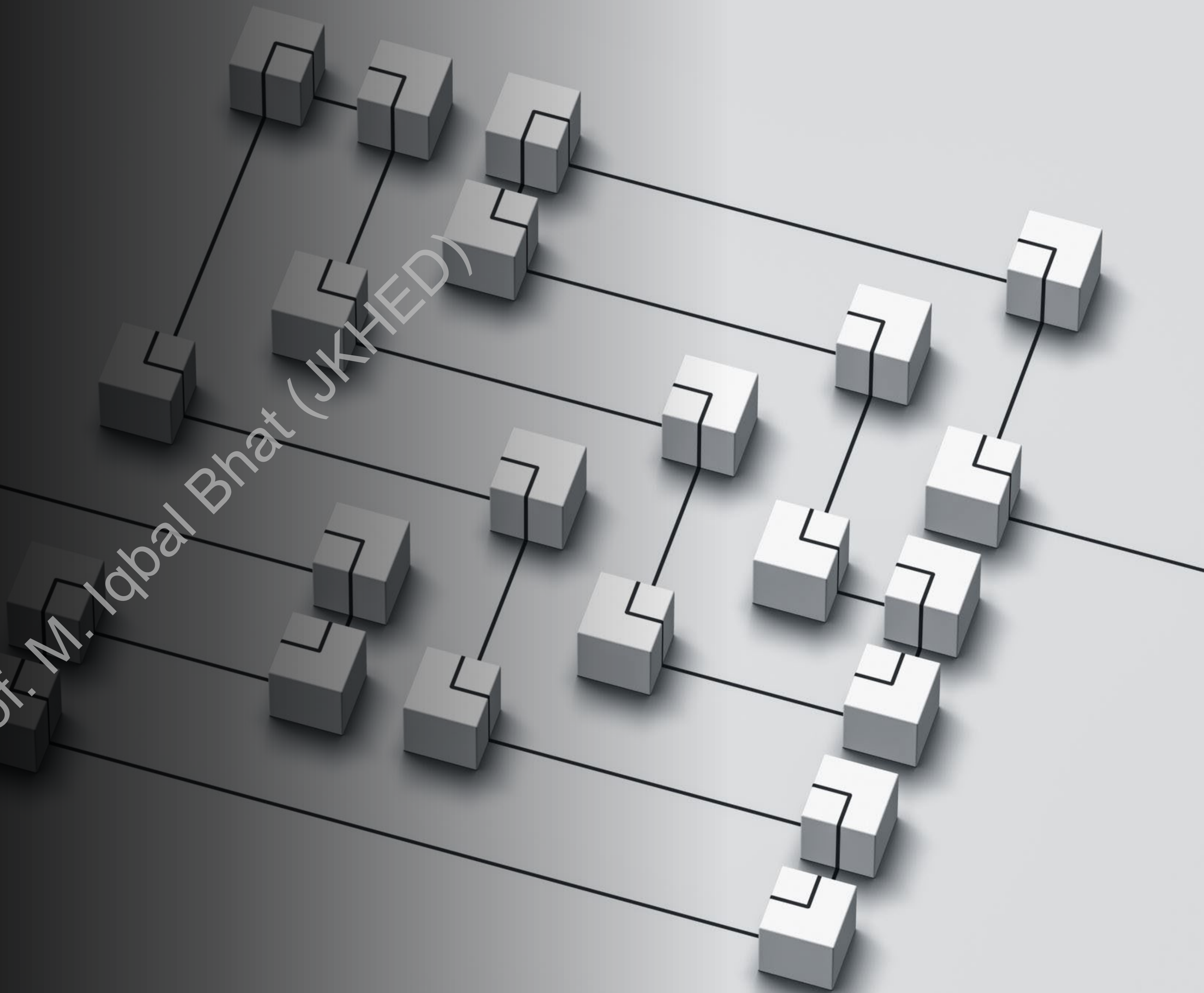
Relational Data Model

By
Prof. M. Iqbal Bhat
Government Degree College Beerwah



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

The Relational Data Model is a method of organizing and representing data in tables (also known as relations) that are related to each other.

It was first proposed by Edgar F. Codd in 1970 as a way to manage data in a more flexible and efficient manner.

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:

- "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 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

Relation Name

STUDENT

Attributes

Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	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
Rohan Panchal	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

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(A1, A2, ...An)

R is the **name** of the relation

The **attributes** of the relation are A1, A2, ..., An

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.

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(A1, A2, ..., An)
  - r(R) $\subset$ dom (A1) X dom (A2) X ....X dom(An)

R(A1, A2, …, An) is the **schema** of the relation

R is the **name** of the relation

A1, A2, …, An 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) = {t1, t2, …, tn} where each ti is an n-tuple
- ti = <v1, v2, …, vn> where each vj *element-of* dom(Aj)

Formal Definitions - Example

Let R(A1, A2) be a relation schema:

Let dom(A1) = {0,1}

Let dom(A2) = {a,b,c}

Then: dom(A1) X dom(A2) is all possible combinations:

{<0,a> , <0,b> , <0,c>, <1,a>, <1,b>, <1,c> }

The relation state r(R) $\subset$ dom(A1) X 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>

Informal Terms		Formal Terms
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Definition Summary

Example – A relation STUDENT

Relation Name

Attributes

STUDENT

Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	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
Rohan Panchal	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

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.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	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
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Chung-cha Kim	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

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 = <v1, v2, …, vn> is a tuple (row) in the relation state r of R(A1, A2, …, An)
    - Then each *vi* must be a value from *dom(Ai)*

  - A special **null** value is used to represent values that are unknown or not available or inapplicable 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

Constraints determine which values are permissible and which are not in the database.

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)

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)

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] $\neq$ 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 <u>underlined</u>.
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, <u>SerialNo</u>, 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

CAR table with two candidate keys – LicenseNumber chosen as Primary Key

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 5.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

Relational Database Schema

- **Relational Database Schema:**
  - A set S of relation schemas that belong to the same database.
  - S is the name of the whole **database schema**
  - S = {R1, R2, ..., Rn} and a set IC of integrity constraints.
  - R1, R2, …, Rn are the names of the individual **relation schemas** within the database S
- Following slide shows a COMPANY database schema with 6 relation schemas

COMPANY Database Schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Relational Database State

- A **relational database state** DB of S is a set of relation states DB = $\{r_1, r_2, ..., r_m\}$ 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.
- 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

Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

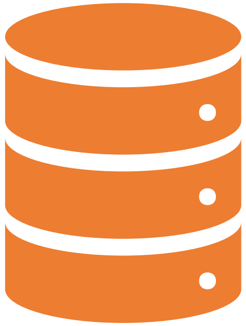
Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Integrity Rules

Integrity Rules:



Relational databases store data in tables that are related to each other.

To ensure data accuracy and consistency, databases enforce integrity rules that prevent certain types of errors or inconsistencies.

Types of Integrity Rules

- Entity Integrity: Every row (or record) in a table must have a unique identifier or primary key that distinguishes it from other rows.
- Referential Integrity: If a table has a foreign key that references another table's primary key, the foreign key must match an existing value in the referenced table, or it must be null (if allowed).
- Domain Integrity: Every column in a table must contain values that are valid according to the column's data type and constraints, such as a range of values or a set of allowed values.
- User-defined Integrity: Database designers can also define custom rules that apply to specific tables or columns, such as check constraints or triggers.

Entity Integrity

Entity integrity is the simplest type of integrity rule.

It ensures that every row in a table has a unique identifier or primary key.

Integrity Rule 1: "If attribute A of relation R(R) is a prime attribute of R(R), then A cannot accept null values."

"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

Entity Integrity Example:

CustomerID	CustomerName	CustomerAddress
1	John	123 Main St
2	Sarah	456 Elm St
3	Alex	789 Oak St

In this example, the **CustomerID** column serves as the primary key for the **Customers** table.

The entity integrity constraint ensures that each row in the table has a unique **CustomerID** value.

Referential Integrity

Referential integrity ensures that a foreign key in one table matches a primary key in another table, or is null (if allowed).

A constraint involving **two** relations.

Used to specify a **relationship** among tuples in two relations: The **referencing relation** and the **referenced relation**.

Integrity Rule 2: Given two relations R and S, suppose R refers to the relation S via a set of attributes that forms the primary key of S and this set of attributes forms a foreign key in R. Then the value of the foreign key in a tuple in R must either be equal to the primary key of a tuple of S or be entirely null.

Referential Integrity Example:

OrderID	CustomerID	OrderDate
1	2	2022-01-01
2	3	2022-02-01
3	2	2022-03-01

In this example, the **CustomerID** column in the **Orders** table references the **CustomerID** column in the **Customers** table. The referential integrity constraint ensures that each value in the **CustomerID** column in the **Orders** table matches a value in the **CustomerID** column in the **Customers** table.

Domain Integrity:

- Domain integrity ensures that each column in a table contains valid data according to its data type and constraints.
- For example, in a table of products, a column for prices might have a constraint that limits the values to a range of valid prices.
- This ensures that prices are consistent and accurate, and prevents data entry errors such as entering an invalid price or an unrealistic value.
- Domain integrity can also ensure that data is formatted consistently, such as ensuring that phone numbers are always entered in a specific format.

Domain Integrity Example:

ProductID	ProductName	ProductPrice
1	T-shirt	19.99
2	Jeans	49.99
3	Sneakers	79.99

- In this example, the **ProductPrice** column has a domain integrity constraint that limits the values to a certain range (e.g., between 0 and 100). The domain integrity constraint ensures that the **ProductPrice** values are accurate and consistent.

User Defined Integrity:

User-defined integrity rules allow database designers to define custom rules that apply to specific tables or columns.

For example, a check constraint might be defined on a column to ensure that only certain values are allowed, or a trigger might be defined to enforce a complex business rule.

User-defined integrity can allow for more specific and complex rules to be enforced, but can also add complexity and overhead to the database design and management.

User Defined Integrity Example:

UserID	Username	Password	Email
1	john123	abc123	john@example.com
2	sarah456	def456	sarah@example.com

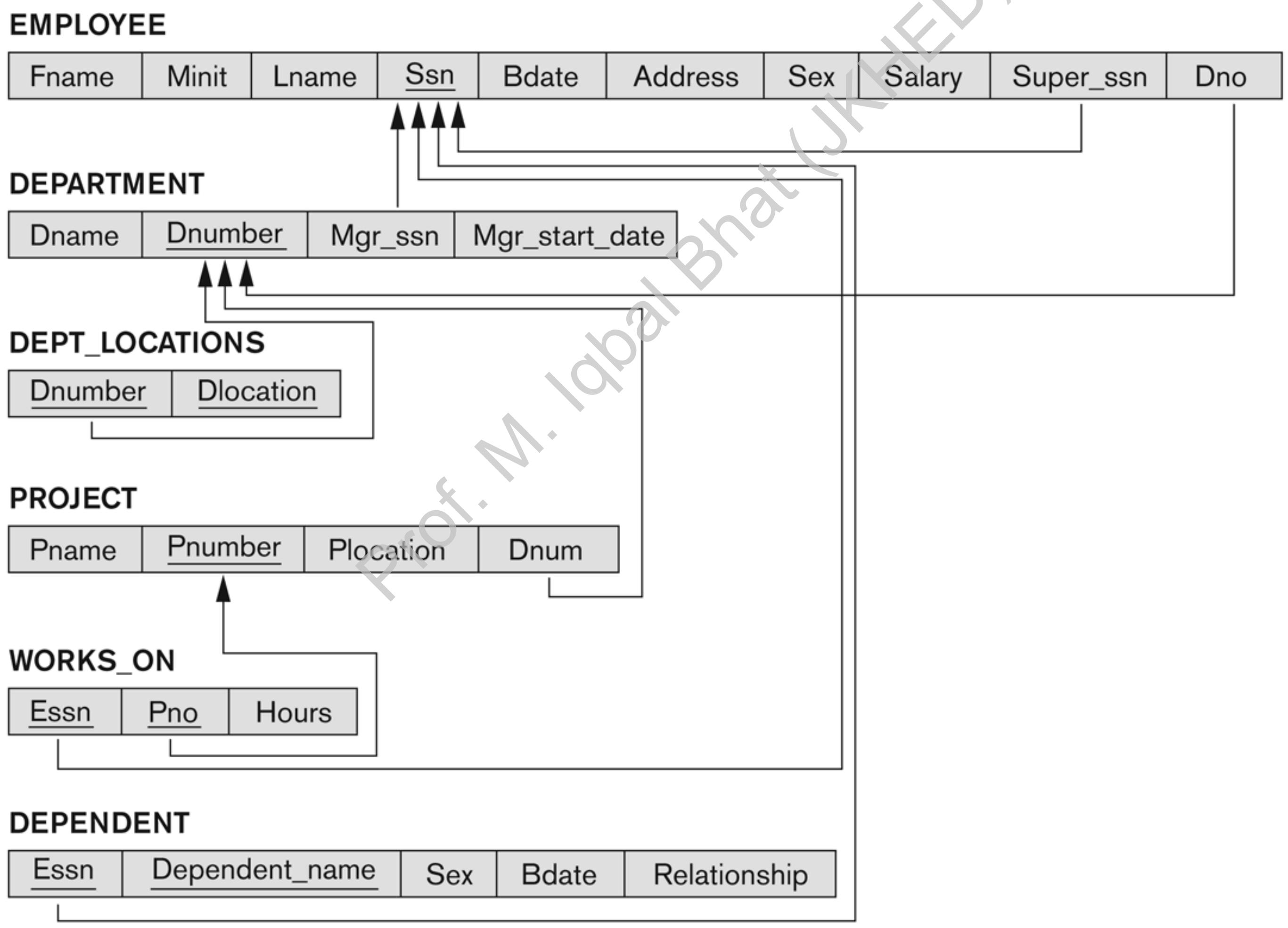
- In this example, a user-defined integrity constraint could be a check constraint that ensures that the **password** contains at least one uppercase letter, one lowercase letter, and one number. The check constraint would prevent passwords like "password" or "123456" from being entered in the Password column.

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 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
Referential integrity constraints displayed on the COMPANY relational database schema.



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: propagate the new primary key value into the foreign keys of the referencing tuples
      - 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

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
    - 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

Summary

- Presented Relational Model Concepts
  - Definitions
  - Characteristics of relations
- Discussed Relational Model Constraints and Relational Database Schemas
  - Domain constraints
  - Key constraints
  - Entity integrity
  - Referential integrity
- Described the Relational Update Operations and Dealing with Constraint Violations

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.