10/3/2015 Relational Model

Lecture Topics
I. Relational Model
II. Integrity Constraints
III. ER Diagrams to Relational Schemas

I. Relational Model

- Formulated by Edgar F. Codd in 1969.
- Based on first order predicate logic.
- Formal foundation for most databases (relational databases).
- Alternative to hierarchal database model (tree-like) and network database model (graph-like, no hierarchy).
- Codd won Turing Award in 1981; RDBMs are multi-billion dollar industry.

I.1 Basic Concepts

- Main construct is a relation.
  A relation (for now) is a set.
  In SQL, a relation is a table.

- A set is a bag of elements.
  Elements can be sets.
  \( 2 \in \{2, 3, 4, 5\} \)
  means 2 is an element of \( \{2, 3, 4, 5\} \).
- You cannot say how many times an element is in the set (not a multiset).
- Cannot specify position of elements (i.e. not a sequence).

sets A and B are equal iff they have the same elements
\( A = B \iff \forall x [x \in A \iff x \in B] \)

A relation is a set of tuples.
In mathematics, a tuple is a collection of elements that has an order, and allows duplication.
In relational model, attribute names are used instead of position.
Intuition, a relation is a table; the table is a set of rows; each row is a tuple.

we will write
\( R(A, B) \)
means table R has columns A, B

relational instance = current value for a relation with defined columns and domains

a relational schema specifies a relation in general:
- defines a finite set of finite relations.

- Defining a schema (informally)
  - Give it a name \((R)\)
  - Choose number of columns \((2)\)
  - Give each column a unique name \((A, B)\)
  - Decide domains of columns (letters, ints)
  - Decide constraints, if any
    (e.g. if two rows have same A, must have same B)

- Note:
  (1) Relations may have duplicates
  (2) \( R(A, B) \) same as \( R(B, A) \)
II. Integrity Constraints

An integrity constraint restricts
the data that can be stored in
an instance of a database.

DBMS must enforce integrity constraints
- e.g. domain constraint to try to put
  3 grades must be less than 10
  - no two students have the same ID.

II.1 Primary Key

<table>
<thead>
<tr>
<th>Student</th>
<th>sid</th>
<th>Grade</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>John</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>Bob</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>Alice</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>Emily</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>John</td>
<td></td>
</tr>
</tbody>
</table>

- A superkey is a set of attributes
  such that for any two tuples in
  the relation, if they are equal in the
  values of the attributes, then they
  must be equal in all other attributes.

- We will be told this - specified in
  some way.

- A relation always has at least one
  superkey (e.g., all attributes).

- A minimal superkey is a key.
- A relation always has at least
  one key. May be more than one.

- One is chosen as primary key.
- Others are called candidate keys.

Note: A superkey is a set of
columns whose values determine
the values of all columns in a row.
A key is a subset of a superkey,
but not all subsets are keys.

Examples

City (Longitude, Latitude, Country, State, Name, Size)
- or -
City (Longitude, Latitude, Country, State, Name, Size)

Primary key is understood

II.2 Foreign Key

Scenario: 5 employees, unique IDs, 0 or more children

<table>
<thead>
<tr>
<th>Employee</th>
<th>ID#</th>
<th>Name</th>
<th>Child</th>
<th>Child</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>Erica</td>
<td>Frank</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>Billy</td>
<td>Julie</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Carol</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>David</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Bob</td>
<td></td>
<td>Frank</td>
<td></td>
</tr>
</tbody>
</table>

Does Not Work!

- There must be a fixed number of
columns.

Child is a multi-valued attribute.

How to fix? Replace with two tables:

```
<table>
<thead>
<tr>
<th>Employee</th>
<th>ID#</th>
<th>Name</th>
<th>Child</th>
<th>Child</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Carol</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>David</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Bob</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Primary key of Employee is ID#.
Primary key of Child is ID#.

Need both.

Strategy to handle multi-valued attributes:
- Create "main" table with all attributes
  other than multi-valued.
- Primary key is original key.
- Create a second table with primary key
  and multi-valued attribute.
- Primary key is original key and
  multi-valued attribute.
- Note: Any value of ID# in child must also be in Employee.
- This is a foreign key.
- A foreign key is a set of attributes that uniquely identify a row in a different table.
- A FK need not be a key of the table (e.g., ID# is not a key in child).
- The attributes must be a key in the other table.

Foreign Key = Binary Many-to-one relationship between tables.

(partial function)

- Names don't have to be the same, but must indicate what FK is referring to.

Many-to-many relationships:

Example: employees like animals.

<table>
<thead>
<tr>
<th>Employee</th>
<th>ID#</th>
<th>Name</th>
<th>Animal</th>
<th>Species</th>
<th>Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alfa</td>
<td></td>
<td>Horse</td>
<td>Asin</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Beta</td>
<td></td>
<td>Will</td>
<td>Asin</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Gam</td>
<td></td>
<td>Cat</td>
<td>Afric</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Dav</td>
<td></td>
<td>Yik</td>
<td>Asin</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>B1</td>
<td></td>
<td>Zebo</td>
<td>Afric</td>
<td></td>
</tr>
</tbody>
</table>

How to fix?

Replace binary many-to-many with a new table and two many-to-one relationships.

Employee | ID# | Name |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Likes</th>
<th>ID#</th>
<th>Species</th>
<th>Animal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Foreign key constraints:

ID# in Likes is an FK referencing Employee.
Species in Likes is an FK referencing Animals.

Referential Integrity

Example:

<table>
<thead>
<tr>
<th>Professor</th>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Robert</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Anthony</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course</th>
<th>Id</th>
<th>Year</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2015</td>
<td>Data Management</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2015</td>
<td>Networking</td>
<td></td>
</tr>
</tbody>
</table>

What if I delete (1, Robert) from Professor?

Part of FK specification

1) ON DELETE NO ACTION = "need" row in professor can't be deleted

2) ON DELETE CASCADE = delete in Professor and Course

3) ON DELETE SET NULL = Replace ID with NULL

[CHANGE?]

ON UPDATE CASCADE = Change ID in Course too

A relational database is a:

- set of relations
- set of binary many-to-one mappings between them
Crows Feet Notation
- expresses cardinality

End of lines:
0..1

0..* ₁

1..₁ ₁

1..* ₁

Pattern of lines:
A dashed line indicates a "non-identifying" relationship.
Primary key of the "many side" does not include the primary key of the "one side"

Each person is born in 0 or 1 countries.
Each country has 0 or many people born in it.

Recursive Relationships
- **Strong Entity**
  - Create table without multi-valued attributes, flatten composite attributes
  - Create table for multi-valued attributes and primary key of "main" table

- **ISA**
  - Don't produce anything for ISA
  - Implement the "super class" as a table
  - Create a table with attributes of subclass, and the primary key of the superclass

- **Weak Entity and Defining relation**
  - Don't produce anything for the defining relation
  - Create a table for the weak entity augmented with the primary key of the strong entity
  - Key is primary key + discriminant

- **Relationship (Not Binary Many-to-one)**
  - Create a table with primary keys of participating tables and relationship attributes
  - Primary key is all the primary keys of participants