Database Systems I

The Relational Data Model

Instructor: Ouldooz Baghban Karimi

CMPT 354 - Summer 2019
The Relational Data Model

✓ What is a data model?

• Important data models

• The relational data model
  • Basic Concepts
  • Constraints
  • Operations

• Basic SQL
What is a Data Model?

- Notation for describing data or information
  - Consisting of
    - **Structure of data**
      - Schema: *Conceptual*
        different from data structure that can be seen as
        Physical data model
      - Defined using a *Data Definition Language* (DDL)
    - Operations on the data
    - Constraints on the data

*From Chapter 3
Kiefer & Bernstein & Lewis book*
What is a Data Model?

- Notation for describing data or information
  - Consisting of
    - Structure of data
    - **Operations** on data
      - **Queries**: Operations that retrieve information
      - **Modifications**: Operations that change the database
      - Described by **Data Manipulation Language** (DML)
        - Different from programming languages
          - Support a set of limited operations
          - Allow for query optimizations
  - Constraints on data
What is a Data Model?

• Notation for describing data or information
  • Consisting of
    • Structure of data
    • Operations on data
    • **Constraints** on data
      • Specify conditions and limitations on the data, and ensure correctness of data: Integrity Constraints
      • Defined using DDL
The Relational Data Model

• What is a data model?

✓ Important data models

• The relational data model
  • Basic Concepts
  • Constraints
  • Operations

• Basic SQL
Data Models

• **Data Model**: Collection of high-level data description constructs that hide many low-level storage details
  
  • Hierarchical Model (Tree)
  
  • Network Model (Graph)
  
  • Object-oriented Model
  
  • **Relational Data Model (Table)**
  
  • Key-value Data Model
  
  • Semi-structured Data Model (Tree)
Data

- Relational
- Key/Value
- Graph
- Document
- Column-Family
- Array/Matrix
- Hierarchical
- Network

NoSQL

Most DBMSs – This Course

Machine Learning

Obsolete - Rare
Hierarchical Model

• Tree-based structure
  • Records: structure that holds data
  • Links: Relation between the records

• IBM Information Management System (IMS)
• Early 1960s

Read more: https://www.db-book.com/db6/appendices-dir/e.pdf
Hierarchical Model

• Example

Supplier
(sNo, sName, sCity, sState)

Part
(pNo, pName, pSize, pQuantity, pPrice)

<table>
<thead>
<tr>
<th>sNo</th>
<th>sName</th>
<th>sCity</th>
<th>sState</th>
<th>parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>First Supplier</td>
<td>New York</td>
<td>NY</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>Second Supplier</td>
<td>SF</td>
<td>CA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pNo</th>
<th>pName</th>
<th>pSize</th>
<th>pQuantity</th>
<th>pPrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>999</td>
<td>Batteries</td>
<td>Large</td>
<td>61</td>
<td>$100</td>
</tr>
<tr>
<td>999</td>
<td>Batteries</td>
<td>Large</td>
<td>14</td>
<td>$99</td>
</tr>
</tbody>
</table>
Network Model

• Graph-based structure
  • Records: structure that holds data
  • Sets: one to many relations between the records

Network Model

• Example

Supplier
(sNo, sName, sCity, sState)

Part
(pNo, pName, pSize)

Supplies

Supplied_By

Supply
(qty, price)
Object-Oriented Model

- Complex data types
  - Non-atomic domains
  - Objects

- 1970s – now

- Few of the original DBMSs in this model exist today, but many of the technologies exist in other forms (XML, JSON)
Object-Oriented Model

• Example

Application Code:

class Student
{
    int id;
    String name;
    String email;
    String phone[];
}

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Adam Smith</td>
<td><a href="mailto:adam@school.edu">adam@school.edu</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>333.333.3333</td>
</tr>
<tr>
<td>1001</td>
<td>444.444.4444</td>
</tr>
</tbody>
</table>
Key-Value Data Model

• Structure of the Data
  • (Key, Value) pairs
  • Domain
    • Key is an integer or string
    • Value can be any object

• Constraints on the data
  • Example
    • Key is unique
    • Value is not NULL

• Query language
  • get(key)
  • put(key, value)
Semi-Structured Data Model

• Semi-structured-data Model
  • Structures: Trees
    • Example
      • XML.
      • Query Language: Xpath
  • Constraints
    • Example
      • <Length> has to be integer
      • Each <Movie> has a <Year> element nested within in

From Chapter 2, The complete book
The Relational Data Model

• What is a data model?

• Important data models

✓ The relational data model
  • Basic Concepts
  • Constraints
  • Operations

• Basic SQL
Preference of Relational Models

• Relational Model provides
  • *Simple, limited* approach to structuring data
    • Yet reasonably versatile, so anything can be modeled
  • A limited collection of operations on data
    • Yet useful collection of operations

• Requirements of large databases
  • Efficiency of access
  • Ease of use: development performance

• Tables **DO NOT** prescribe how they are implemented or stored on disk
  • This is called physical data independence
Relational Data Model

• Single way to represent data:
  • Two-dimensional table called a relation or relation instance

• Relation instances are dynamic
  • We can add, delete, modify tuples

• Attributes: Columns of a relation named by attributes
• Tuples: Rows of a relation other than header (one component)
  • Components of each tuple are atomic. I.e. string, integer, etc., not an structure

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>length</th>
<th>genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gone with the Wind</td>
<td>1939</td>
<td>231</td>
<td>drama</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>sciFi</td>
</tr>
<tr>
<td>Wayne’s World</td>
<td>1992</td>
<td>95</td>
<td>comedy</td>
</tr>
</tbody>
</table>
Relational Data Model

- **Schema**: Name of a relation and set of attributes for a relation
  
  Schema for relation Movies:
  
  $\text{Movies}(\text{title}, \text{year}, \text{length}, \text{genre})$

- **Relation schema**
  
  - The name of a relation: **Movies**
  
  - The set of attributes for a relation
    
    $(\text{title}, \text{year}, \text{length}, \text{genre})$

- Schema changes possible but expensive
  
  - Millions of tuples to be rewritten to add or delete components

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>length</th>
<th>genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gone with the Wind</td>
<td>1939</td>
<td>231</td>
<td>drama</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>sciFi</td>
</tr>
<tr>
<td>Wayne’s World</td>
<td>1992</td>
<td>95</td>
<td>comedy</td>
</tr>
</tbody>
</table>

*The relation Movies*  
*From Chapter 2, The complete book*
The set of schemas for the relations of a database is called a relational database schema, or just a database schema.

Students(studentId, studentName, age, gpa)
Course(courseId, courseName, credit)
Take(studentId, courseId)

or

Movies(title, year, length, genre)
Books(title, year, pages, genre)
Music(title, year, length, genre)
Relational Data Model

- Tables: Relations
- Columns: Attributes, Fields
- Rows: Tuples, Records
- **Degree (arity)** of a relation = number of attributes
- **Cardinality** of a relation = number of tuples

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Mike</td>
<td>21</td>
<td>3.8</td>
</tr>
<tr>
<td>1001</td>
<td>Bill</td>
<td>19</td>
<td>3.4</td>
</tr>
<tr>
<td>1002</td>
<td>Alice</td>
<td>20</td>
<td>3.6</td>
</tr>
</tbody>
</table>
Relational Data Model

• Relations are set of tuples (not a list of tuples)
  • Order does not matter (for rows)
  • Order is important in columns (Change in schema, change in tuples)

Equivalent Representations of a Relation:

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Mike</td>
<td>21</td>
<td>3.8</td>
</tr>
<tr>
<td>1001</td>
<td>Bill</td>
<td>19</td>
<td>3.4</td>
</tr>
<tr>
<td>1002</td>
<td>Alice</td>
<td>20</td>
<td>3.6</td>
</tr>
</tbody>
</table>

= 

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>gpa</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Mike</td>
<td>3.8</td>
<td>21</td>
</tr>
<tr>
<td>1001</td>
<td>Bill</td>
<td>3.4</td>
<td>19</td>
</tr>
<tr>
<td>1002</td>
<td>Alice</td>
<td>3.6</td>
<td>20</td>
</tr>
</tbody>
</table>

= 

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Mike</td>
<td>21</td>
<td>3.8</td>
</tr>
<tr>
<td>1002</td>
<td>Alice</td>
<td>20</td>
<td>3.6</td>
</tr>
<tr>
<td>1001</td>
<td>Bill</td>
<td>19</td>
<td>3.4</td>
</tr>
</tbody>
</table>

How many different ways to represent this relation? 3! x 4! = 144
Relational Data Model

• **Domain**: Associated with each attribute of a relation is a domain. The value of each component should be within the corresponding domain.

• Requirement: Data atomicity of the values in the domain
  • I.e. The relational model does not specify any means for looking into the internal structure of the value, so that the values appear indivisible to the relational operators

• Possible to include the domain or data type for each attribute in a schema:

  \[
  \text{Movies}(\text{title}: \text{string}, \text{year}: \text{integer}, \text{length}: \text{integer}, \text{genre}: \text{string})
  \]

  \[
  \text{Students}(\text{studentId}: \text{integer}, \text{studentName}: \text{string}, \text{age}: \text{integer}, \text{gpa}: \text{real})
  \]
Relational Data Model

- Domain Examples
  - Text: CHAR(20), VARCHAR(50), TEXT
  - Integer: INT, SMALLINT
  - Real: DOUBLE, FLOAT
  - Few more that are vendor specific

\[
\text{Student}(\text{studentId:INT, studentName:VARCHAR(50), age:INT, gpa:FLOAT})
\]
Key

• An attribute or a set of attributes forms a key for a relation if we do not allow two tuples in a relation instance to have the same values in all the attributes of the key. The key is used to uniquely identify a record.

Example: Title and year are the key in movies relation

\[\text{Accounts}(\text{acctNo, type, balance})\]

<table>
<thead>
<tr>
<th>acctNo</th>
<th>type</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>savings</td>
<td>12000</td>
</tr>
<tr>
<td>23456</td>
<td>checking</td>
<td>1000</td>
</tr>
<tr>
<td>34567</td>
<td>savings</td>
<td>25</td>
</tr>
</tbody>
</table>

• Many real-world databases use artificial keys as may not be safe to make assumptions about the values of attributes
  • E.g. Employee ID
Multiple Keys

• A set of attributes form a key

• Example: Title and year are the key in movies relation

\[ \text{Movies}(\text{title}, \text{year}, \text{length}, \text{genre}) \]

• Neither of these attributes are the key alone

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>Genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gone with the Wind</td>
<td>1939</td>
<td>231</td>
<td>drama</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>sciFi</td>
</tr>
<tr>
<td>Wayne's World</td>
<td>1992</td>
<td>95</td>
<td>comedy</td>
</tr>
</tbody>
</table>
Foreign Key

- Attribute(s) whose value is a key of a record in some other relation
  - The attributes used to cross-reference relations need not have the same name
  - A foreign key can consist of more than one attribute

### Accounts

<table>
<thead>
<tr>
<th>acctNo</th>
<th>type</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>savings</td>
<td>12000</td>
</tr>
<tr>
<td>23456</td>
<td>checking</td>
<td>1000</td>
</tr>
<tr>
<td>34567</td>
<td>savings</td>
<td>25</td>
</tr>
</tbody>
</table>

### Customers

<table>
<thead>
<tr>
<th>fname</th>
<th>lname</th>
<th>idNo</th>
<th>account</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robbie</td>
<td>Banks</td>
<td>901-222</td>
<td>12345</td>
</tr>
<tr>
<td>Lena</td>
<td>Hand</td>
<td>805-333</td>
<td>12345</td>
</tr>
<tr>
<td>Lena</td>
<td>Hand</td>
<td>805-333</td>
<td>23456</td>
</tr>
</tbody>
</table>

Foreign key to Accounts: acctNo
Integrity Constraints (IC)

• An instance of a schema is legal if it satisfies all integrity constraints associated with the schema
  • Type constraint
  • Key constraints
  • Referential integrity constraints
  • Semantics constraints
Type Constraint

• To meet the type constraint, a relation instance must satisfy
  • Column naming
    • Matching between column name and associated record attribute
  • Domain constraints
    • The values that appear in each column must belong to the defined domain

| Illegal |
|------------------|--------|--------|--------|
| **Title**   | **Year** | **Length** | **Genre** |
| Gone with the Wind | 1939   | 231    | drama   |
| Star Wars      | 1977   | 124    | sciFi   |
| Wayne’s World  | comedy | 95     | 1992    |

| Illegal |
|------------------|--------|--------|--------|
| **Title**   | **Year** | **Length** | **Genre** |
| Gone with the Wind | 1939   | 231    | drama   |
| Star Wars      | 1977   | 124    | sciFi   |
| Wayne’s World  | 1992   | 95.5   | comedy  |
Key Constraints

• If a relation has several keys, they are referred to as candidate keys. One key is usually used as primary key.

• Key constraint
  • A relation instance of a schema should not contain a pair of distinct tuples whose values agree on all of the attributes in the key.

Illegal

<table>
<thead>
<tr>
<th>acctNo</th>
<th>type</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>savings</td>
<td>12000</td>
</tr>
<tr>
<td>23456</td>
<td>checking</td>
<td>1000</td>
</tr>
<tr>
<td>23456</td>
<td>savings</td>
<td>25</td>
</tr>
</tbody>
</table>
Referential Integrity

• The requirement that referenced tuples must exist is referential integrity
• Foreign key constraints
  • A key is a foreign key for a relation if and only if for every tuple there is a tuple that has the same values over the attributes in the referenced relation as does over the corresponding attributes in the original relation.

<table>
<thead>
<tr>
<th>acctNo</th>
<th>type</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>savings</td>
<td>12000</td>
</tr>
<tr>
<td>23456</td>
<td>checking</td>
<td>1000</td>
</tr>
<tr>
<td>34567</td>
<td>savings</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fname</th>
<th>lname</th>
<th>idNo</th>
<th>account</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robbie</td>
<td>Banks</td>
<td>901-222</td>
<td>12335</td>
</tr>
<tr>
<td>Lena</td>
<td>Hand</td>
<td>805-333</td>
<td>12355</td>
</tr>
<tr>
<td>Lena</td>
<td>Hand</td>
<td>805-333</td>
<td>23456</td>
</tr>
</tbody>
</table>

Illegal foreign key to Accounts: acctNo
Semantic Constraints

• Implement a business rule or convention
• Little to do with structure
• Derived from application domain
• Example
  • Students should not register in more than 5 courses
  • Students should have prerequisites to register for a course
  • No employee should have more than one boss
Enforcing Integrity Constraints

• When integrity violated
  • Transaction aborted
  • Remedial action taken: Reactive constraints

• **Triggers (event-condition-action)**
  • Triggered by an event tests a *condition*
    • Condition does not hold then nothing happens in response to this event
    • Condition is satisfied, the *action is* performed by the DBMS
      • Action could be
        • Modify the effects of the event in some way (e.g. remedial action)
        • Aborting the transaction
      • The action could be any sequence of database operations, including operations not connected in any way to the triggering event
The Relational Data Model

• What is a data model?

• Important data models

• The relational data model
  • Basic Concepts
  • Constraints
  • Operations

✓ Basic SQL
SQL

• **SQL: Structured Query Language**
  - Data Definition Language (DDL)
    The **Data-Definition** sublanguage for declaring database schemas
  - Data Manipulation Language (DML)
    The **Data-Manipulation** sublanguage for querying databases and for modifying the database

• SQL makes a distinction between
  - **Stored relations (Tables)**: Exists in DB and can be modified and queried
  - **Views** (Defined by Computation): Not stored but constructed when needed
  - **Temporary tables**: Constructed by SQL processor when it performs its job of executing queries and data modifications
SQL: Data Types

- Primitive Data Types
  - Character string of fixed or varying Length ([CHAR](n), [VARCHAR](n))
  - Bit Strings ([BIT VARYING](n))
  - Boolean ([BOOLEAN])
  - Integer ([INT], [INTEGER], [SHORTINT])
  - Floating-point ([FLOAT], [REAL], [DOUBLE PRECISION], [DECIMAL], [NUMERIC])
  - Date and Time ([DATE], [TIME])

**NOTE:** SQL keywords (create and table for example) are not case sensitive. Named objects (tables, columns etc.) may be.
SQL: Create

• To create a table use the `CREATE TABLE` statement
  • Specify the table name, field names and domains

```sql
CREATE TABLE Customer (  
    sin    CHAR(11),  
    firstName  CHAR(20),  
    lastName  CHAR(20),  
    age      INTEGER,  
    income   REAL  
)  

CREATE TABLE Movies (  
    title    CHAR(100),  
    year     INT,  
    length   INT,  
    genre    CHAR(10),  
    studioName  CHAR(30),  
    studioAddress  CHAR(50)  
)  
```
SQL: Modify Tables

• Modifying Relation Schemas
  • Delete a relation R: `DROP TABLE R`;

  • Modify Schema of relation R: `ALTER TABLE R`
    • `ADD` followed by an attribute name and its data type
    • `DROP` followed by an attribute name

• Example

  `ALTER TABLE Movie ADD producerCNum INT;`
  `ALTER TABLE Movie DROP studioAddress;`
Default Values

- The value used when no other value is known
- Keyword `DEFAULT`, Value either `NULL` or a Constant
- Example

```sql
gender CHAR(1) DEFAULT '?',
birthdate DATE DEFAULT DATE '0000-00-00',

ALTER TABLE MovieStar ADD phone CHAR(16) DEFAULT 'unlisted';
```
SQL: Keys

• Declaring Keys
  • Declare key when attribute listed in the relation schema
    • Can be used only when the key is a single attribute
  • Or add to the list of items declared in schema an additional declaration that an attribute or set of attributes from the key
    • This method should be used if the key consists of more than one attribute
• Two Declarations
  • Either PRIMARY KEY
  • Or UNIQUE
• Set of attributes $S$ key for relation $R$: Two tuples in $R$ cannot agree on all of the attributes in set $S$, unless one of them is NULL. Violating action rejected.
SQL: Keys

• Primary Key Example

```sql
CREATE TABLE Movies(
    title CHAR(100),
    year INT,
    length INT,
    genre CHAR(10),
    studioName CHAR(30),
    producerCNum INT,
    PRIMARY KEY (title, year)
);
```
SQL: Insert

• To insert a record into an existing table use the **INSERT** statement
  • The list of column names is optional
    • If omitted the values must be in the same order as the columns

```
INSERT INTO Customer(sin, firstName, lastName, age, income)
    VALUES ('111', 'Sam', 'Spade', 23, 65234)
```
SQL: Modify Records

• Use the **UPDATE** statement to modify a record, or records, in a table
  • Note that the **WHERE** statement is evaluated *before* the **SET** statement

• Like **DELETE** the **WHERE** clause specifies which records are to be updated

```
UPDATE Customer
SET age = 37
WHERE sin = '111'
```
SQL: Delete

• To delete a record use the **DELETE** statement
  • The **WHERE** clause specifies the record(s) to be deleted

```sql
DELETE
FROM Customer
WHERE sin = '111'
```

• Be careful, the following SQL query deletes *all* the records in a table

```sql
DELETE
FROM Customer
```
Acknowledgements

I have used materials from the following resources in preparation of this course:

• Database Systems: The Complete Book
• Database Systems (Kifer, Bernstein, Lewis)
• Database System Concepts: [https://www.db-book.com](https://www.db-book.com)
• Course offerings
  • CMPT 354 (Jiannan Wang - SFU): [https://sfu-db.github.io/cmpt354/](https://sfu-db.github.io/cmpt354/)
  • W 4111 (Eugene Wu - Columbia): [https://w4111.github.io/](https://w4111.github.io/)
  • CS 186 (Joe Hellerstein - Berkeley): [https://sites.google.com/site/cs186fall17/](https://sites.google.com/site/cs186fall17/)