Lecture 2:

• Relational Model Basics

Announcements:

• Problem Set 1 out (due Thur, Sept 15)

Running Example

DB for a fictional e-scooter/e-bike sharing company ("micromobility")

• loosely based on parts of Global Bikeshare Feed Specification (GBFS)
• which is used by Lime(∗) and Bird
• we’ll use the running example in class and in some homeworks

Basic data stored:

• Types of vehicles and “real-time” vehicle data
• Basic customer information
• Price plans
• Customer trip information

(∗) e.g.: https://wrangler-mds-production.herokuapp.com/gbfs/Seattle,%20WA/free_bike_status.json
A relation’s “schema” includes

- The name of the relation (VehicleType)
- The set of named attributes (vt.id, make, model)
- Data types and constraints (aka the “domains” ... more later)

We often write a relation’s schema as $R(a_1, a_2, \ldots, a_n)$, e.g.:

VehicleType(vt.id, make, model)

Informally the relation’s schema + instance is called a “table”

A “database schema” is the set of relation schemas of the database

A “database instance” is the set of relation instances of the database
Arity vs Cardinality

VehicleType

<table>
<thead>
<tr>
<th>vt_id</th>
<th>make</th>
<th>model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Segway</td>
<td>Ninebot</td>
</tr>
<tr>
<td>2</td>
<td>Lime-S</td>
<td>Gen4</td>
</tr>
</tbody>
</table>

The “arity” (or “degree”) of a relation is the number of attributes

• What is the arity of this table? ... 3

The “cardinality” of a relation instance is the number of tuples

• What is the cardinality of this table? ... 2

Attribute Domains

A “domain” is both:

• the storage type (data type) of an attribute
• the set of allowable values of an attribute (constraint)

We assume domains are “atomic”

• as opposed to collection-based
• non-atomic example: vehicles attribute with vehicle ids of the type
• typical domain data types are integer, double, string, date, etc.
• domains can also restrict the allowable values (e.g., ages from 0 to 110)

The NULL value is special ...

• usually means the value is unknown or doesn’t exist (not applicable)
• e.g., some vehicle types might not have an “alternative_name” value
Attribute Keys

A “key” is an attribute (or set of attributes) with unique values
- Each row in the table must have a unique key value (cannot be NULL)
- We typically underline the key attribute(s)

Q: For the following relations, what are potential keys? ...

VehicleType(vt_id, make, model)
Vehicle(v_id, vt_id, lat, lon)
Trip(v_id, c_id, start_date, start_time, total_charge)

Note: Trip’s key is composite since it has more than 1 attribute
- we write composite keys in parens, e.g., (v_id, c_id, start_date, start_time)

More on Keys

Relations can have more than one set of key attributes:

Customer(c_id, email, ssn, ...)
- For example, this relation has three possible keys!
- each is a separate “candidate” key (vs a composite key)

Designating an attribute(s) as a key is a type of (operational) constraint
- DBMS will not allow duplicate row values to be inserted!
- We say the DBMS “enforces” a (designated) key constraint

A “primary key” is the designated key the DBMS enforces
- DBMSs typically enforce only the “primary” key (chosen by designer)
- if multiple candidate keys, only one selected as primary
- primary keys also imply a certain physical representation (more later)