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

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

Figure 2.1: An example relation

```
<Movies>
    <Movie title="Gone With the Wind">
        <Year>1939</Year>
        <Length>231</Length>
        <Genre>drama</Genre>
    </Movie>
    <Movie title="Star Wars">
        <Year>1977</Year>
        <Length>124</Length>
        <Genre>sciFi</Genre>
    </Movie>
    <Movie title="Wayne's World">
        <Year>1992</Year>
        <Length>95</Length>
        <Genre>comedy</Genre>
    </Movie>
</Movies>
```

Figure 2.2: Movie data as XML

Ullman, Widom, *A First Course*, p. 19/20

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

$R(A_1, A_2, \ldots, A_n)$ **relational schema**

R : Name of Relation
$A_1, A_2, \ldots, A_n$: **Attributes**
n : **degree** (**arity**) of R

Example:

```
Movie(movieID, title, genre, length, rating)
```

Domains ?

Relation Schema with Domains

$R(A_1:D_1, A_2:D_1, \ldots, A_n:D_n)$ ← A_i has domain D_i

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

Instance of schema R is a table with data from domains

Example: `Student(LastName, FirstName, SID, SSN, Career, Program, City, Started)`

LastName	FirstName	SID	SSN	Career	Program	City	Started
Snowdon	Jonathan	8871	123123123	GRD	INFO-SYS	Springfield	2009
Winter	Abigail	11035	111111111	GRD	PHD	Chicago	2009
Patel	Deepa	14862		GRD	COMP-SCI	Evanston	2013
Degroff	Jarvis	14998	113311331	GRD	COMP-GAM	Evanston	2012
Starck	Jason	19992	789789789	UGRD	INFO-SYS	Springfield	2009
Johnson	Peter	32105	123456789	UGRD	COMP-SCI	Chicago	2010
Pollard	Joya	39077		GRD	COMP-SCI	Springfield	2010
Kubik	Dwayne	57923	979797979	UGRD	COMP-SCI	Springfield	2013
Skelly	Trinity	58992	555222555	GRD	PHD	Springfield	2012
Krol	Angelo	60973		UGRD	COMP-SCI	Springfield	2011
Patel	Prakash	75234		UGRD	COMP-SCI	Chicago	2011
Brennigan	Marcus	90421	987654321	UGRD	COMP-GAM	Evanston	2010
Snowdon	Jennifer	93321	321321321	GRD	COMP-SCI	Springfield	2012

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?

student

LastName	FirstName	SID	SSN	Career	Program	City	Started
Snowdon	Jonathan	8871	123123123	GRD	INFO-SYS	Springfield	2009
Winter	Abigail	11035	111111111	GRD	PHD	Chicago	2009
Patel	Deepa	14862		GRD	COMP-SCI	Evanston	2013
Degroff	Jarvis	14998	113111331	GRD	COMP-GAM	Evanston	2012
Starck	Jason	19992	789789789	UGRD	INFO-SYS	Springfield	2009
Johnson	Peter	32105	123456789	UGRD	COMP-SCI	Chicago	2010
Pollard	Joya	39077		GRD	COMP-SCI	Springfield	2010
Kubik	Dwayne	57923	979797979	UGRD	COMP-SCI	Springfield	2013
Skelly	Trinity	58992	555222555	GRD	PHD	Springfield	2012
Krol	Angelo	60973		UGRD	COMP-SCI	Springfield	2011
Patel	Prakash	75234		UGRD	COMP-SCI	Chicago	2011
Brennigan	Mertus	90421	987654321	UGRD	COMP-GAM	Evanston	2010
Snowdon	Jennifer	93321	321321321	GRD	COMP-SCI	Springfield	2012

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)

ENROLED(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}

Student	SID	FName	LName
	101	Mark	Spencer
	971	Charles	Loeffler

Activity	StudentID	Activity	Fee
	971	Piano	$20
	971	Swimming	$10

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

Foreign Key 1

A set of attributes in one relation (R_1) refering to a unique tuple in a second relation (R_2) through R_2's primary key.

Student	SID	FName	LName
	101	Mark	Spencer
	353	Gil	Ryle
	971	Charles	Loeffler

Activity	StudentID	Activity	Fee
	971	Piano	$20
	353	Reading	$5
	971	Swimming	$10

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: R1 = R2 is possible

Example

EMPLOYEE (with supervisor)

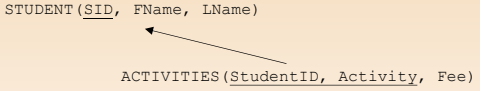
MOVIE (remakes)

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



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