The Relational Data Model

Data Models

Structure - Operations - Constraints

Types:
- Relational data model
- Object data model
- Hierarchical data model (Trees)
- Network data model (Graphs)
- Semistructured Data Model

Relational vs Semistructured

Figure 2.1: An example relation.

Figure 2.2: Movie data as XML.
Relational Database Model

Introduced by E.F. Codd (1970)
http://www.acm.org/classics/nov95/

Based on relational algebra and logic developed by

Schröder (1880s)
Charles Peirce (1890s)
Russell and Whitehead (1900s)

Codd's Twelve Rules

1. Information represented at the logical level in tables.
2. Data is determined by table, primary key, and column.
3. Missing information is modeled as null values.
4. Metadata is part of the database.
5. Single language for all tasks in DBMS.
6. Views and tables must change simultaneously.
7. Single operations for retrieve, insert, delete, update.
8. Operations independent of physical storage and access.
9. Database modifiable without affecting applications.
10. Constraints are part of database.
11. DML independent of physical layer (distributed, etc.)
12. Row-processing obeys same rules as set-processing.

Relations

Extensional versus intensional
Extensional Representation:

- table of values
- rows = records
- columns = attributes

Note:
- rows in tables are ordered,
- instances of relations are not
Domains

Set of atomic values for an attribute
atomic = indivisible
(e.g. CSC 355 = CSC + 355 is divisible)

Examples
age: integer
sex: (male, female)

Physical Level: data type + format

Relation Schema

R(A₁, A₂, … , Aᵦ) relational schema
R : Name of Relation
A₁, A₂, … , Aᵦ : Attributes
n : degree (arity) of R

Example:
Movie(movieID, title, genre, length, rating)

Domains

Relation Schema with Domains

R(A₁:D₁, A₂:D₂, … , Aᵦ:Dᵦ) ← Aᵦ has domain Dᵦ

Example:
Movie(movieID: integer, title:string, genre: Genres, length: Lengths, rating: Ratings)

dom(genre) = Genres = {Musical, Horror, …}
dom(length) = Lengths = {x: x is valid time}
dom(rating) = Ratings = {NR, G, PG, PG-13, R, NC-17}

or
Movie(movieID: integer, title:string, genre: string, length: integer, rating: string)
Relational Schemas Example

CUSTOMER(Customer_ID, Customer_Name, City, State, Postal_Code)
ORDER(Order_ID, Order_Date, Customer_ID)
ORDERLINE(Order_ID, Product_ID, Ordered_Quantity)
PRODUCT(Product_ID, Product_Description, Product_Finish, Standard_Price)

Instances

Given \( R(A_1, A_2, ..., A_n) \), \( A_i \) has domain \( D_i \)

Instance of schema \( R \) is a table with data from domains

Example: 

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Genre</th>
<th>Length</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Thirty-Nine Steps</td>
<td>mystery</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>510</td>
<td>Monkey Business</td>
<td>comedy</td>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>

null: value unknown, or attribute does not apply
values atomic: no multiple values (first normal form)
(indivisible (name = first name + last name))
What about multiple values?

Multiple programs?
Telephone numbers?
Movie Genres?

Constraints

- Domain constraints
- Key (uniqueness) constraints
- Entity integrity constraints
- Referential integrity constraints
- Data dependencies (functional dependencies, etc.)

Domain Constraints

Restriction on values of attributes (domain).
Specified as data-type: integer, char, etc., or user-defined type
Operations on data-types: +, *, <, =, ...
not null constraint for an attribute
Keys

Key: smallest set of attributes that uniquely identify a record in the relational schema underlined in relational schema

composite: more than one attribute

Examples:
• MOVIE(MovieID, Title, Year, Length, Rating)
  
  (MovieID, Title) is not a key!

• MEMBER(StudentID, Groupname, Joined)

Key Examples

PRODUCT(ProductID, Name, Description, PricePerUnit, UnitSize)

ACTIVITY(StudentID, Activity, Fee)

COURSE(CourseID, Title, Enrolment)

ENROLLED(StudentID, CourseID, Qt, Year)

Candidate Keys

If a relation has more than one key, these keys are called candidate keys.

Examples
• EMPLOYEE(EmpID, FirstName, LastName, Salary, Gender)
• DePaul students: peoplesoftID and SSN
• COURSE(Department, Number, Name, Instructor)
• CAR(OwnerName, Vehicle#, Engine#, Color)

One candidate key is declared the primary key of the relation (underlined in schema)
Relational Databases and Schemas

Relational Database Schema: Collection of Relations
Relational Database State: Collection of Instances

ACTIVITIES = (STUDENT, ACTIVITY)

<table>
<thead>
<tr>
<th>Student</th>
<th>SID</th>
<th>FName</th>
<th>LName</th>
<th>Activity</th>
<th>StudentID</th>
<th>Activity</th>
<th>Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>101</td>
<td>Mark</td>
<td>Spencer</td>
<td></td>
<td>971</td>
<td>Piano</td>
<td>$20</td>
</tr>
<tr>
<td></td>
<td>971</td>
<td>Charles</td>
<td>Loeffler</td>
<td></td>
<td>971</td>
<td>Swimming</td>
<td>$10</td>
</tr>
</tbody>
</table>

**Note:** different names (SID, StudentID) for the same concepts

Foreign Key 1

A set of attributes in one relation (R₁) referring to a unique tuple in a second relation (R₂) through R₂'s primary key.

<table>
<thead>
<tr>
<th>Student</th>
<th>SID</th>
<th>FName</th>
<th>LName</th>
<th>Activity</th>
<th>StudentID</th>
<th>Activity</th>
<th>Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>101</td>
<td>Mark</td>
<td>Spencer</td>
<td></td>
<td>971</td>
<td>Piano</td>
<td>$20</td>
</tr>
<tr>
<td></td>
<td>353</td>
<td>Gil</td>
<td>Ryle</td>
<td></td>
<td>973</td>
<td>Reading</td>
<td>$5</td>
</tr>
<tr>
<td></td>
<td>971</td>
<td>Charles</td>
<td>Loeffler</td>
<td></td>
<td>971</td>
<td>Swimming</td>
<td>$10</td>
</tr>
</tbody>
</table>

**Terminology**

R₁ referencing relation
R₂ referenced relation

Foreign Key 2

**Examples**

REGISTRATION = {STUDENT, ENROLMENT, COURSE}
COMPANY = {EMPLOYEE, WORKS_ON, PROJECT}
SUPPLY = {SUPPLIES, SUPPLIER, PART, COMPANY}

**Note:** R₁ = R₂ is possible

**Example**

EMPLOYEE (with supervisor)
MOVIE (remakes)
Referential Integrity

declaration of foreign keys in a database schema

STUDENT(SID, FName, LName)
ACTIVITIES(StudentID references STUDENT, Activity, Fee)

or visually, by an arrow from foreign key to primary key

STUDENT(SID, FName, LName)

ACTIVITIES(StudentID, Activity, Fee)

Integrity Constraints

- Domain Constraints: declaration of domains
- Not Null Constraints: attribute values can not be null
- Key Constraints: candidate keys (uniqueness)
- Entity Integrity Constraint: primary key is not null
- Referential Integrity Constraint: declaring foreign keys

A valid state is a database state fulfilling all integrity constraints
Integrity Constraints defined by DDL
Semantic constraints (transitions) later