Chapter 3
The Relational Data Model and Relational Database Constraints

Chapter Outline
- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- Update Operations and Dealing with Constraint Violations

Relational Model Concepts
- The relational Model of Data is based on the concept of a Relation.
- A Relation is a mathematical concept based on the ideas of sets.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.
- We review the essentials of the relational approach in this chapter.
Relational Model Concepts

- The model was first proposed by Dr. E.F. Codd of IBM in 1970 in the following paper: "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970.

The above paper caused a major revolution in the field of Database management and earned Ted Codd the coveted ACM Turing Award.

Informal Definitions

- RELATION: A table of values
  - A relation may be thought of as a set of rows.
  - A relation may alternately be thought of as a set of columns.
  - Each row represents a fact that corresponds to a real-world entity or relationship.
  - Each row has a value of an item or set of items that uniquely identifies that row in the table.
  - Sometimes row-ids or sequential numbers are assigned to identify the rows in the table.
  - Each column typically is called by its column name or column header or attribute name.
Relations, and Schemas

- A Relation may be defined in multiple ways.
- The Schema of a Relation: \( R \) (A1, A2, ...An)
  
  Relation schema \( R \) is defined over attributes A1, A2, ...An
  
  For Example -
  
  CUSTOMER (Cust-id, Cust-name, Address, Phone#)

  Here, CUSTOMER is a relation defined over the four attributes
  Cust-id, Cust-name, Address, Phone#, each of which has a domain or a set of valid values. For example, the domain of Cust-id is 6 digit numbers.

Tuples

- A tuple is an ordered set of values
- Each value is derived from an appropriate domain.
- Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four values.

  - <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000"> is a tuple belonging to the CUSTOMER relation.

- A relation may be regarded as a set of tuples (rows).
- Columns in a table are also called attributes of the relation.

Domains

- A domain has a logical definition: e.g., "USA_phone_numbers" are the set of 10 digit phone numbers valid in the U.S.
- A domain may have a data-type or a format defined for it. The USA_phone_numbers may have a format: (ddd) ddd-dddd where each d is a decimal/digit. E.g., Dates have various formats such as monthname, date, year or yyyy-mm-dd, or dd, mm,yyyy etc.
- An attribute designates the role played by the domain. E.g., the domain Date may be used to define attributes “Invoice-date” and “Payment-date”.
Relations Defined

- The relation is formed over the Cartesian product of the sets; each set has values from a domain; that domain is used in a specific role which is conveyed by the attribute name.
- For example, attribute Cust-name is defined over the domain of strings of 25 characters. The role these strings play in the CUSTOMER relation is that of the name of customers.
- Formally,
  \[ R(A_1, A_2, \ldots, A_n) \]
  \[ r(R) \quad \text{dom}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n) \]
- \( R \) is the schema of the relation
- \( r(R) \) is a specific “value” or population of \( R \).
- \( R \) is also called the intension of a relation
- \( r \) is also called the extension of a relation

Example

- Let \( S_1 = \{0,1\} \)
- Let \( S_2 = \{a,b,c\} \)
- Let \( R \subseteq S_1 \times S_2 \)
- Then for example: \( r(R) = \{<0,a>, <0,b>, <1,c>\} \)
  is one possible “state” or “population” or “extension” \( r \) of the relation \( R \), defined over domains \( S_1 \) and \( S_2 \). It has three tuples.

Definition Summary

<table>
<thead>
<tr>
<th>Informal Terms</th>
<th>Formal Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Relation</td>
</tr>
<tr>
<td>Column</td>
<td>Attribute/Domain</td>
</tr>
<tr>
<td>Row</td>
<td>Tuple</td>
</tr>
<tr>
<td>Values in a column</td>
<td>Domain</td>
</tr>
<tr>
<td>Table Definition</td>
<td>Schema of a Relation</td>
</tr>
<tr>
<td>Populated Table</td>
<td>Extension</td>
</tr>
</tbody>
</table>
Ordering of tuples in a relation $r(R)$: The tuples are not considered to be ordered, even though they appear to be in the tabular form.

Ordering of attributes in a relation schema $R$ (and of values within each tuple): We will consider the attributes in $R(A_1, A_2, ..., A_n)$ and the values in $t=<v_1, v_2, ..., v_n>$ to be ordered.

Values in a tuple: All values are considered atomic (indivisible). A special null value is used to represent values that are unknown or inapplicable to certain tuples.
Two identical tuples when the order of attributes and values is not part of relation definition

\[ t_1 = \langle \text{Name: Dick Davidson}, \text{Civ: 420-11-2300}, \text{HomePhone: null}, \text{Address: 3452 Elgin Road}, \text{OfficePhone: 749-1253}, \text{Age: 25}, \text{Civ: 3,536} \rangle \]

\[ t_2 = \langle \text{Address: 3452 Elgin Road}, \text{Name: Dick Davidson}, \text{Civ: 420-11-2300}, \text{Age: 25}, \text{OfficePhone: 749-1253}, \text{Civ: 3,536}, \text{HomePhone: null} \rangle \]

Notations for Relations

- Notation:
  - We refer to component values of a tuple \( t \) by \( t[A_i] = v_i \) (the value of attribute \( A_i \) for tuple \( t \)).
  - Similarly, \( t[A_{u}, A_{v}, ..., A_{w}] \) refers to the subtuple of \( t \) containing the values of attributes \( A_u, A_v, ..., A_w \), respectively.

Meaning of a Relation

- A Relation may represent:
  - An Entity
  - A Relationship
Relational model Constraints

- Inherent Model-based constraints
- Relational Integrity Constraints (also known as Schema-based constraints)
- Application-based constraints

Relational Integrity Constraints

- Constraints are conditions that must hold on all valid relation instances. There are three main types of constraints:
  1. Key constraints
  2. Entity integrity constraints
  3. Referential integrity constraints

Key Constraints

- **Superkey of** R: A set of attributes SK of R such that no two tuples in any valid relation instance r(R) will have the same value for SK. That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK].
- **Key of** R: A “minimal” superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

**Example:** The CAR relation schema:

```
CAR(State, Reg#, SerialNo, Make, Model, Year)
```

has two keys Key1 = (State, Reg#), Key2 = (SerialNo), which are also superkeys. (SerialNo, Make) is a superkey but not a key.

- If a relation has several candidate keys, one is chosen arbitrarily to be the primary key. The primary key attributes are underlined.
The CAR relation, with two candidate keys: LicenseNumber and EngineSerialNumber

<table>
<thead>
<tr>
<th>LicenseNumber</th>
<th>EngineSerialNumber</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas 460-785</td>
<td>A00898</td>
<td>Ford</td>
<td>Mustang</td>
<td>96</td>
</tr>
<tr>
<td>Florida 974-A7</td>
<td>B48098</td>
<td>Oldsmobile</td>
<td>Cutlass</td>
<td>99</td>
</tr>
<tr>
<td>New York 154-12B</td>
<td>C80558</td>
<td>Oldsmobile</td>
<td>Delta</td>
<td>96</td>
</tr>
<tr>
<td>California 620-72Y</td>
<td>C62704</td>
<td>Mercedes</td>
<td>SEC-0</td>
<td>99</td>
</tr>
<tr>
<td>California 765-020</td>
<td>Y00355</td>
<td>Toyota</td>
<td>Corry</td>
<td>98</td>
</tr>
<tr>
<td>Texas 105-528</td>
<td>U003980</td>
<td>Jaguar</td>
<td>JUS</td>
<td>98</td>
</tr>
</tbody>
</table>

Entity Integrity

- **Relational Database Schema**: A set $S$ of relation schemas that belong to the same database. $S$ is the name of the database.

  $$S = \{R_1, R_2, \ldots, R_n\}$$

- **Entity Integrity**: The primary key attributes PK of each relation schema $R$ in $S$ cannot have null values in any tuple of $r(R)$. This is because primary key values are used to identify the individual tuples.

- **Note**: Other attributes of $R$ may be similarly constrained to disallow null values, even though they are not members of the primary key.

Schema diagram for the COMPANY relational database
A possible database state for the COMPANY database schema

Referential Integrity

- A constraint involving two relations
- Used to specify a relationship among tuples in two relations: the referencing relation and the referenced relation.
- Tuples in the referencing relation $R_1$ have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation $R_2$. A tuple $t_1$ in $R_1$ is said to reference a tuple $t_2$ in $R_2$ if $t_1[\text{FK}] = t_2[\text{PK}]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1$,FK to $R_2$.

Referential integrity constraints displayed on the COMPANY relational database schema
Referential Integrity Constraint

The value in the foreign key column (or columns) FK of the referencing relation $R_1$ can be either:
1. a value of an existing primary key value of the corresponding primary key PK in the referenced relation $R_2$, or...
2. null
In case (2), the FK in $R_1$ should not be a part of its own primary key.

Other Types of Constraints

Semantic Integrity Constraints:
- application dependent based on application semantics and cannot be expressed by the model per se
- E.g., “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- A constraint specification language may have to be used to express these
- Triggers may be used to handle violations

Other Types of Constraints

Functional Dependency Constraints:
- A dependency between 2 attributes X and Y
- Represented as $X \rightarrow Y$
- This indicates that the value of X determines the value of Y in all states of the relation
Update Operations on Relations

- INSERT a tuple
- DELETE a tuple
- MODIFY a tuple
- Integrity constraints should not be violated by the update operations
- Several update operations may have to be grouped together
- Updates may propagate to cause other updates automatically. This may be necessary to maintain integrity constraints.

In case of integrity violation, several actions can be taken:

- Cancel the operation that causes the violation (REJECT option)
- Perform the operation but inform the user of the violation
- Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
- Execute a user-specified error-correction routine

The Insert Operation

- Provides a list of attribute values for a new tuple \( t \) that is to be inserted into a relation \( R \)
- Can violate any of the four types of constraints
- If an insertion violates one or more constraints
  - Default option is to reject the insertion
The Delete Operation

- Can violate only referential integrity
  - If tuple being deleted is referenced by foreign keys from other tuples
  - **Restrict**
    - Reject the deletion
  - **Cascade**
    - Propagate the deletion by deleting tuples that reference the tuple that is being deleted
  - **Set null or set default**
    - Modify the referencing attribute values that cause the violation

The Update Operation

- Necessary to specify a condition on attributes of relation
  - Select the tuple (or tuples) to be modified
- If attribute not part of a primary key nor of a foreign key
  - Usually causes no problems
- Updating a primary/foreign key
  - Similar issues as with Insert/Delete

The Transaction Concept

- **Transaction**
  - Executing program
  - Includes some database operations
  - Must leave the database in a valid or consistent state
- **Online transaction processing (OLTP) systems**
  - Execute transactions at rates that reach several hundred per second