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

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 Model

- A relation is a table of values

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

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

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:

Achtung!

The ordering of attributes is important mathematically, but not in practice.
Tuples is not important mathematically, but is in practice.

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

An Alternative Definition of Relation

- A relation schema R=\{A1, A2, ..., An\} is a set of attributes.
- A relation r(R) is a finite set of mappings
  r={t1, t2, ..., tm},
  - where each ti is a mapping from R to D,
  - D is the union of dom(Ai)'s,
  - t[AI] must be in dom(Ai).
- Each tuple can be considered as a set of attribute-value pairs.

A Mathematical Definition of Relation

- A relation r(R) is a mathematical relation of degree n of the domains dom(A1), dom(A2), ..., dom(An), which is a subset of the Cartesian product of the domains that define R:
r(R) \subseteq (dom(A1) \times dom(A2) \times ... \times dom(An))
- Mind exercise:
  - What is the number of all possible relations of R?

Notation

- Schemas: Q, R, S
- Instances: q, r, s
- Tuples: t, u, v
- t[AI] = vi, where t = <v1 ... vn>
- t[AI, Aj, ..., Ak] = <vi, vj, ... vk> (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 \( IC \).

Key Constraints

- Superkey: subset of \( \{ A_1 \ldots A_n \} \) that uniquely identifies \( R \)
- Key: minimal superkey
- Because \( r(R) \) must be sets, \( \{ A_1 \ldots A_n \} \) is always a superkey
- Candidate key: key
- Primary key: any single key so designated

Constraints

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

An Example

<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-759</td>
<td>A691324</td>
<td>Ford</td>
<td>Mustang</td>
<td>1996</td>
</tr>
<tr>
<td>Florida TVP-234</td>
<td>B431253123</td>
<td>Toyota</td>
<td>Camery</td>
<td>2000</td>
</tr>
<tr>
<td>NC 341-321</td>
<td>2HG32541235</td>
<td>Honda</td>
<td>Civic</td>
<td>1998</td>
</tr>
</tbody>
</table>

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

Domain Constraints

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

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.

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

An Example

<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>02/23/60</td>
<td>Florida TVP-234</td>
</tr>
<tr>
<td>John</td>
<td>444-55-6666</td>
<td>05/03/56</td>
<td>NC 341-1324</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LicenseNumber</th>
<th>Engine/SerialNumber</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas ABC-739</td>
<td>A693124</td>
<td>Ford</td>
<td>Mustang</td>
<td>1996</td>
</tr>
<tr>
<td>Florida TVP-234</td>
<td>B43123123</td>
<td>Toyota</td>
<td>Camry</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>

General Strategy for Operations
- When legal, execute
- When illegal
  - reject
  - restore consistency (referential integrity)
    - change referenced tuple (relation)
    - change referencing tuple
Insert

Insert \(<v_1 \ldots v_n>\) into \(R\) (means current instance)
- execute: just add it
- reject, e.g., when the key is already used
- Correct the problem
  - Ask user to create referenced tuple or change the referencing tuple.
  - Creating referenced tuple may be cascading.

An Example

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

- Insert
  - \(<\text{John, } "N/A", 2, "CS">\)
  - \(<\text{John, NULL, 2, } "CS">\)
  - \(<\text{John, 17, 2, } "CS">\)
  - \(<\text{John, 18, 2, } "CS">\)

Modify

- As above

An Example

<table>
<thead>
<tr>
<th>Grade</th>
<th>StudentID</th>
<th>Course</th>
<th>...</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>17</td>
<td>CSC742</td>
<td>...</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>CSC742</td>
<td>...</td>
<td>B+</td>
</tr>
</tbody>
</table>

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

- Consider the following operations:
  - Delete the 1st tuple from Student.
  - Insert \(<18, "CSC742", \ldots, "A-">\) 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).

Delete

Delete \(<v_1 \ldots v_n>\) 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