Chapter 2
The relational Model of data

Relational model introduction
Contents

- What is a data model?
- Basics of the relational model

Next:
- How to define?
- How to query?
- Constraints on relations
What is a Data Model?

A data model is a notation for describing data or information. It consists of three parts:

- **Structure** of the data: mathematical representation of data
- **Operations** on data.
- **Constraints**.
Two important data models

* The relational model (and object-relational model):
  relational model = tables

* The semistructured-data model
  semistructured model = trees/graphs
  XML and its related standards.
A relation is a Table

Each attribute has a domain, an element type.

<table>
<thead>
<tr>
<th>name</th>
<th>manf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winterbrew</td>
<td>Pete’s Bud Lite</td>
</tr>
<tr>
<td>Bud Lite</td>
<td>Anheuser-busch</td>
</tr>
</tbody>
</table>

Relation name

Attributes (column headers)

Tuples (rows)

Beers
Schemas (模式)

* **Relation schema** = relation name and attribute list.
  * Optionally: types of attributes.
  * Example: `Beers(name, manf)` or `Beers(name: string, manf: string)`

* **Database** = collection of relations.

* **Database schema** = set of all relation schemas in the database.
Relation Instances (关系实例)

- is **current** set of rows for a relation schema.
- Example: beer relation

<table>
<thead>
<tr>
<th>Name</th>
<th>Manf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winterblue</td>
<td>Peters</td>
</tr>
<tr>
<td>Budlit</td>
<td>A.B.</td>
</tr>
</tbody>
</table>

Dynamic changing
Key of Relations

- There are many constraints on relations
- Key constraints is one of them

For example:
Beer(name, manf)

If name is a key, do not allow two tuples to have the same name.

- Each object should be distinguished in the world
Why Relations?

- Very simple model.
- Often matches how we think about data.
- Abstract model that underlies SQL, the most important database language today.
a Running Example

Beers(name, manf)
Bars(name, addr, license)
Drinkers(name, addr, phone)
Likes(drinker, beer)
Sells(bar, beer, price)
Frequents(drinker, bar)

* Underline = key (tuples cannot have the same value in all key attributes).
Database Schemas in SQL

- SQL is primarily a **query language**, for getting information from a database.
- SQL also includes a **data-definition** component for describing database schemas.
Creating (Declaring) a Relation

* Simplest form is:

```
CREATE TABLE <name> (  
    <list of elements>
);
```

* To delete a relation:

```
DROP TABLE <name>;
```
Creating (Declaring) a Relation (cont.)

- To modify schemas

  ALTER TABLE <name> ADD <new attribute>

  ALTER TABLE <name> DROP <attribute>
Three kinds of table

* **Stored relations**: tables, a relation that exists in the database, can be modified or queried. real, stored.

* **Views**: relations defined by a computation. virutal, not really exists.

* **Temporary tables**: constructed by the SQL processor when it performs. thrown away, not stored.
Elements of Table Declarations

Most basic element: an attribute and its type.

The most common types are:

* INT or INTEGER (synonyms).
* REAL or FLOAT (synonyms).
* CHAR(n) = fixed-length string of n characters.
* VARCHAR(n) = variable-length string of up to n characters.
Example: Create Table

CREATE TABLE Sells (  
  bar   CHAR(20),
  beer  VARCHAR(20),
  price REAL  
);
SQL Values

- Integers
- reals
- Strings requires *single quotes*.  
  - Two single quotes = real quote, e.g., ’Joe’s Bar’.
- Bit strings of fixed or varying length, BIT(n) means bit string of length n
- Any value can be NULL.
- Boolean: true, false, unknown
Dates and Times in SQL

- The form of a date value is:
  \[
  \text{DATE} \ 'yyyy-mm-dd'
  \]

- The form of a time value is:
  \[
  \text{TIME} \ 'hh:mm:ss'
  \]
  Example: TIME '15:30:02.5' = two and a half seconds after 3:30PM.
Declaring Keys

* An attribute or list of attributes may be declared PRIMARY KEY or UNIQUE.
* Meaning: no two tuples of the relation may agree in all the attribute(s) on the list.
* PRIMARY KEY or UNIQUE attributes can be declared when creating a table.
Declaring Single-Attribute Keys

* Place PRIMARY KEY or UNIQUE after the type in the declaration of the attribute.

* Example:

```sql
CREATE TABLE Beers (  
  name CHAR(20) UNIQUE,  
  manf CHAR(20)  
) ;
```
A key declaration can also be another element in the list of elements of a CREATE TABLE statement. This form is essential if the key consists of more than one attribute. May be used even for one-attribute keys.
Example: Multiattribute Key

* The bar and beer together are the key for Sells:

```sql
CREATE TABLE Sells (  
  bar  CHAR(20),
  beer VARCHAR(20),
  price REAL,
  PRIMARY KEY (bar, beer)
);
```
In a table declaration:

1. **PRIMARY KEY**: only one PRIMARY KEY, No attribute of a PRIMARY KEY can ever be NULL in any tuple.

2. **UNIQUE**: several UNIQUE attributes, may have NULL’s values.
Other Attributes Properties

- **NOT NULL** = every tuple must have a real value for this attribute. i.e. the value for this attribute may never be NULL.

- **DEFAULT value** = says that if there is no specific value known for this attribute’s component in some tuple, use the stated <value>. 
Example: Default Values

CREATE TABLE Drinkers (  
  name CHAR(30) PRIMARY KEY,  
  addr CHAR(50)  
  DEFAULT '123 Sesame St.',  
  phone CHAR(16)  
);
Effect of Defaults

[bullet point]

insert the fact that Sally is a drinker, but we know neither her address nor her phone.

```
INSERT INTO Drinkers(name)
VALUES(‘Sally’);
```
Effect of Defaults (cont.)

- What tuple appears in Drinkers?

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘Sally’</td>
<td>‘123 Sesame St’</td>
<td>NULL</td>
</tr>
</tbody>
</table>

- If we had declared phone NOT NULL, this insertion would have been rejected.
Semistructured Data

Based on trees.

**Motivation:**

- flexible representation of data.
- sharing of *documents* among systems and databases.
Graphs of Semistructured Data

- **Nodes** = objects.
- **Labels** on arcs (like attribute names).
- **Atomic values** at leaf nodes (nodes with no arcs out).
- **Flexibility**: no restriction on
  - Labels out of a node.
  - Number of successors with a given label.
Example: Data Graph

Notice a new kind of data.

The bar object for Joe’s Bar

The beer object for Bud

root

beer

manf

name

addr

name

servedAt

root

beer

manf

name

1995

year

award

Gold

The beer object for Bud

The bar object for Joe’s Bar

Joe’s

Bud

Maple

M’lob

A.B.
JavaScript Object Notation (JSON)

- Standard for “serializing” data objects
- Human-readable, useful for data interchange
- Useful for representing and storing semistructured data
JSON example

{"Beers":
 [ {"name": "Bud",
   "manf": "A.B.",
   "price": 13},
 {"name": "Mobel",
   "manf": "A.B.",
   "Prize": {"year": 1995,
           "award": "gold"}}
 ]

Basic constructs (recursive)
• Base values
  number, string, boolean, ...
• Objects { }
  sets of label-value pairs
• Arrays [ ]
  lists of values
## Relational Model versus JSON

<table>
<thead>
<tr>
<th></th>
<th>Relational</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure</strong></td>
<td>Tables</td>
<td>Nested sets, array</td>
</tr>
<tr>
<td><strong>schema</strong></td>
<td>Fixed in advance</td>
<td>Flexible, self describing</td>
</tr>
<tr>
<td><strong>Queries</strong></td>
<td>Simple expressive language</td>
<td>Not widely used</td>
</tr>
<tr>
<td><strong>Ordering</strong></td>
<td>none</td>
<td>arrays</td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>Native system</td>
<td>NOSQL system</td>
</tr>
</tbody>
</table>
## XML versus JSON

<table>
<thead>
<tr>
<th></th>
<th>XML</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verbosity</td>
<td>More</td>
<td>Less</td>
</tr>
<tr>
<td>Complexity</td>
<td>More</td>
<td>Less</td>
</tr>
<tr>
<td>Validity</td>
<td>DTD, XSD, widely used</td>
<td>JSON scheme, not widely used</td>
</tr>
<tr>
<td>Prog. Interface</td>
<td>mismatch</td>
<td>More direct</td>
</tr>
<tr>
<td>Querying</td>
<td>Xpath,Xquery</td>
<td>Json Path, Json Query</td>
</tr>
</tbody>
</table>
Summarization

Relational model, XML model, JSON notations

A data model consists of three parts:

* Data structure ✓
* Operations on the data ?
* Constraints ?

Next:

* Relational algebra: operations & constraints.
* Relational algebra: the core of the SQL.