CHAPTER 5

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
  - 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 formal relational model in this chapter
- In practice, there is a standard model based on SQL – this is described in Chapters 6 and 7 as a language
- Note: There are several important differences between the formal model and the practical model, as we shall see
Relational Model Concepts

- A Relation is a mathematical concept based on the ideas of sets.
- The model was first proposed by Dr. E.F. Codd of IBM Research in 1970 in the following paper:
- The above paper caused a major revolution in the field of database management and earned Dr. Codd the coveted ACM Turing Award.
Informal Definitions

- Informally, a relation looks like a table of values.

- A relation typically contains a set of rows.

- The data elements in each row represent certain facts that correspond to a real-world entity or relationship.
  - In the formal model, rows are called tuples.

- Each column has a column header that gives an indication of the meaning of the data items in that column.
  - In the formal model, the column header is called an attribute name (or just attribute).
Example of a Relation

<table>
<thead>
<tr>
<th>Name</th>
<th>Ssn</th>
<th>Home_phone</th>
<th>Address</th>
<th>Office_phone</th>
<th>Age</th>
<th>Gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benjamin Bayer</td>
<td>305-61-2435</td>
<td>373-1616</td>
<td>2918 Bluebonnet Lane</td>
<td>NULL</td>
<td>19</td>
<td>3.21</td>
</tr>
<tr>
<td>Chung-cha Kim</td>
<td>381-62-1245</td>
<td>375-4409</td>
<td>125 Kirby Road</td>
<td>NULL</td>
<td>18</td>
<td>2.89</td>
</tr>
<tr>
<td>Dick Davidson</td>
<td>422-11-2320</td>
<td>NULL</td>
<td>3452 Elgin Road</td>
<td>749-1253</td>
<td>25</td>
<td>3.53</td>
</tr>
<tr>
<td>Rohan Panchal</td>
<td>489-22-1100</td>
<td>376-9821</td>
<td>265 Lark Lane</td>
<td>749-6492</td>
<td>28</td>
<td>3.93</td>
</tr>
<tr>
<td>Barbara Benson</td>
<td>533-69-1238</td>
<td>839-8461</td>
<td>7384 Fontana Lane</td>
<td>NULL</td>
<td>19</td>
<td>3.25</td>
</tr>
</tbody>
</table>

**Figure 5.1**

The attributes and tuples of a relation STUDENT.
Informal Definitions

- **Key of a Relation:**
  - Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
  - Called the *key*
  - In the STUDENT table, SSN is the key

- Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
  - Called *artificial key* or *surrogate key*
Formal Definitions - Schema

- **The Schema (or description) of a Relation:**
  - Denoted by $R(A_1, A_2, \ldots, A_n)$
  - $R$ is the **name** of the relation
  - The **attributes** of the relation are $A_1, A_2, \ldots, A_n$

- **Example:**
  - CUSTOMER (Cust-id, Cust-name, Address, Phone#)
    - CUSTOMER is the relation name
    - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#

- **Each attribute has a domain or a set of valid values.**
  - For example, the domain of Cust-id is 6 digit numbers.
Formal Definitions - Tuple

- A **tuple** is an **ordered** set of values (enclosed in angled brackets ‘< … >’)
- Each value is derived from an appropriate domain.
- A row in the CUSTOMER relation is a **4-tuple** and would consist of four values, for example:
  - `<632895, "John Smith", "101 Main St. Atlanta, GA 30332", 
    "(404) 894-2000">`
  - This is called a 4-tuple as it has 4 values
  - A tuple (row) in the CUSTOMER relation.
- A relation is a **set** of such tuples (rows)
Formal Definitions - Domain

- A **domain** has a **logical definition**:
  - Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.

- A domain also has 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.
  - Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.

- **Cardinality**: total number of values in domain

- The **attribute name** designates the **role** played by a domain in a relation:
  - Used to interpret the meaning of the data elements corresponding to that attribute
  - Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings
Formal Definitions - State

- The **relation state** is a **subset** of the **Cartesian product** of the domains of its attributes
  - each domain contains the set of **all possible values** the attribute can take.
- Example: attribute **Cust-name** is defined over the domain of character strings of maximum length 25
  - dom(Cust-name) is varchar(25)
- The **role** these strings play in the CUSTOMER relation is that of the **name of a customer**.
Formal Definitions - Summary

- Formally,
  - Given $R(A_1, A_2, \ldots, A_n)$
  - $r(R) \subseteq \text{dom}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n)$
  - $R(A_1, A_2, \ldots, A_n)$ is the **schema** of the relation
  - $R$ is the **name** of the relation
  - $A_1, A_2, \ldots, A_n$ are the **attributes** of the relation
  - $r(R)$: a specific **state** (or "value" or “population”) of relation $R$ – this is a **set of tuples** (rows)
    - $r(R) = \{t_1, t_2, \ldots, t_m\}$ where each $t_i$ is an $n$-tuple
    - $t_i = <v_1, v_2, \ldots, v_n>$ where each $v_j$ *element-of* $\text{dom}(A_j)$
Formal Definitions - Example

- Let $R(A1, A2)$ be a relation schema:
  - Let $\text{dom}(A1) = \{0,1\}$
  - Let $\text{dom}(A2) = \{a,b,c\}$
- Then: $\text{dom}(A1) \times \text{dom}(A2)$ is all possible combinations:
  $$\{<0,a>, <0,b>, <0,c>, <1,a>, <1,b>, <1,c>\}$$

- The relation state $r(R) \subseteq \text{dom}(A1) \times \text{dom}(A2)$
- For example: $r(R)$ could be $\{<0,a>, <0,b>, <1,c>\}$
  - this is one possible state (or “population” or “extension”) $r$ of the relation $R$, defined over $A1$ and $A2$.
  - It has three 2-tuples: $<0,a>$, $<0,b>$, $<1,c>$
## 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 Header</td>
<td>Attribute</td>
</tr>
<tr>
<td>All possible Column Values</td>
<td>Domain</td>
</tr>
<tr>
<td>Row</td>
<td>Tuple</td>
</tr>
<tr>
<td>Table Definition</td>
<td>Schema of a Relation</td>
</tr>
<tr>
<td>Populated Table</td>
<td>State of the Relation</td>
</tr>
</tbody>
</table>
Example – A relation STUDENT

<table>
<thead>
<tr>
<th>Name</th>
<th>Ssn</th>
<th>Home_phone</th>
<th>Address</th>
<th>Office_phone</th>
<th>Age</th>
<th>Gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benjamin Bayer</td>
<td>305-61-2435</td>
<td>373-1616</td>
<td>2918 Bluebonnet Lane</td>
<td>NULL</td>
<td>19</td>
<td>3.21</td>
</tr>
<tr>
<td>Chung-cha Kim</td>
<td>381-62-1245</td>
<td>375-4409</td>
<td>125 Kirby Road</td>
<td>NULL</td>
<td>18</td>
<td>2.89</td>
</tr>
<tr>
<td>Dick Davidson</td>
<td>422-11-2320</td>
<td>NULL</td>
<td>3452 Elgin Road</td>
<td>749-1253</td>
<td>25</td>
<td>3.53</td>
</tr>
<tr>
<td>Rohan Panchal</td>
<td>489-22-1100</td>
<td>376-9821</td>
<td>265 Lark Lane</td>
<td>749-6492</td>
<td>28</td>
<td>3.93</td>
</tr>
<tr>
<td>Barbara Benson</td>
<td>533-69-1238</td>
<td>839-8461</td>
<td>7384 Fontana Lane</td>
<td>NULL</td>
<td>19</td>
<td>3.25</td>
</tr>
</tbody>
</table>

*Figure 5.1*

The attributes and tuples of a relation STUDENT.
Characteristics Of Relations

- **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(A1, A2, ..., An) and the values in t=<v1, v2, ..., vn> to be ordered.
  - (However, a more general alternative definition of relation does not require this ordering. It includes both the name and the value for each of the attributes).
  - Example: t= { <name, “John” >, <SSN, 123456789> }
  - This representation may be called as “self-describing”.
Same state as previous Figure (but with different order of tuples)

**Figure 5.2**
The relation STUDENT from Figure 5.1 with a different order of tuples.

<table>
<thead>
<tr>
<th>Name</th>
<th>Ssn</th>
<th>Home_phone</th>
<th>Address</th>
<th>Office_phone</th>
<th>Age</th>
<th>Gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dick Davidson</td>
<td>422-11-2320</td>
<td>NULL</td>
<td>3452 Elgin Road</td>
<td>749-1253</td>
<td>25</td>
<td>3.53</td>
</tr>
<tr>
<td>Barbara Benson</td>
<td>533-69-1238</td>
<td>839-8461</td>
<td>7384 Fontana Lane</td>
<td>NULL</td>
<td>19</td>
<td>3.25</td>
</tr>
<tr>
<td>Rohan Panchal</td>
<td>489-22-1100</td>
<td>376-9821</td>
<td>265 Lark Lane</td>
<td>749-6492</td>
<td>28</td>
<td>3.93</td>
</tr>
<tr>
<td>Chung-cha Kim</td>
<td>381-62-1245</td>
<td>375-4409</td>
<td>125 Kirby Road</td>
<td>NULL</td>
<td>18</td>
<td>2.89</td>
</tr>
<tr>
<td>Benjamin Bayer</td>
<td>305-61-2435</td>
<td>373-1616</td>
<td>2918 Bluebonnet Lane</td>
<td>NULL</td>
<td>19</td>
<td>3.21</td>
</tr>
</tbody>
</table>
Characteristics Of Relations

- **Values in a tuple:**
  - All values are considered **atomic** (indivisible).
  - Each value in a tuple must be from the domain of the attribute for that column.
    - If tuple \( t = <v_1, v_2, ..., v_n> \) is a tuple (row) in the relation state \( r \) of \( R(A_1, A_2, ..., A_n) \)
      - Then each \( v_i \) must be a value from \( \text{dom}(A_i) \)

- A special **null** value is used to represent values that are **unknown** or **not available** (value exist) or **inapplicable** (value undefined) in certain tuples.
Characteristics Of Relations

- **Notation:**
  - We refer to *component values* of a tuple t by:
    - t[Ai] or t.Ai
    - This is the value vi of attribute Ai for tuple t
  - Similarly, t[Au, Av, ..., Aw] refers to the subtuple of t containing the values of attributes Au, Av, ..., Aw, respectively in t
Constraints determine which values are permissible and which are not in the database.

- Restrictions on the actual values in a database state
- Derived from the rules in the miniworld that the database represents

They are of three main types:

1. **Inherent or Implicit Constraints**: These are based on the data model itself. (E.g., relational model does not allow a list as a value for any attribute, i.e., atomic attribute value, no duplicate tuples)

2. **Schema-based or Explicit Constraints**: They are expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model, domain constraints, key constraints, null constraints…)

3. **Application based or semantic constraints**: These are beyond the expressive power of the model and must be specified and enforced by the application programs.
Relational Integrity Constraints

- Constraints are **conditions** that must hold on all valid relation states.

- There are three *main types* of (explicit schema-based) constraints that can be expressed in the relational model:
  - **Key** constraints
  - **Entity integrity** constraints
  - **Referential integrity** constraints

- Another schema-based constraint is the **domain** constraint
  - Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)
Key Constraints

- **Superkey of R:**
  - Is a set of attributes SK of R with the following condition:
    - No two tuples in any valid relation state 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]
    - This condition must hold in any valid state r(R)

- **Key of R:**
  - A "minimal" superkey
  - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)
  - A Key is a Superkey but not vice versa
Key Constraints (continued)

- **Example:** Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - CAR has two keys:
    - Key1 = {State, Reg#}
    - Key2 = {SerialNo}
  - Both are also superkeys of CAR
  - {SerialNo, Make} is a superkey but **not** a key.

- **In general:**
  - Any *key* is a superkey (but not vice versa)
  - Any set of attributes that *includes a key* is a superkey
  - A *minimal* superkey is also a key
Key Constraints (continued)

- If a relation has several candidate keys, one is chosen arbitrarily to be the primary key.
  - The primary key attributes are underlined.
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to uniquely identify each tuple in a relation
  - Provides the tuple identity
- Also used to reference the tuple from another tuple
  - General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
  - Not always applicable – choice is sometimes subjective
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

### CAR

<table>
<thead>
<tr>
<th>License_number</th>
<th>Engine_serial_number</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas ABC-739</td>
<td>A69352</td>
<td>Ford</td>
<td>Mustang</td>
<td>02</td>
</tr>
<tr>
<td>Florida TVP-347</td>
<td>B43696</td>
<td>Oldsmobile</td>
<td>Cutlass</td>
<td>05</td>
</tr>
<tr>
<td>New York MPO-22</td>
<td>X83554</td>
<td>Oldsmobile</td>
<td>Delta</td>
<td>01</td>
</tr>
<tr>
<td>California 432-TFY</td>
<td>C43742</td>
<td>Mercedes</td>
<td>190-D</td>
<td>99</td>
</tr>
<tr>
<td>California RSK-629</td>
<td>Y82935</td>
<td>Toyota</td>
<td>Camry</td>
<td>04</td>
</tr>
<tr>
<td>Texas RSK-629</td>
<td>U028365</td>
<td>Jaguar</td>
<td>XJS</td>
<td>04</td>
</tr>
</tbody>
</table>
Relational Database Schema

- **Relational Database Schema:**
  - $S = \{R_1, R_2, \ldots, R_n\}$ and a set of **integrity constraints** (IC)
  - $S$ is the name of the whole **database schema**
  - $R_1, R_2, \ldots, R_n$ are the names of the individual **relation schemas** within the database $S$
    - A set $S$ of relation schemas $\{R_1, R_2, \ldots, R_n\}$ that belong to the same database.
- Following slide shows a **COMPANY database schema** with 6 relation schemas
### COMPANY Database Schema

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>[ Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno ]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>[ Dname, Dnumber, Mgr_ssn, Mgr_start_date ]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DEPT_LOCATIONS</th>
<th>[ Dnumber, Dlocation ]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PROJECT</th>
<th>[ Pname, Pnumber, Plocation, Dnum ]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>WORKS_ON</th>
<th>[ Essn, Pno, Hours ]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DEPENDENT</th>
<th>[ Essn, Dependent_name, Sex, Bdate, Relationship ]</th>
</tr>
</thead>
</table>

**Figure 5.5**
Schema diagram for the COMPANY relational database schema.
A relational database state DB of S is a set of relation states DB = {r₁, r₂, ..., rₘ} such that each rᵢ is a state of Rᵢ and such that the rᵢ relation states satisfy the integrity constraints specified in IC.

A relational database state is sometimes called a relational database snapshot or instance.

We will not use the term instance since it also applies to single tuples.

A database state that does not meet the constraints is an invalid state.
Populated database state

- Each relation will have many tuples in its current relation state
- The relational database state is a union of all the individual relation states
- Whenever the database is changed, a new state arises
- Basic operations for changing the database:
  - INSERT a new tuple in a relation
  - DELETE an existing tuple from a relation
  - MODIFY an attribute of an existing tuple
- Next slide (Fig. 5.6) shows an example state for the COMPANY database schema shown in Fig. 5.5.
### Populated database state for COMPANY

#### EMPLOYEE

<table>
<thead>
<tr>
<th>Name</th>
<th>First</th>
<th>Last</th>
<th>SSN</th>
<th>Date</th>
<th>Address</th>
<th>Sex</th>
<th>Salary</th>
<th>Super_SS</th>
<th>DOE</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>B</td>
<td>Smith</td>
<td>123456789</td>
<td>1955-01-09</td>
<td>731 Fondren, Houston, TX</td>
<td>M</td>
<td>300000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Franklin</td>
<td>T</td>
<td>Wang</td>
<td>333445555</td>
<td>1955-12-08</td>
<td>838 Voss, Houston, TX</td>
<td>M</td>
<td>400000</td>
<td>888665555</td>
<td>5</td>
</tr>
<tr>
<td>Alicia</td>
<td>J</td>
<td>Zelaya</td>
<td>987654321</td>
<td>1956-01-19</td>
<td>3231 Castle, Spring, TX</td>
<td>F</td>
<td>250000</td>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>Jennifer</td>
<td>S</td>
<td>Wallace</td>
<td>987654321</td>
<td>1941-06-20</td>
<td>291 Berry, Bellaire, TX</td>
<td>F</td>
<td>430000</td>
<td>888665555</td>
<td>4</td>
</tr>
<tr>
<td>Ramesh</td>
<td>K</td>
<td>Narayan</td>
<td>868884444</td>
<td>1962-05-15</td>
<td>075 Fire Oak, Humble, TX</td>
<td>M</td>
<td>300000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Joyce</td>
<td>A</td>
<td>English</td>
<td>453453453</td>
<td>1972-07-31</td>
<td>5631 Rice, Houston, TX</td>
<td>F</td>
<td>250000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Ahmad</td>
<td>V</td>
<td>Jabbar</td>
<td>987987987</td>
<td>1989-03-29</td>
<td>980 Dallas, Houston, TX</td>
<td>M</td>
<td>250000</td>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>James</td>
<td>E</td>
<td>Borg</td>
<td>888665555</td>
<td>1937-11-10</td>
<td>450 Stone, Houston, TX</td>
<td>M</td>
<td>550000</td>
<td>NULL</td>
<td>1</td>
</tr>
</tbody>
</table>

#### DEPARTMENT

<table>
<thead>
<tr>
<th>Name</th>
<th>Dnumber</th>
<th>Mgr_SS</th>
<th>Mgr_startdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>1988-05-22</td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
<td>1995-01-01</td>
</tr>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>888665555</td>
<td>1981-06-19</td>
</tr>
</tbody>
</table>

#### DEPT_LOCATIONS

- Dnumber: 1
  - Location: Houston
- Dnumber: 4
  - Location: Stafford
- Dnumber: 5
  - Location: Bellaire
- Dnumber: 5
  - Location: Sugarland
- Dnumber: 5
  - Location: Houston

#### PROJECT

<table>
<thead>
<tr>
<th>Name</th>
<th>Dnumber</th>
<th>Location</th>
<th>Drum</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductX</td>
<td>1</td>
<td>Bellaire</td>
<td>5</td>
</tr>
<tr>
<td>ProductY</td>
<td>2</td>
<td>Sugarland</td>
<td>5</td>
</tr>
<tr>
<td>ProductZ</td>
<td>3</td>
<td>Houston</td>
<td>5</td>
</tr>
<tr>
<td>Computerization</td>
<td>10</td>
<td>Stafford</td>
<td>4</td>
</tr>
<tr>
<td>Reorganization</td>
<td>20</td>
<td>Houston</td>
<td>1</td>
</tr>
<tr>
<td>Newbenefits</td>
<td>30</td>
<td>Stafford</td>
<td>4</td>
</tr>
</tbody>
</table>

#### DEPENDENT

<table>
<thead>
<tr>
<th>Essn</th>
<th>Dependent_name</th>
<th>Sex</th>
<th>Date</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>333445555</td>
<td>Alice</td>
<td>F</td>
<td>1986-04-05</td>
<td>Daughter</td>
</tr>
<tr>
<td>333445555</td>
<td>Theodore</td>
<td>M</td>
<td>1982-10-25</td>
<td>Son</td>
</tr>
<tr>
<td>333445555</td>
<td>Joy</td>
<td>F</td>
<td>1958-05-03</td>
<td>Spouse</td>
</tr>
<tr>
<td>987654321</td>
<td>Abner</td>
<td>M</td>
<td>1942-02-28</td>
<td>Spouse</td>
</tr>
<tr>
<td>123456789</td>
<td>Michael</td>
<td>M</td>
<td>1988-01-04</td>
<td>Son</td>
</tr>
<tr>
<td>123456789</td>
<td>Alice</td>
<td>F</td>
<td>1968-12-30</td>
<td>Daughter</td>
</tr>
<tr>
<td>123456789</td>
<td>Elizabeth</td>
<td>F</td>
<td>1967-05-05</td>
<td>Spouse</td>
</tr>
</tbody>
</table>
Entity Integrity

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.
  - t[PK] \( \neq \) null for any tuple t in r(R)
  - If PK has several attributes, null is not allowed in any of these attributes

- Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key. This is called **NOT NULL constraint**.
Referential Integrity

- A constraint involving two relations
  - The previous constraints involve a single relation.
- Used to specify a relationship among tuples in two relations:
  - The referencing relation and the referenced relation.
- Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.
Referential Integrity

- Tuples in the referencing relation R1 have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation R2.
  - A tuple t1 in R1 is said to reference a tuple t2 in R2 if t1[FK] = t2[PK].

- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2. (see Fig. 5.7)
Referential Integrity (or foreign key) Constraint

- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the referencing relation R1 can be either:
    1. a value of an existing primary key value of a corresponding primary key PK in the referenced relation R2, or
    2. a null
  - In case (2), the FK in R1 should not be a part of its own primary key.
Displaying a relational database schema and its constraints

- Each relation schema can be displayed as a row of attribute names.
- The name of the relation is written above the attribute names.
- The primary key attribute (or attributes) will be underlined.
- A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table.
  - Can also point the primary key of the referenced relation for clarity.
- Next slide shows the COMPANY relational schema diagram with referential integrity constraints.
Referential Integrity Constraints for COMPANY database

**Figure 5.7**
Referrential integrity constraints displayed on the COMPANY relational database schema.
Other Types of Constraints

- **Semantic Integrity Constraints:**
  - based on application semantics and cannot be expressed by the model per se (model self)
  - Example: “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

- SQL-99 allows **CREATE TRIGGER** and **CREATE ASSERTION** to express some of these semantic constraints

- Keys, Permissibility of Null values, Candidate Keys (Unique in SQL), Foreign Keys, Referential Integrity etc. are expressed by the **CREATE TABLE** statement in SQL.
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.
Update Operations on Relations

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (RESTRICT or 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
Possible violations for each operation

- **INSERT** may violate any of the constraints:
  - **Domain constraint:**
    - if one of the attribute values provided for the new tuple is not of the specified attribute domain
  - **Key constraint:**
    - if the value of a key attribute in the new tuple already exists in another tuple in the relation
  - **Referential integrity:**
    - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
  - **Entity integrity:**
    - if the primary key value is `null` in the new tuple
Possible violations for each operation

- DELETE may violate only **referential integrity**:  
  - If the **primary key** value of the tuple being deleted is ** referenced** from other tuples in the database  
    - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL (see Chapter 6 for more details)  
      - RESTRICT option: reject the deletion  
      - CASCADE option:  
        - attempt to cascade the deletion by deleting tuples that **reference** the tuple being deleted (ON CASCADE DELETE)  
        - propagate the new primary key value into the **foreign keys** of the referencing tuples (ON CASCADE UPDATE)  
      - SET NULL option: set the **foreign keys** of the referencing tuples to NULL

- One of the above options must be specified during database design for each foreign key constraint

modify the referencing attributes that cause the violation
Possible violations for each operation

- UPDATE may violate **domain constraint** and **NOT NULL constraint** on an attribute being modified
- Any of the other constraints may also be violated, depending on the attribute being updated:
  - Updating the primary key (PK):
    - Similar to a DELETE followed by an INSERT
    - Can violate **key constraints** and **referential integrity constraints**
    - Need to specify similar options to DELETE
  - Updating a foreign key (FK):
    - May violate **referential integrity**
  - Updating an ordinary attribute (neither PK nor FK):
    - Can only violate **domain constraints** (or **NOT NULL constraints**)

Copyright © 2017 Ramez Elmasri and Shamkant B. Navathe
Summary

- Presented Relational Model Concepts
  - Definitions
  - Characteristics of relations
- Discussed Relational Model Constraints and Relational Database Schemas
  - Domain constraints (not NULL constraint)
  - Key constraints
  - Entity integrity
  - Referential integrity
- Described the Relational Update Operations and Dealing with Constraint Violations