

Fundamentals of  
**DATABASE  
SYSTEMS**

FOURTH EDITION

ELMASRI  NAVATHE

## **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.
- 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.

## FORMAL DEFINITIONS

- A **Relation** may be defined in multiple ways.
- The **Schema** of a Relation:  $R (A_1, A_2, \dots, A_n)$   
Relation schema  $R$  is defined over **attributes**  $A_1, A_2, \dots, A_n$   
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.

## FORMAL DEFINITIONS

- 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.
- $\langle 632895, \text{"John Smith"}, \text{"101 Main St. Atlanta, GA 30332"}, \text{"(404) 894-2000"} \rangle$  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.

## FORMAL DEFINITIONS

- 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”.

## FORMAL DEFINITIONS

- 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,  
Given  $R(A_1, A_2, \dots, A_n)$   
 $r(R) \subset \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$
- R: schema of the relation
- r of R: 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

## FORMAL DEFINITIONS

- Let  $S1 = \{0,1\}$
- Let  $S2 = \{a,b,c\}$
- Let  $R \subset S1 \times S2$
- Then for example:  $r(R) = \{ \langle 0,a \rangle, \langle 0,b \rangle, \langle 1,c \rangle \}$   
is one possible “state” or “population” or  
“extension”  $r$  of the relation  $R$ , defined over domains  
 $S1$  and  $S2$ . It has three tuples.

## DEFINITION SUMMARY

### Informal Terms

Table  
Column  
Row  
Values in a column  
Table Definition  
Populated Table

### Formal Terms

Relation  
Attribute/Domain  
Tuple  
Domain  
Schema of a Relation  
Extension

## Example - Figure 5.1

STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

## 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  $R(A_1, A_2, \dots, A_n)$  and the values in  $t = \langle v_1, v_2, \dots, v_n \rangle$  to be *ordered*.  
(However, a more general *alternative definition* of relation does not require this ordering).
- **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.

## CHARACTERISTICS OF 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,  $t[A_u, A_v, \dots, A_w]$  refers to the subtuple of  $t$  containing the values of attributes  $A_u, A_v, \dots, A_w$ , respectively.

## CHARACTERISTICS OF RELATIONS- Figure 5.2

STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21

## 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  $t_1$  and  $t_2$  in  $r(R)$ ,  $t_1[SK] \neq t_2[SK]$ .
- **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  $Key_1 = \{State, Reg\# \}$ ,  $Key_2 = \{SerialNo \}$ , 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*.

# Key Constraints

Figure 5.4 The CAR relation with two candidate keys: LicenseNumber and EngineSerialNumber.

CAR	LicenseNumber	EngineSerialNumber	Make	Model	Year
Texas ABC-739		A69352	Ford	Mustang	96
Florida TVP-347		B43696	Oldsmobile	Cutlass	99
New York MPO-22		X83554	Oldsmobile	Delta	95
California 432-TFY		C43742	Mercedes	190-D	93
California RSK-629		Y82935	Toyota	Carny	98
Texas RSK-629		U028365	Jaguar	XJS	98

© Addison Wesley Longman, Inc. 2000, Elmasri/Navathe, Fundamentals of Database Systems, Third Edition

# Entity Integrity

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

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

- **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.

$$t[\text{PK}] \neq \text{null for any tuple } t \text{ in } r(R)$$

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

## 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**.
- 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 Constraint

### Statement of the constraint

The value in the foreign key column (or columns) FK of the 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) a null.

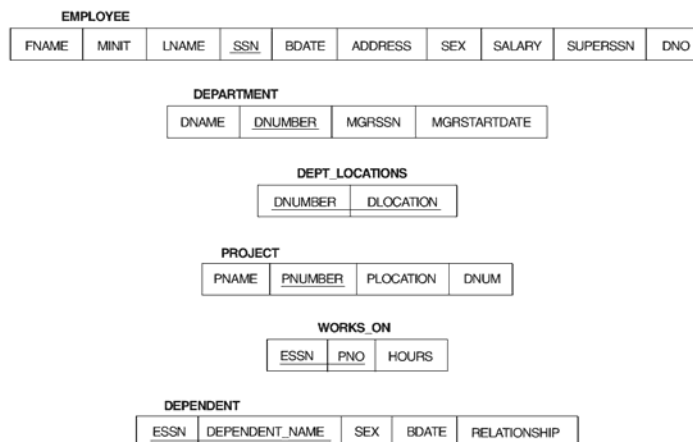
In case (2), the FK in  $R_1$  should not be a part of its own primary key.

## Other Types of Constraints

Semantic Integrity Constraints:

- 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
- SQL-99 allows triggers and ASSERTIONS to allow for some of these

Figure 5.5 Schema diagram for the COMPANY relational database schema; the primary keys are underlined.





## 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 (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

# In-Class Exercise

(Taken from Exercise 5.15)

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(SSN, Name, Major, Bdate)

COURSE(Course#, Cname, Dept)

ENROLL(SSN, Course#, Quarter, Grade)

BOOK\_ADOPTION(Course#, Quarter, Book\_ISBN)

TEXT(Book\_ISBN, Book\_Title, Publisher, Author)

**Draw a relational schema diagram specifying the foreign keys for this schema.**