Motivation

A relation is a mathematical abstraction for a table

- The theory of relations provides an elegant basis for databases
- Relational databases can be efficiently implemented on current architectures
Relation Model

- A relation is a table of values

<table>
<thead>
<tr>
<th>Relation Name</th>
<th>Attribute</th>
<th>Tuple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Student ID</td>
<td>Class</td>
</tr>
<tr>
<td>Smith</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>Brown</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

- Each tuple represents a collection of related data.
- All values in a column are of the same data type.

Is this a relation?

<table>
<thead>
<tr>
<th>Student</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Student ID</td>
<td>Emails</td>
</tr>
<tr>
<td>Smith</td>
<td>17</td>
<td>{<a href="mailto:smith@csc.ncsu.edu">smith@csc.ncsu.edu</a>, <a href="mailto:smith@unity.ncsu.edu">smith@unity.ncsu.edu</a>, <a href="mailto:smith@eos.ncsu.edu">smith@eos.ncsu.edu</a>}</td>
</tr>
<tr>
<td>Brown</td>
<td>8</td>
<td>{<a href="mailto:brown@unity.ncsu.edu">brown@unity.ncsu.edu</a>}</td>
</tr>
</tbody>
</table>
Domain D

- A domain of an attribute specifies all possible values the attribute can have.
- A domain is a set of atomic values.
  - Each value in the domain is indivisible.
  - Often expressed as a data type.
- Describe the domains of the following attributes
  - US_Phone_number:
  - Age:
  - Social_Security_Number:
  - Name:

Relation Schema

- A relation schema R(A1 ... An) describes the intension of a relation
  - R: relation name
  - A1, ..., An: a list of attributes
  - n: the degree of a relation of this schema.
  - attributes Ai name the roles or columns
  - dom(Ai) = Di: the domain of Ai.
  - attributes are distinct but may share their domain
An Example Schema

- Student(Name, SSN, HomePhone, Address, OfficePhone, Age, GPA)
  - Relation name: ?
  - Attributes: ?
  - Domains of the attributes: ?
  - Degree:

Relation

- A relation r(R) is an instance of the corresponding relation schema R.
  - Set of tuples of the form <v1 ... vn>
  - Each tuple is an ordered list
  - vi belongs to dom(Ai) or is null
  - thus vi is atomic: definition of 1NF
  - Also called relation extension
A Mathematical Definition of Relation

- A relation \( r(R) \) is a mathematical relation of degree \( n \) of the domains \( \text{dom}(A_1), \text{dom}(A_2), \ldots \), \( \text{dom}(A_n) \), which is a subset of the Cartesian product of the domains that define \( R \):
  \[
  r(R) \subseteq (\text{dom}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n))
  \]

- Mind exercise:
  - What is the number of all possible relations of \( R \)?

Achtung!

The ordering of
- attributes is important mathematically, but not in practice
- tuples is not important mathematically, but is in practice
An Alternative Definition of Relation

- A relation schema \( R = \{ A_1, A_2, \ldots, A_n \} \) is a set of attributes.
- A relation \( r(R) \) is a finite set of mappings 
  \[ r = \{ t_1, t_2, \ldots, t_m \}, \]
  - where each \( t_i \) is a mapping from \( R \) to \( D \),
  - \( D \) is the union of \( \text{dom}(A_i) \)'s,
  - \( t[A_i] \) must be in \( \text{dom}(A_i) \).
- Each tuple can be considered as a set of attribute-value pairs.

Notation

- Schemas: \( Q, R, S \)
- Instances: \( q, r, s \)
- Tuples: \( t, u, v \)
- \( t[A_i] = v_i \), where \( t = <v_1 \ldots v_n> \)
- \( t[A_i, A_j, \ldots A_k] = <v_i, v_j, \ldots v_k> \) (subtuple)
Relational Databases

- A relational database schema is
  - A set of relation schemas $S = \{ R_1, R_2, \ldots, R_m \}$
  and
  - A set of integrity constraints $IC$.
- A relational database is a set of relations $DB = \{ r_1, r_2, \ldots, r_m \}$ such that
  - each $r_i$ is a relation of $R_i$,
  - and $r_i$ satisfies the constraints in $IC$.

Constraints

- Constraints that can specified on relational schemas.
  - Domain constraints
  - Key constraints
  - Constraint about NULL
  - Reference integrity constraints
Domain Constraints

- $v_i$ belongs to $\text{dom}(A_i)$
- Domains map to standard data types

Key Constraints

- Superkey: subset of $\{A_i \ldots A_n\}$ that uniquely identifies $R$
- Key: minimal superkey
- Because $r(R)$ must be sets, $\{A_i \ldots A_n\}$ is always a superkey
- Candidate key: key
- Primary key: any single key so designated
An Example

Car

<table>
<thead>
<tr>
<th>LicenseNumber</th>
<th>EngineSerialNumber</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas ABC-739</td>
<td>A691324</td>
<td>Ford</td>
<td>Mustang</td>
<td>1996</td>
</tr>
<tr>
<td>Florida TVP-234</td>
<td>B43123123</td>
<td>Toyota</td>
<td>Camery</td>
<td>2000</td>
</tr>
<tr>
<td>NC 341-1324</td>
<td>2HG32341235</td>
<td>Honda</td>
<td>Civic</td>
<td>1998</td>
</tr>
</tbody>
</table>

- Super Keys:
- Keys:
- Candidate Keys:
- Primary Key:

Entity Integrity Constraints

- No primary key attribute may be null
- Motivation: a violation would be analogous to an uninitialized object
Referential Integrity Constraints

- Specified on two relations
  - Note that the previous constraints are on individual relations.

- Intuition
  - If a tuple in one relation refers to a tuple in another relation, the second tuple should exist.
  - Specified through foreign keys.

Foreign Keys

- A set of attributes, FK in relation R1, is foreign key iff
  - Attributes in FK occur as (primary) key in R2
  - FK reference or refer to the relation R2.
An Example

Driver

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>BirthDate</th>
<th>Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown</td>
<td>123-45-6789</td>
<td>01/01/50</td>
<td>Texas ABC-739</td>
</tr>
<tr>
<td>Smith</td>
<td>222-33-4444</td>
<td>02/22/60</td>
<td>Florida TVP-234</td>
</tr>
<tr>
<td>John</td>
<td>444-55-6666</td>
<td>05/01/56</td>
<td>NC 341-1324</td>
</tr>
</tbody>
</table>

Car

<table>
<thead>
<tr>
<th>LicenseNumber</th>
<th>EngineSerialNumber</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas ABC-739</td>
<td>A691324</td>
<td>Ford</td>
<td>Mustang</td>
<td>1996</td>
</tr>
<tr>
<td>Florida TVP-234</td>
<td>B43123123</td>
<td>Toyota</td>
<td>Camery</td>
<td>2000</td>
</tr>
<tr>
<td>NC 341-1324</td>
<td>2HG32341235</td>
<td>Honda</td>
<td>Civic</td>
<td>1998</td>
</tr>
</tbody>
</table>

Referential Integrity Constraints

- If t[FK] in r1 is all non-null, then u with u[FK]=t[FK] exists in r2
- Motivation: a violation is analogous to a dangling pointer
Semantic Integrity Constraints

- Application-specific constraints
  - typically involve restrictions on the values of an attribute with respect to some other attributes
  - some varieties of them may be specified as assertions in SQL2

General Strategy for Operations

- When legal, execute
- When illegal
  - reject
  - restore consistency (referential integrity)
    - change referenced tuple (relation)
    - change referencing tuple
An Example

Student

<table>
<thead>
<tr>
<th>Name</th>
<th>StudentID</th>
<th>Class</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>17</td>
<td>1</td>
<td>CS</td>
</tr>
<tr>
<td>Brown</td>
<td>8</td>
<td>2</td>
<td>CS</td>
</tr>
</tbody>
</table>

- Insert
  - <John, “N/A”, 2, “CS”>
  - <John, NULL, 2, “CS”>
  - <John, 17, 2, “CS”>
  - <John, 18, 2, “CS”>
Delete

Delete \(<v1 \ldots vn>\) from R

- execute: remove
- reject, e.g., if other tuples refer to it
- Correct the problem
  - cascade by deleting referring tuples
  - change values to null or default

_modify

As above
An Example

Consider the following operations:

- Delete the 1st tuple from Student.
- Insert <18, “CSC742”, …, “A-”> into Grade
- Change StudentID of the 1st tuple from 17 to 15.

Options: reject, correct with cascade, correct by changing the values (set to NULL, or a valid value).

<table>
<thead>
<tr>
<th>Name</th>
<th>StudentID</th>
<th>Class</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>17</td>
<td>1</td>
<td>CS</td>
</tr>
<tr>
<td>Brown</td>
<td>8</td>
<td>2</td>
<td>CS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StudentID</th>
<th>Course</th>
<th>…</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>CSC 742</td>
<td>…</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>CSC 742</td>
<td>…</td>
<td>B+</td>
</tr>
</tbody>
</table>