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: tables are ordered, relations are not
Domains

Set of atomic values for an attribute:
- atomic = indivisible
  (e.g. \( IT \ 240 = IT + 240 \) is divisible)

Examples
- phone_number = \( \{x: x \text{ is a valid phone number}\} \)
- age = \( \{x: x \text{ is a valid age}\} \)

Physical Level: data type + format

Relation Schema

\( R(A_1, A_2, \ldots, A_n) \) relational schema
- \( R \): Name of Relation
- \( A_1, A_2, \ldots, A_n \): Attributes \( A_i \) has domain \( D_i \)
- \( N \): degree (arity) of \( R \)

Movie(movieID, title, genre, length, rating)
- \( \text{dom(movieID)} = \text{MovieIDs} = \{x: x \text{ is a number}\} \)
- \( \text{dom(title)} = \text{Titles} = \{x: x \text{ is a word}\} \)
- \( \text{dom(genre)} = \text{Genres} = \{\text{Musical, Horror, } \ldots\} \)
- \( \text{dom(length)} = \text{Lengths} = \{x: x \text{ is valid time}\} \)
- \( \text{dom(rating)} = \text{Ratings} = \{\text{PG, R, PG-13, NC-17, } \ldots\} \)

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, Product_Line_ID)

Figure 5-3, page 191
Instances

$R(A_1, A_2, \ldots, A_n)$ relational schema,
$A_i$ has domain $D_i$

Instance of schema $R$ is a table with data from domains

Example:

Figure 5-4, page 192 (tables)
Table 5-1, page 193 (domains)

Schema and Relation Examples

STUDENT(Name, SSN, HomePhone, Address, OfficePhone, Age, GPA)

MOVIE

VIDEO

RENTAL

Records

<101, "Thirty-Nine Steps", mystery, 101, R>
<510, "Monkey Business", comedy, null, null>

are possible records (or tuples) in

MOVIE(movieID, title, genre, length, rating).  

null: value unknown, or attribute does not apply values atomic: no multiple values (first normal form)  
(e.g. several genres)

indivisible (name = first name + last name)
What about multiple values?

Example

Telephone numbers

Figure 5-2, page 190

Functional Dependencies I

ORDER_DETAIL(Product, Quantity, PricePerUnit, Subtotal)

Example

<Soy milk, 3, $3.19, $9.57> in ORDER_DETAIL

(Quantity, PricePerUnit) → Subtotal

Quantity and PricePerUnit determine Subtotal

Functional Dependencies II

A → B (A determines B)

if setting the values of attribute(s) of A determines the values of B

Example:

SSN → Name in STUDENT(SSN, Name)
Functional Dependencies III

- EMPLOYEE(EmpID, FirstName, LastName, Salary, Sex)
- STUDENT(Name, SSN, HomePhone, Address, OfficePhone, Age, GPA)
- PRODUCT(ProductID, Name, Description, PricePerUnit, UnitSize)
- CAR(OwnerName, Vehicle#, Engine#, Color)
- CUSTOMER(accountID, lastName, firstName, street, city, state, zipcode)

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 determines all attributes in a relation. If key has more than one attribute, it is called composite.

Example:
STUDENT(Name, SSN, HomePhone, Address, OfficePhone, Age, GPA)
(SSN): key
(SSN, Name, Address): not key

Key Examples

MOVIE(Title, Year, Director, Length, Rating)
PRODUCT(ProductID, Name, Description, PricePerUnit, UnitSize)
ACTIVITY(StudentID, Activity, Fee)
COURSE(CourseID, Title, Enrolment)
ENROLLED(StudentID, CourseID, CourseTitle, Grade)

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: peoplesofID 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>StudentID</th>
<th>FName</th>
<th>LName</th>
<th>Activity</th>
<th>StudentID</th>
<th>Activity</th>
<th>Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Mark</td>
<td>Spencer</td>
<td>Piano</td>
<td>971</td>
<td>Piano</td>
<td>$20</td>
</tr>
<tr>
<td>971</td>
<td>Charles</td>
<td>Loeffler</td>
<td>Swimming</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 ($R_1$) referring to a unique tuple in a second relation ($R_2$) through $R_2$’s primary key.

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

Terminology

$R_1$: referencing relation
$R_2$: referenced relation

Foreign Key 2

Examples

REGISTRATION = (STUDENT, ENROLMENT, COURSE)
COMPANY = (EMPLOYEE, WORKS_ON, PROJECT)
SUPPLY = (SUPPLIES, SUPPLIER, PART, COMPANY)

Note: $R_1 = R_2$ 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 (transition) later