

# The Relational Data Model and Relational Database Constraints

- 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 8 and 9
- Note: There are several important differences between the *formal* model and the *practical* model, as we shall see

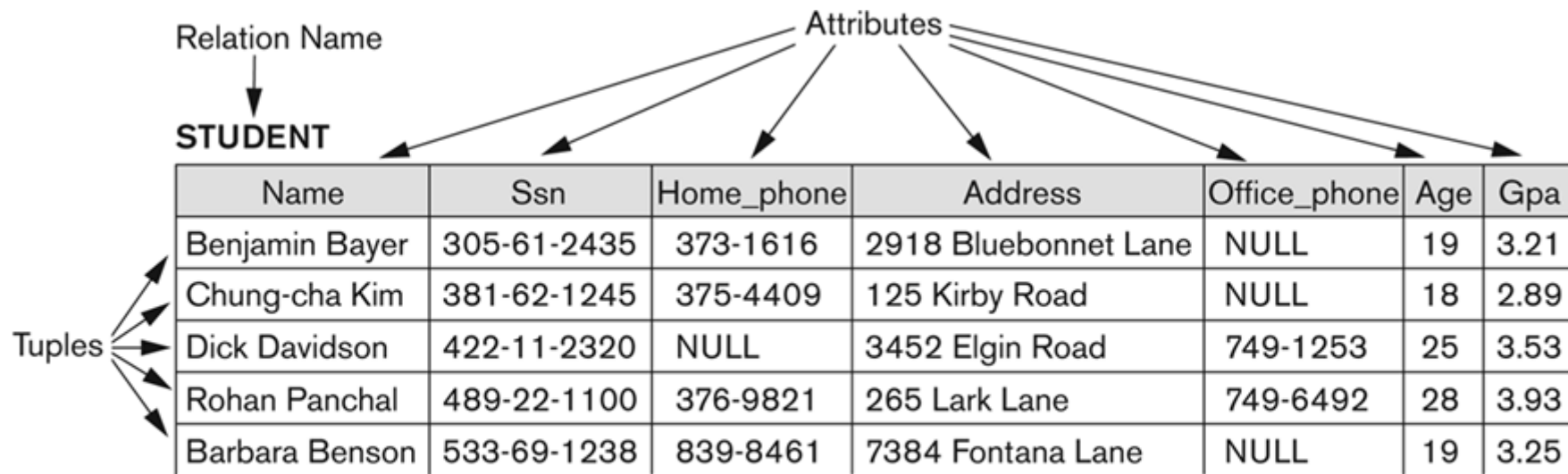
# Relational Model Concepts

- 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:
  - "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



# 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  $R(A_1, A_2, \dots, A_n)$
  - $R$  is the **name** of the relation
  - The **attributes** of the relation are  $A_1, A_2, \dots, A_n$
- 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 ' $\langle \dots \rangle$ ')
  - 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:
    - $\langle 632895, \text{"John Smith"}, \text{"101 Main St. Atlanta, GA 30332"}, \text{"(404) 894-2000"} \rangle$
    - 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
  - $\text{dom}(\text{Cust-name})$  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  $R(A_1, A_2, \dots, A_n)$
  - $r(R) \subset \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$
- $R(A_1, A_2, \dots, A_n)$  is the **schema** of the relation
- $R$  is the **name** of the relation
- $A_1, A_2, \dots, A_n$  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)
  - $r(R) = \{t_1, t_2, \dots, t_n\}$  where each  $t_i$  is an  $n$ -tuple
  - $t_i = \langle v_1, v_2, \dots, v_n \rangle$  where each  $v_j$  *element-of*  $\text{dom}(A_j)$

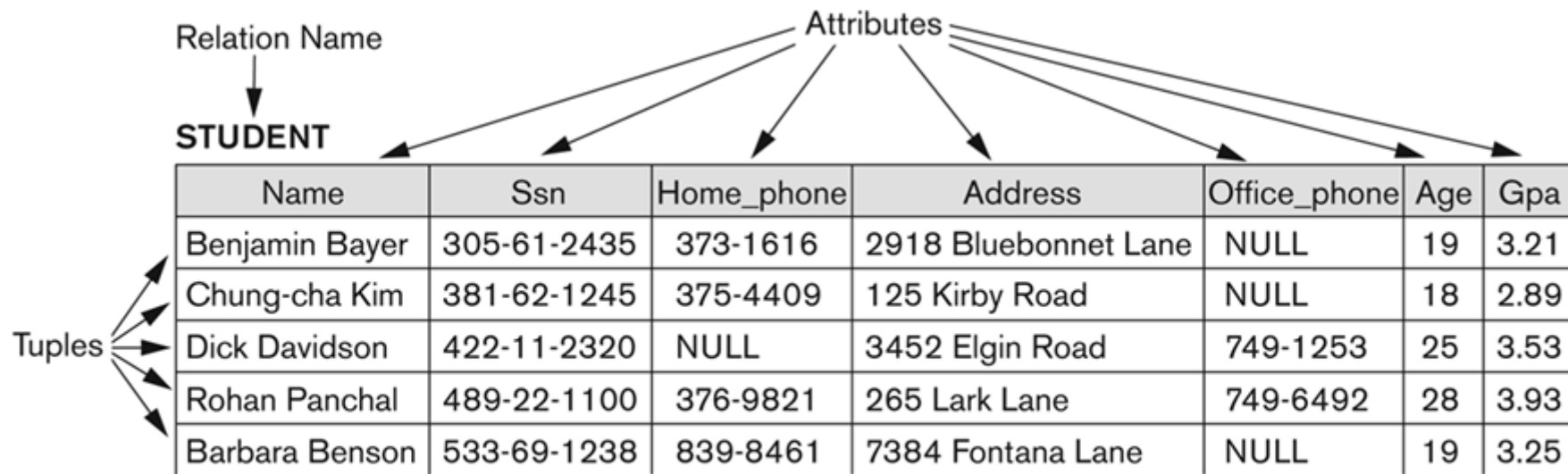
# Formal Definitions - Example

- 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:  
 $\{ \langle 0,a \rangle , \langle 0,b \rangle , \langle 0,c \rangle , \langle 1,a \rangle , \langle 1,b \rangle , \langle 1,c \rangle \}$
- The relation state  $r(R) \subset \text{dom}(A1) \times \text{dom}(A2)$
- For example:  $r(R)$  could be  $\{ \langle 0,a \rangle , \langle 0,b \rangle , \langle 1,c \rangle \}$ 
  - this is one possible state (or “population” or “extension”)  $r$  of the relation  $R$ , defined over  $A1$  and  $A2$ .
  - It has three 2-tuples:  $\langle 0,a \rangle , \langle 0,b \rangle , \langle 1,c \rangle$

# Definition Summary

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

# Example – 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  $R(A_1, A_2, \dots, A_n)$  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).

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  $t = \langle v_1, v_2, \dots, v_n \rangle$  is a tuple (row) in the relation state  $r$  of  $R(A_1, A_2, \dots, A_n)$
    - Then each  $v_i$  must be a value from  $dom(A_i)$
  - 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:
    - $t[A_i]$  or  $t.A_i$
    - This is the value  $v_i$  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 in  $t$

# Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three *main types* of constraints in the relational model:
  - **Key** constraints
  - **Entity integrity** constraints
  - **Referential integrity** constraints
- Another implicit 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  $t_1$  and  $t_2$  in  $r(R)$ ,  $t_1[SK] \neq t_2[SK]$
    - 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)

# 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

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

**CAR**

<u>License_number</u>	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**
  - $S = \{R_1, R_2, \dots, R_n\}$
  - $R_1, R_2, \dots, R_n$  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	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

## DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

## DEPT\_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

## PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

## WORKS\_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

## DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

**Figure 5.5**  
Schema diagram for  
the COMPANY  
relational database  
schema.

# 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.
    - $t[PK] \neq \text{null}$  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.

# 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**.

# 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.

# Referential Integrity (or foreign key) Constraint

- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the 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.

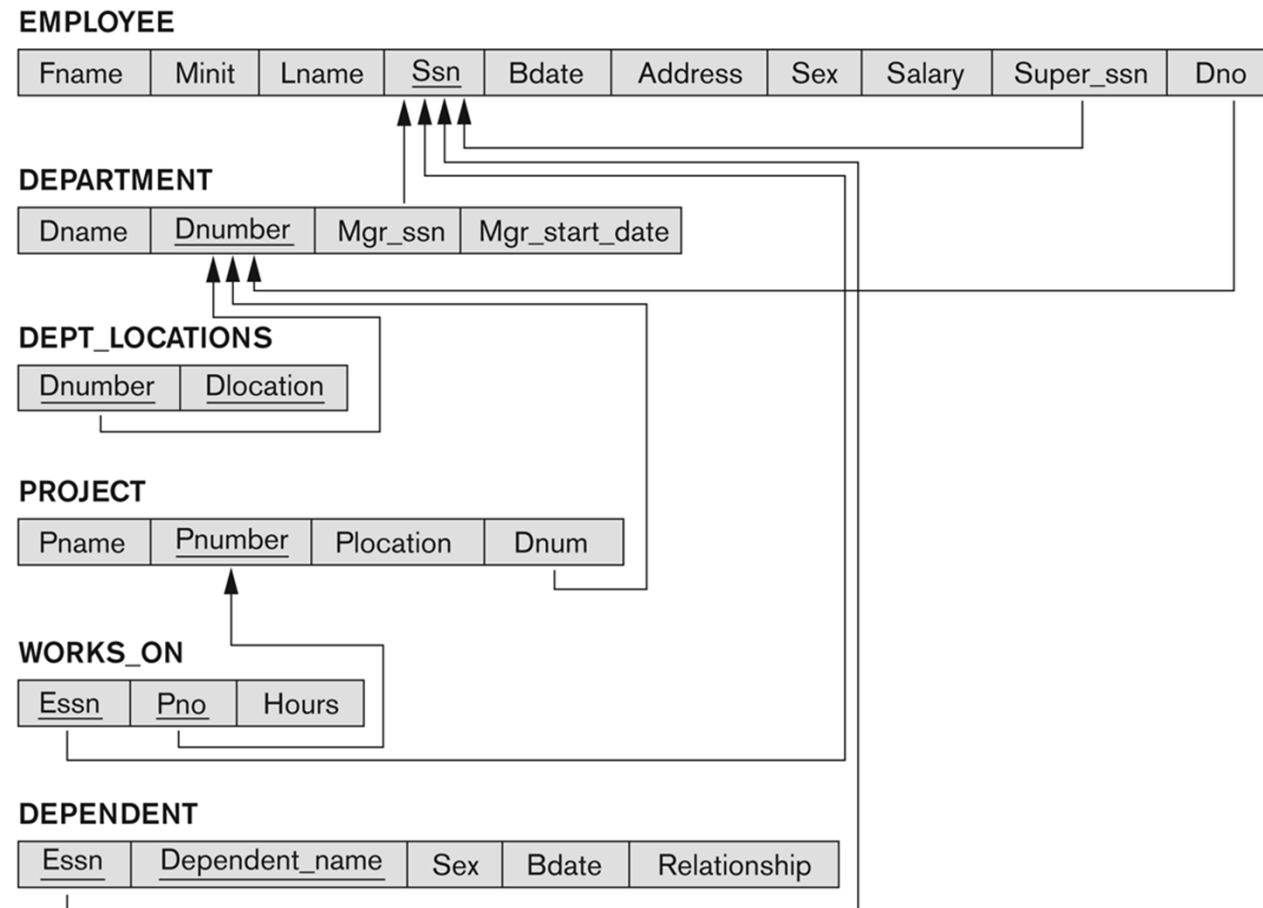
# Displaying a relational database schema and its constraints

- 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 to the primary key of the referenced relation for clarity
- Next slide shows the COMPANY **relational schema diagram**

## Referential Integrity Constraints for COMPANY database

**Figure 5.7**

Referential 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
  - 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 triggers and **ASSERTIONS** to express for some of these



# 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 shows an example state for the COMPANY database

# 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

# 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 8 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

# Alter

## **Alter Table Statement:**

Alter command is used to perform the following action on the table:

- Adding column in the existing table
- Increasing and decreasing the column size and changing data types
- Dropping column
- Renaming the column
- Adding and dropping constraints to the table( discussed in constraints topics)
- Enabling & disabling constraints in the table( discussed in constraints topics)

- 
- `alter table employee add address varchar2 (20);`
  - `alter table employee modify address varchar2 (10);`
  - `alter table employee drop column address;`
  - `alter table employee rename column mgr to manager;`



# 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

# Constraints

- **CREATE TABLE** emp\_demo2 ( employee\_id  
NUMBER(6) **PRIMARY KEY**, first\_name  
VARCHAR2(20) **NOT NULL**, last\_name  
VARCHAR2(25) **NOT NULL**, email  
VARCHAR2(25) **UNIQUE**, phone\_number  
VARCHAR2(20) **UNIQUE**, job\_id  
VARCHAR2(10), salary NUMBER(8,2)  
**CHECK(SALARY>0)**, deptid NUMBER(4)  
) ;

# Constraints with Name

```
CREATE TABLE emp_demo3  
( employee_id    NUMBER(6) CONSTRAINT emp_eid PRIMARY  
KEY,  
  first_name     VARCHAR2(20),  
  last_name      VARCHAR2(25) CONSTRAINT emp_last_name_nn  
NOT NULL,  
  email          VARCHAR2(25) CONSTRAINT emp_email_nn NOT  
NULL,  
  phone_number   VARCHAR2(20),  job_id      VARCHAR2(10)  
CONSTRAINT emp_job_nn NOT NULL,  salary  
  NUMBER(8,2) CONSTRAINT emp_salary_nn NOT NULL,  
  deptid         NUMBER(4), CONSTRAINT emp_dept FOREIGN  
KEY(deptid) REFERENCES department(deptid) ,  
CONSTRAINT emp_salary_min CHECK (salary > 0) ,  
CONSTRAINT emp_email_uk UNIQUE (email) ) ;
```



```
CREATE TABLE emp_demo4  ( emp_id  
NUMBER(6),      emp_name  VARCHAR2(15),  
      salary    NUMBER(10) CHECK (salary between  
1000 and 10000)  
      );
```

# Adding Constraints

**ALTER TABLE <tablename> ADD CONSTRAINT <constraint\_name> constraint\_type (<column name>);**

**Examples:**

**ALTER TABLE emp\_demo4 ADD CONSTRAINT con\_pk1 PRIMARY KEY(emp\_id);**

**ALTER TABLE emp\_demo4 ADD CONSTRAINT con\_emp\_uk UNIQUE(phoneno);**

**ALTER TABLE emp\_demo4 ADD CONSTRAINT con\_empfk FOREIGN KEY(DNO) REFERENCES department(dno);**

**ALTER TABLE emp\_demo4 ADD CONSTRAINT con\_emp\_ck CHECK ( salary >0 );**

**ALTER TABLE emp\_demo4 MODIFY (<Column name> <datatype> CONSTRAINT constraint\_name NOT NULL);**

# Drop Constraints

- **ALTER TABLE <tablename> DROP CONSTRAINT < constraint name**
- **ALTER TABLE employees DROP UNIQUE (email);**

# ON DELETE CASCADE

A foreign key with a cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted. This is called a cascade delete.

```
CREATE TABLE supplier  
(supplier_id number(10)not null,  
supplier_name varchar2(50)not null,  
contact_name varchar2(50),  
CONSTRAINT supplier_pk PRIMARY KEY (supplier_id));
```

```
CREATE TABLE products  
(product_id number(10)not null,  
suppl_id number(10) not null,  
CONSTRAINT fk_supplier FOREIGN KEY (suppl_id) REFERENCES  
supplier(supplier_id) ON DELETE CASCADE);
```

Because of the cascade delete, when a record with a particular supplier\_id is deleted from supplier table, then all the records of the same supplier\_id will be deleted from products table also.

## On Delete set null/On Delete Set Default

- A **foreign key** with "**set null on delete**" means that if a record in the parent table is **deleted**, then the corresponding records in the child table will have the **foreign key** fields **set to null**. The records in the child table will not be **deleted**.
- create table Users ( UserID int primary key, UserName varchar(100), ThemeID int default 1 constraint Users\_ThemeID\_FK references Themes(ThemeID) on delete set default )



# 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.**