Chapter (5)
The Relational Data Model, Relational Constraints, and the
Relational Algebra

Objectives
• Describe the basic principals of the relational model of data
• Define the modeling concepts and notation of the relational model
• Learn about the relational constraints
• Define the update operations of the relational model
• Handling the violations of the integrity constraints
• Learn about relational algebra that is used to manipulate relations and
  specifying queries.

The relational model was first introduced by Ted Codd of IBM Research in a 1970 in a classic paper [Codd 1970].

Relational Model Concepts

The relational model represents the database as a collection of relations. A relation is often resembles a table of values or to some extent, a “flat” file of records.

There are important differences between relations and files.

A relation that is thought as a table of values contains rows that represent a collection of related values.

In the formal relational model terminology, a row is called a tuple, a column header is called an attribute, and the table is called a relation.

The data type describing the types of values that can appear in each column is called a domain.
Domains
A domain $D$ is a set of values that are indivisible as far as the relational model is concerned, i.e., it contains a set of atomic values.
Examples: USA_Phone numbers. Which is …
Local_phone_numbers. Which is …
Names: The set of names of persons.
Social_security_numbers: The set of valid 9-digit social security numbers.
GPA: A real value between 0 and 4.
There is a data type of format associated with each domain.

Relation
A relation schema $R$, denoted by $R(A_1, A_2, \ldots, A_n)$ is made up of a relation name $R$ and a list of attributes $A_1, A_2, \ldots, A_n$.
Each attribute $A_i$ is the name of a role played by some domain $D$ in the relation schema $R$.

$D$ is called the domain of $A_i$ and is denoted by $\text{Dom}(A_i)$.

Relation-cont.
- A relation schema is used to describe a relation, $R$ is called the name of this relation.
- The degree of a relation is the number of attributes $n$ of its relation schema.

![Figure 5.1](image_url) The attributes and tuples of a relation STUDENT.

What are the domains?
A relation $r(R)$ is a **mathematical relation** of degree $n$ on the domains $\text{dom}(A_1), \text{dom}(A_2), \ldots, \text{dom}(A_n)$.

$$r(R) \subseteq (\text{dom}(A_1) \times \text{dom}(A_2) \times \cdots \times \text{dom}(A_n))$$

If we denote number of values or **cardinality** of a domain $D$ by $|D|$, and assume that all domains are finite, the total number of tuples in the above Cartesian product is:

$$|\text{dom}(A_1)| \times |\text{dom}(A_2)| \times \cdots \times |\text{dom}(A_n)|$$

This represents many combinations, out of which, a relation state at a given time – the current relation state – reflects only the valid tuples that represent a particular state of the real world.

It is possible that several attributes have the same domain. The attributes indicate different **roles**, or interpretations, for the domain.

---

**Example:**
Suppose we have 4 students each are taking 3 courses. How many tuples do I need to represent all data. Each students are identified with a unique ID.
Relation-cont.

Suppose I had another relation as:

<table>
<thead>
<tr>
<th>COURSE</th>
<th>SSN</th>
<th>Semester</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1440</td>
<td>305-61-2435</td>
<td>4</td>
<td>2001</td>
</tr>
<tr>
<td>CS2440</td>
<td>381-62-1245</td>
<td>3</td>
<td>2001</td>
</tr>
<tr>
<td>CS2440</td>
<td>305-61-2435</td>
<td>1</td>
<td>2002</td>
</tr>
<tr>
<td>CS3460</td>
<td>305-61-2435</td>
<td>2</td>
<td>2002</td>
</tr>
<tr>
<td>CS3430</td>
<td>305-61-2435</td>
<td>3</td>
<td>2002</td>
</tr>
<tr>
<td>CS4465</td>
<td>305-61-2435</td>
<td>4</td>
<td>2002</td>
</tr>
<tr>
<td>CS1440</td>
<td>381-62-1245</td>
<td>1</td>
<td>2002</td>
</tr>
<tr>
<td>CS2440</td>
<td>381-62-1245</td>
<td>2</td>
<td>2002</td>
</tr>
<tr>
<td>CS3460</td>
<td>381-62-1245</td>
<td>3</td>
<td>2002</td>
</tr>
</tbody>
</table>

Suppose you want to list the courses that each student has taken with the semester and year along with all the information in the first relation shown in Figure 5.1. How many tuples do you think you will have?

Characteristics of Relations

There are several characteristics that separate a table from a relation:

1. Ordering of Tuples in a Relation

   The records in a file are in some order, 1st, 2nd, … ith. While in a relation, such ordering does not exist. Tuple ordering is not part of relation definition.

2. Ordering of Values within a Tuple, and an Alternative Definition of a Relation.

   The ordering of values (attributes) in a relation schema definition is important. However, at the logical level, the order of attributes and their values are not really important as long as the correspondence between attributes and values is maintained.

   An alternative definition can be given such that the ordering of tuples in a relation is not necessary.
2. Ordering of Values within a Tuple, and an Alternative Definition of a Relation. – cont.

\[ R = \{ A_1, A_2, ..., A_n \} \]

\[ r(R) : r = \{ t_1, t_2, ..., t_m \}, \text{ where} \]

\[ t_i \leftarrow R \leftrightarrow D \]

\[ D = \text{dom}(A_1) \cup \text{dom}(A_2) \cup \text{dom}(A_3) ... \text{dom}(A_n) \]

In this definition, \( t[A_i] \) must be in \( \text{dom}(A_i) \) for \( 1 \leq i \leq n \) for each mapping \( t \) in \( r \).

According to this definition, a tuple can be considered as a set of \((\text{attributes}, \text{value})\) pairs, where each pair gives the value of the mapping from an attribute \( A_i \) to a value \( v_i \) from \( \text{dom}(A_i) \).

3. Values in the Tuples

Each value is a tuple is an atomic value, i.e., is not divisible into components within the framework of the basic relational model.

Composite and multivalued attributes are not allowed.

Multivalued attributes must be represented by separate relations, and composite attributes are represented only by their simple component attributes.

The values of some attributes within a particular tuple may be unknown or may not apply to that tuple. For such cases a value called null is used.

A null value may have one the following values: “value unknown”, value exists but not available”, or “attribute does not apply to this tuple.”

Incorporating different types of null values into relational model operations has proved difficult.
4. Interpretation of a Relation

The relation schema can be interpreted as a declaration or a type of assertion.

Example: In relation STUDENT (Fig 5.1) each tuple contains the:
Name, SSN, HomePhone, Address, OfficePhone, Age, and GPA.
Thus, each tuple is reflecting a fact or a particular instance of assertion.

Some relations may represent facts about entities, whereas other relations may represent facts about relationships.

Example: A relation schema MAJORS (StudentSSN, DepartmentCode) asserts that students major in academic departments. A tuple in this relation relates a student to his or her major department.

Hence, the relational model represents facts about both entities and relationships uniformly as relations.

5.1 Relational Model Notation

Following notation is commonly used:
• A relation schema R of degree n is denoted by R(A₁, A₂,…,Aₙ).
• An n-tuple t in a relation r(R) is denoted by t = <v₁, v₂, …, vₙ>, where vᵢ is the value corresponding to attribute Aᵢ. The components values of tuples are denoted as:
  • Both t[Aᵢ] and t.Aᵢ refer to the value vᵢ in t for attribute Aᵢ.
  • Both t[Aᵢ, Aᵢ, …, Aᵢ] and t.(Aᵢ, Aᵢ, …, Aᵢ), where Aᵢ, Aᵢ, …, Aᵢ is a list of all attributes from R, refer to the subtuple of values < v₁, v₂, …, vₙ> from t corresponding to the attributes specified in the list.
• The letters Q, R, S denote relation names.
• The letters q, r, s denote relation states
• The letters t, u, v denote tuples
• The name of a relation schema also refers to the current set of tuples in that relation.
• An attribute A can be qualified with the relation name R to which it belongs by using the dot notation R.A. STUDENT.Name, EMPLOYEE.Name.
We will discuss various constraints on data that can be specified on a relational database schema in the form of constraints.

Common constraints are:
- Domain constraints
- Key constraints
- Entity integrity
- Referential integrity

Other types of constraints are:
- Data dependencies (functional and multivalued)
Which is used mainly for design and is discussed in Chapter 14 and 15.

**Domain Constraints**

The value of each attribute A must be an atomic value from the domain $\text{dom}(A)$.

The data types associated with domains typically include standard numeric data types for:
- integers (short-integer, integer, long-integer)
- real numbers (float and double-precision float)

Characters, fixed-length strings, and variable length strings are also available.

Others possibilities:
- Date, Time, Timestamp, and Money data types.
Key Constraints

A relation is a set of tuples. By definition, all elements of a set are distinct, hence, all tuples in a relation must also be distinct.

Suppose we denote a subset of relation R as SK such that no two tuples in any relation r of R have the same combination of values for these attributes.

Then, for any two distinct tuples t₁ and t₂ in a relation state r of R, we have the constraint that:

\[ t₁[SK] \neq t₂[SK] \]

Any such set of attributes SK is called a superkey of the relation schema R.

The superkey specifies a uniqueness constraint that no two distinct tuples in state r of R can have the same value for SK.

Key Constraints – cont.

Every relation has at least one default superkey (the set of all its attributes).

A superkey can have redundant attributes, however, no redundancy is desirable.

A key (K) of a relation schema R is a superkey of R with the additional property that removing any attribute A from K leaves a set of attributes K’ that is not a superkey of R.

Example: Set[SSN] is a key of STUDENT relation. No two students can have the same SSN, thus any set of attributes {SSN, Name, Age} or {SSN, Name, Class} is a superkey. But, neither one of these are the key to STUDENT, because removing Name, Age, or both still leaves the remaining part a superkey.
**Key Constraints – cont.**

A key is time-invariant, which means, once defined at the stage 0, it cannot be changed at the later stage of the relation.

If a relation schema has more than one key each of the keys is called a **candidate key**. One of the candidate keys can be defined as the **primary key**.

Example:

```
CAR
LicenseNumber    EngineSerialNumber    Make   Model   Year
```

Both LicenseNumber and EngineSerialNumber are keys. Either one of these two can be defined as the primary key for that relation.

It is better to choose a primary key with a single attribute or small number of attributes.

Can you give an example?

---

**Relational Databases and Relational Database Schemas**

In general a **relational database schema** $S$ is a set of relation schemas $S = \{R_1, R_2, \ldots, R_m\}$ and a set of **integrity constraints (IC)**.

A **relational database state** $DB$ of $S$ is a set of relation states $DB = \{r_1, r_2, \ldots, r_m\}$ such that each $r_i$ is a state of $R_i$ and such that the $r_i$ relation states satisfy the integrity constraints specified in IC.

Think about the COMPANY relational database schema:

```
COMPANY = \{EMPLOYEE, DEPARTMENT, DEPT_LOCATION, PROJECT, WORKS_ON, DEPENDENT\}
```

Each relational DBMS must have a Data Definition Language (DDL) for defining a relational database schema. Current DBMSs are mostly using SQL.
Integrity, Referential Integrity, and Foreign Keys

The **entity integrity constraint** states that no primary key can be null. Why?

The **referential integrity constraint** is specified between two relations and is used to maintain the consistency among tuples of two relations. You must refer to an existing tuple at all time.

There is another type of constraint called **semantic integrity constraints**. Examples of this type of constraints is the salary of an employee that must be less than that of his/her supervisor. Or maximum number of hours that an employee can work which must be less than 56 hours/week.

These types of constraints can be specified and enforced using a general purpose **constraints specification language**. A mechanism called **triggers** and **assertions** can be used.

Integrity, Referential Integrity, and Foreign Keys – cont.

The previous constraints are also known as **state constraints**, because they define the constraints that a **valid state** of the database must satisfy.

Another type of constraint is called **transition constraint**. This constraint deals with the changes in the database. An example is the salary of employees that can only increase.

The four types of constraints are:
- Domain Constraints
- Key Constraints and Constraints on null
- Entity Integrity
- Referential Integrity
Update Operations and Dealing with Constraint Violations

The main operations of the relational model can be categorized into retrieval and updates.

There are three basic update operations:
- **Insert**
  - is used to insert new tuple(s) in a relation.
- **Delete**
  - is used to delete tuples.
- **Modify**
  - is used to change the values of some attributes in existing tuples.

Whenever an update operation is applied, the integrity constraints specified on the relational database schema should not be violated.

A Quick Note

The insertion, deletion, and update is only possible when the table (entity) which is the target of these modifications already exists. In Chapter (8) we will discuss the creation of tables. Just to give you an idea on how a SQL CREATE command looks like, I have listed the CREATE procedure for EMPLOYEE table below.

```sql
CREATE TABLE EMPLOYEE
    (FNAME VARCHAR(15) NOT NULL,
    MINIT CHAR,
    LNAME VARCHAR(15) NOT NULL,
    SSN CHAR(9) NOT NULL,
    BDATE DATE,
    ADDRESS VARCHAR(30),
    SEX CHAR,
    SALARY DECIMAL(10,2),
    SUPERSSN CHAR(9),
    DNO INT NOT NULL,
    PRIMARY KEY (SSN),
    FOREIGN KEY (SUPPERSSN) REFERENCES EMPLOYEE(SSN),
    FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNUMBER));
```
The Insert Operations

The insert operation provides a list of values for a new tuple \( t \) that is to be inserted into a relation \( R \).

Insert can violate any of the four constraints that we discussed earlier.

- Domain constraint can be violated if an attribute value is given that does not appear in the corresponding domain.


What is violated here?

- Key constraint can be violated if a key value in the new tuple \( t \) already exists in another tuple in the relation \( r(R) \).


What is violated here?

- Entity integrity can be violated if reference is made to an invalid attribute.


What is violated here?

- Referential integrity can be violated if the value of any foreign key in \( t \) refers to a tuple that does not exist in the referenced relation.


What is violated here?
The Insert Operation – cont.

If an insertion violates any of the four constraints, it will be rejected.

In case of rejection, it would be useful if the DBMS could explain to
the user why the insertion was rejected.

Another option is to attempt to correct the problem. This is not
common for Insert, rather for Delete and Update.

Example:
In operation (1) in the previous page, the DBMS could ask user to
provide a value for SSN and accept the insertion if a valid SSN value
were provided.

The Delete Operation
The Delete operation can violate only referential integrity, if the tuple
being deleted is referenced by the foreign keys from other tuples in the
database.
Examples:
1. Delete the WORKS_ON tuple with ESSN = ‘999887777’ and PNO = 10.
Is this deletion accepted?

2. Delete the EMPLOYEE tuple with SSN=‘999887777’.
How about this one?

3. Delete the EMPLOYEE tuple with SSN=‘333445555’,
What kind of problem this deletion will cause?

Three options are available:
1) reject the deletion
2) attempt to cascade (propagate) the deletion to to the references
3) modify the the referencing attribute values that cause the violation
(set to null or changed to reference to another valid tuple).
The Update Operation
The Update is used to change the values of one or more attributes in a tuple (or tuples) of some relation R.

Examples:
1. Update the SALARY of the EMPLOYEE tuple with SSN = ‘999887777’ to 28000.

Will this be accepted?

2. Update the DNO of the EMPLOYEE tuple with SSN=‘999887777’ to 1.

Will this be accepted?

3. Update the DNO of the EMPLOYEE tuple with SSN=‘999887777’ to 7.

Will this be accepted?

4. Update the SSN of the EMPLOYEE tuple with SSN=‘999887777’ to ‘987654321’.

Will this be accepted?