instance of Course that violates it.
- Does every relation have a superkey?

Keeping it minimal

- If $\{\text{dept}, \text{number}\}$ is a superkey, then so is $\{\text{dept}, \text{number}, \text{name}\}$.
 - This follows from the definition.
- But we are more interested in a *minimal* set of attributes with the superkey property.
 - Minimal in the sense that no attributes can be removed from the superkey without making it no longer a superkey.

Keys

- **key:** a minimal superkey.
- In the schema, by convention we often underline a key.
- Aside: The term “superkey” is related to the term “superset”.
 - A superkey 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 defines a kind of integrity constraint.

“Primary key”??

- Some of you have heard the term primary key.
- This is specific to SQL and we’ll learn about it later.

Example: Movies schema handout, Q1-6

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:
 $R_1[X] \subseteq R_2[Y]$
 - X and Y may be lists of attributes, of same arity
 - Y must be a key in R_2
- Example: $\text{Roles}[\text{mID}] \subseteq \text{Movies}[\text{mID}]$

Example: Movies schema handout, Q7-9

Referential integrity constraints

- These $R_1[X] \subseteq R_2[Y]$ relationships are a kind of **referential integrity constraint** or **inclusion dependency**.
- Not all referential integrity constraints are foreign key constraints.
- For example, we could say
 $\text{Artists}[\text{aID}] \subseteq \text{Roles}[\text{aID}]$
Here 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.