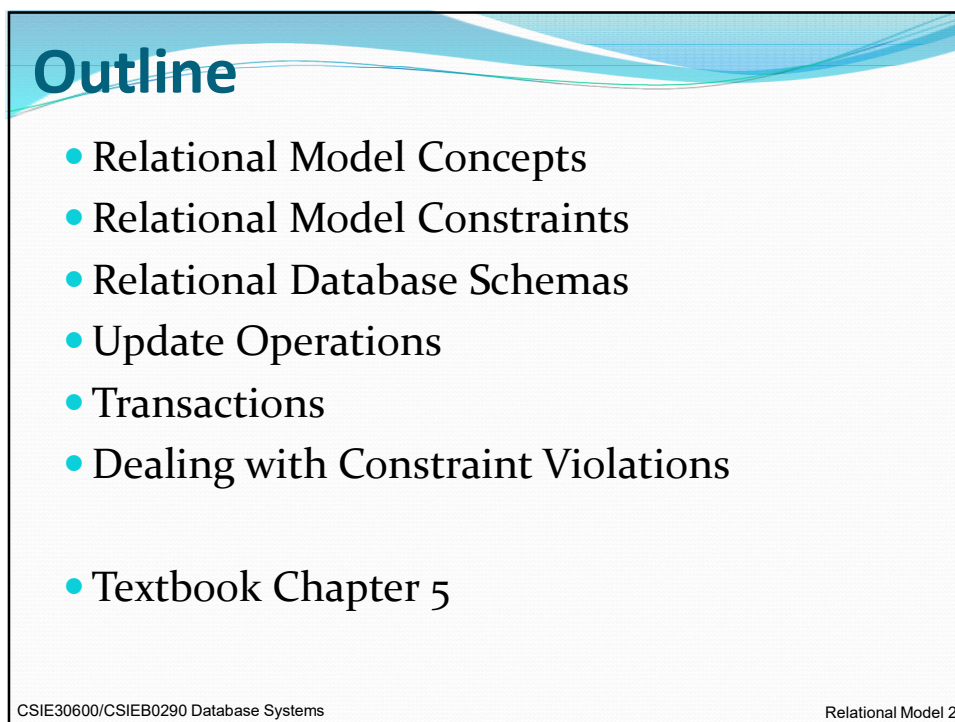



CSIE30600/CSIEB0290  
Database Systems

**Lecture 3:  
Relational Model**



## Outline

- Relational Model Concepts
- Relational Model Constraints
- Relational Database Schemas
- Update Operations
- Transactions
- Dealing with Constraint Violations
  
- Textbook Chapter 5

CSIE30600/CSIEB0290 Database Systems Relational Model 2

## Data Model Revisited

- Provides the means for **specifying** particular **data structures**, for **constraining** the data sets associated with these structures, and for **manipulating** the data
- **Data definition language (DDL)**: define structures and constraints
- **Data manipulation language (DML)**: specify manipulations/operations over the data

## Why Study the Relational Model

- Extremely useful and simple
  - Single data-modeling concept: **relations** = 2-D **tables**
  - Allows **clean** yet **powerful** manipulation languages
- Most widely used model
  - Vendors: Oracle, IBM(Informix), Microsoft, SAP(Sybase), etc.
- Recent competitors: **object-oriented model**, **semi-structured model (XML)**, **multi-model**
  - ObjectStore, Versant, Tamino
  - A synthesis emerging:
    - *object-relational model*: PostgreSQL, IBM Informix, Oracle, DB2
    - XML-enabled databases: Oracle, DB2, SQLServer

## OO DBMS Ranking

17 systems in ranking, October 2017

Rank			DBMS	Database Model	Score		
Oct 2017	Sep 2017	Oct 2016			Oct 2017	Sep 2017	Oct 2016
1.	1.	1.	Caché	Multi-model	2.90	+0.16	+0.82
2.	2.	2.	Db4o	Object oriented DBMS	1.39	-0.03	-0.20
3.	4.	3.	Versant Object Database	Object oriented DBMS	1.19	+0.08	-0.11
4.	3.	4.	ObjectStore	Object oriented DBMS	1.11	-0.02	+0.13
5.	5.	5.	Matisse	Object oriented DBMS	0.72	+0.05	-0.08
6.	6.	6.	Objectivity/DB	Object oriented DBMS	0.60	-0.02	+0.16
7.	8.	8.	ObjectDB	Object oriented DBMS	0.52	+0.01	+0.18
8.	9.	9.	GemStone/S	Object oriented DBMS	0.49	-0.01	+0.17
9.	7.	7.	Perst	Object oriented DBMS	0.48	-0.04	+0.10
10.	10.	13.	Jade	Object oriented DBMS	0.41	+0.00	+0.32
11.	11.	10.	Starcounter	Object oriented DBMS	0.30	-0.00	+0.09
12.	13.	12.	Siaqodb	Object oriented DBMS	0.10	+0.02	-0.00
13.	14.	15.	Eloquera	Object oriented DBMS	0.10	+0.02	+0.02
14.	15.	16.	Versant FastObjects	Object oriented DBMS	0.10	+0.03	+0.02
15.	12.	11.	VelocityDB	Multi-model	0.09	-0.02	-0.05
16.	16.	14.	WakandaDB	Object oriented DBMS	0.07	+0.02	-0.02
17.	17.	17.	OrigoDB	Multi-model	0.00	±0.00	±0.00

CSIE30600/CSIEB0290 Database Systems Relational Model 5

## Native XML DBMS Ranking

8 systems in ranking, October 2017

Rank			DBMS	Database Model	Score		
Oct 2017	Sep 2017	Oct 2016			Oct 2017	Sep 2017	Oct 2016
1.	1.	1.	MarkLogic	Multi-model	11.81	-0.29	+1.51
2.	2.	2.	Oracle Berkeley DB	Multi-model	2.70	-0.15	-0.06
3.	3.	3.	Virtuoso	Multi-model	1.86	-0.03	-0.83
4.	4.	6.	BaseX	Native XML DBMS	1.29	+0.13	+0.37
5.	5.	5.	webMethods Tamino	Native XML DBMS	0.85	+0.02	-0.18
6.	6.	4.	Sedna	Native XML DBMS	0.73	-0.06	-0.49
7.	7.	7.	eXist-db	Native XML DBMS	0.72	+0.01	+0.52
8.	8.	8.	searchxml	Multi-model	0.00	-0.01	-0.01

CSIE30600/CSIEB0290 Database Systems Relational Model 6

## 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 – to be described in later lectures
- Note: There are several important differences between the *formal* model and the *practical* model, as we shall see

CSIE30600/CSIEB0290 Database Systems

Relational Model 7

## 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.
  - use relations as data structures, algebra for specifying queries, no mechanisms for updates or constraints
  - follow-up papers introduced new language based on first-order logic and showed it is equivalent to the algebra, introduced integrity constraints
- These papers caused a major revolution in the field of database management and earned Dr. Codd the coveted ACM Turing Award

CSIE30600/CSIEB0290 Database Systems

Relational Model 8

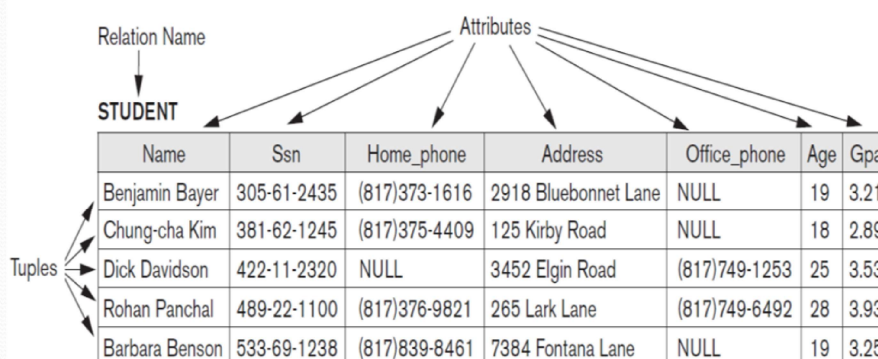
## Informal Definitions

- Informally, a **relation** looks like a **table** of values.
- A relation 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 in that column
  - In the formal model, the column header is called an **attribute name** (or just **attribute**)

CSIE30600/CSIEB0290 Database Systems

Relational Model 9

## Example of a Relation



**Figure 5.1**

The attributes and tuples of a relation STUDENT.

CSIE30600/CSIEB0290 Database Systems

Relational Model 10

## 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**

CSIE30600/CSIEB0290 Database Systems

Relational Model 11

## 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  $A_1, A_2, \dots, A_n$
  - The **degree** (or **arity**) of a relation: no. of attributes (n)
- Example:
 

CUSTOMER (Cust-id, Cust-name, Address, Phone#)

  - CUSTOMER is the relation name
  - Defined over the 4 attributes: Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** (a set of valid values)
  - For example, the domain of Cust-id is 6 digit numbers.
  - Denoted as  $dom(A_1), dom(A_2), \dots$

CSIE30600/CSIEB0290 Database Systems

Relational Model 12

## Formal Definitions - Tuple

- An **n-tuple** is an ordered list of  $n$  values (enclosed in angled brackets ' $\langle v_1, v_2, \dots, v_n \rangle$ ')
- Each value  $v_i$ ,  $1 \leq i \leq n$ , is an element of  $\text{dom}(A_i)$  or is a special **NULL** value
- A row in the CUSTOMER relation is a 4-tuple consisting of 4 values, for example:
  - $\langle 632895, \text{"John Smith"}, \text{"101 Main St. Atlanta, GA 30332"}, \text{"(404) 894-2000"} \rangle$
- A **relation**  $r = \{t_1, t_2, \dots, t_m\}$  is a **set** of  $n$ -tuples (rows)

CSIE30600/CSIEB0290 Database Systems

Relational Model 13

## 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.

CSIE30600/CSIEB0290 Database Systems

Relational Model 14

## Attribute Name and 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

## Attribute Name and Domain

- More 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*.



## Relation 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.
- Cartesian product of the domains is the set of all possible combinations of attribute values

## Relation State - Examples

- Let  $R(A_1, A_2)$  be a relation schema:
  - Let  $\text{dom}(A_1) = \{0,1\}$
  - Let  $\text{dom}(A_2) = \{a,b,c\}$
- Cartesian product of the domains:  $\text{dom}(A_1) \times \text{dom}(A_2)$  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  $r(R) \subset \text{dom}(A_1) \times \text{dom}(A_2)$
- Eg.:  $r(R)$  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  $A_1$  and  $A_2$ .
  - It has three 2-tuples:  $\langle 0,a \rangle, \langle 0,b \rangle, \langle 1,c \rangle$

## 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)
  - $r(R) = \{t_1, t_2, \dots, t_m\}$  where each  $t_i$  is an  $n$ -tuple
  - $t_i = \langle v_1, v_2, \dots, v_n \rangle$  where each  $v_j$  is an *element-of*  $\text{dom}(A_j)$

## 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

## 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**.

CSIE30600/CSIEB0290 Database Systems

Relational Model 21

## Alternative Definition

- Tuple considered as a **set** of ( $\langle$ attribute $\rangle$ ,  $\langle$ value $\rangle$ ) pairs
- Each pair gives the value of the mapping from an **attribute  $A_i$**  to a value  $v_i$  from  **$\text{dom}(A_i)$**
- Use the first definition of relation
  - Attributes and the values within tuples are ordered
  - Simpler notation

CSIE30600/CSIEB0290 Database Systems

Relational Model 22

## STUDENT Relation Revisited

		Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Tuples	Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21	
	Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89	
	Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53	
	Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93	
	Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25	

**Figure 5.1**  
The attributes and tuples of a relation STUDENT.

## 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	(817)749-1253	25	3.53	
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25	
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93	
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89	
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21	

## Characteristics of Relations

- Values in tuples:
  - 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  $dom(A_i)$
  - **Flat relational model**
    - Composite and multivalued attributes not allowed
    - **First normal form** assumption

CSIE30600/CSIEB0290 Database Systems

Relational Model 25

## Characteristics of Relations

- NULL values
  - A special **NULL** value is used to represent values that are unknown or inapplicable.
- Meanings for NULL values
  - Value **unknown**
  - Value exists but is **not available**
  - Attribute **does not apply** to this tuple (also known as value **undefined**)

CSIE30600/CSIEB0290 Database Systems

Relational Model 26

## Characteristics of Relations

- Interpretation (meaning) of a relation
  - **Assertion**
    - Each tuple in the relation is a **fact** or a particular instance of the assertion
  - **Predicate**
    - Values in each tuple interpreted as values that satisfy predicate

CSIE30600/CSIEB0290 Database Systems

Relational Model 27

## Relational Model Notation

- Relation schema  $R$  of degree  $n$ 
  - Denoted by  $R(A_1, A_2, \dots, A_n)$
- Uppercase letters  $Q, R, S$ 
  - Denote relation names
- Lowercase letters  $q, r, s$ 
  - Denote relation states
- Letters  $t, u, v$ 
  - Denote tuples

CSIE30600/CSIEB0290 Database Systems

Relational Model 28

## Relational Model 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,  $t[A_u, A_v, \dots, A_w]$  and  $t(A_u, A_v, \dots, A_w)$  refers to the subtuple of  $t$  containing the values of attributes  $A_u, A_v, \dots, A_w$ , respectively in  $t$

CSIE30600/CSIEB0290 Database Systems

Relational Model 29

## Relational Model Constraints

- **Constraints**
  - Restrictions on the actual values in a database state
  - Derived from the rules in the miniworld that the database represents
  - Must hold on **all** valid relation states.
  - Three categories (below)
- **Inherent model-based constraints** or **implicit constraints**
  - **Inherent** in the data model

CSIE30600/CSIEB0290 Database Systems

Relational Model 30

## Relational Model Constraints

- **Schema-based constraints** or **explicit constraints**
  - Can be **directly expressed** in schemas of the data model
- **Application-based** or **semantic constraints** or **business rules**
  - Cannot be directly expressed in schemas
  - Expressed and enforced by application program

CSIE30600/CSIEB0290 Database Systems

Relational Model 31

## Domain Constraints

- An implicit constraint is the **domain** constraint
  - Every value in a tuple must be from the *domain of its attribute* (or **null**, if allowed for that attribute)
- Typically include:
  - Numeric data types for integers and real numbers
  - Characters
  - Booleans
  - Fixed-length strings
  - Variable-length strings
  - Date, time, timestamp
  - Money
  - Other special data types

CSIE30600/CSIEB0290 Database Systems

Relational Model 32



## Key Constraints

- No two tuples can have the same combination of values for all their attributes. (no duplicate tuples)
- **Superkey** of R is a **set of attributes** SK of R such that:
  - No two tuples in any valid 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 (candidate 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)

CSIE30600/CSIEB0290 Database Systems

Relational Model 33

## Key Constraints (continued)

- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - CAR has two keys:
    - Key<sub>1</sub> = {State, Reg#}
    - Key<sub>2</sub> = {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*

CSIE30600/CSIEB0290 Database Systems

Relational Model 34

## Key Constraints (continued)

- A relation can have several **candidate keys**
- **Primary key** of the relation
  - Designated among candidate keys
  - Underline attribute
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- Other candidate keys are designated as **unique keys**

CSIE30600/CSIEB0290 Database Systems

Relational Model 35

## Key Constraints (continued)

- 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

CSIE30600/CSIEB0290 Database Systems

Relational Model 36

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

**Figure 5.4**  
The CAR relation, with two candidate keys:  
License\_number and  
Engine\_serial\_number.

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

CSIE30600/CSIEB0290 Database Systems

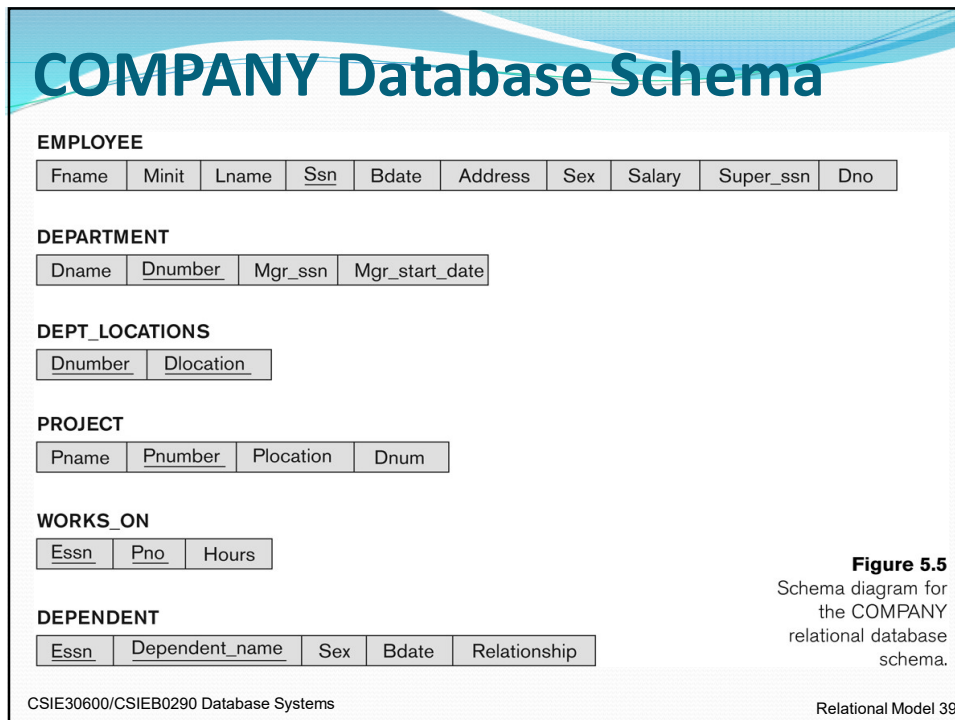
Relational Model 37

## Database Schema & State

- **Relational Database Schema:**
  - A set  $S = \{R_1, R_2, \dots, R_n\}$  of **relation schemas** that belong to the same database.
  - Set of **integrity constraints IC**
- Following slide shows a COMPANY database schema with 6 relation schemas
- **Relational database state**
  - Set of relation states  $DB = \{r_1, r_2, \dots, r_m\}$
  - Each  $r_i$  is a state of  $R_i$  and such that the  $r_i$  relation states satisfy integrity constraints specified in IC

CSIE30600/CSIEB0290 Database Systems

Relational Model 38



## Relational Database State

- **Valid state**
  - Satisfies all the constraints in the defined set of integrity constraints IC
- **Invalid state**
  - Does not obey all the integrity constraints

CSIE30600/CSIEB0290 Database Systems Relational Model 40

## 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

CSIE30600/CSIEB0290 Database Systems

Relational Model 41

## Populated Database State for COMPANY

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

Empno	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

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

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

Empno	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

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

Empno	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

CSIE30600/CSIEB0290 Database Systems

Relational Model 42

## Entity Integrity

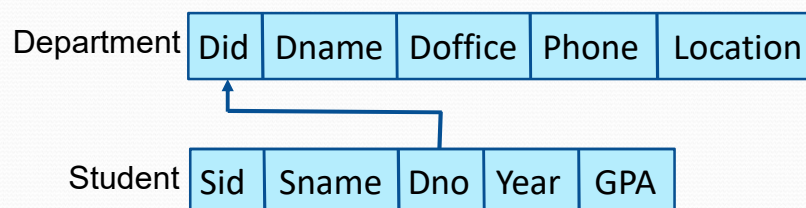
- The **primary key attributes** PK of each relation schema R in S **cannot have null values** in any tuple of r(R).
  - This is because primary key values are used to *identify* the individual tuples.
  - **t[PK] ≠ null** for any tuple t in r(R)
  - If PK has several attributes, null is not allowed 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.

CSIE30600/CSIEB0290 Database Systems

Relational Model 43

## 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**.
  - Maintains **consistency** among tuples in two relations



CSIE30600/CSIEB0290 Database Systems

Relational Model 44

## Referential Integrity

- 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**  $R_2$ .
  - Value of FK in a tuple  $t_1$  of the current state  $r_1(R_1)$  either occurs as a value of PK for some tuple  $t_2$  in the current state  $r_2(R_2)$  or is NULL
  - $t_1$  is said to **reference**  $t_2$  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  $R_1.\text{FK}$  to  $R_2$ .

CSIE30600/CSIEB0290 Database Systems

Relational Model 45

## Referential Integrity (or foreign key) Constraint

- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the **referencing relation**  $R_1$  can be **either**:
    - (1) a value of an **existing primary key value** of a 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.

CSIE30600/CSIEB0290 Database Systems

Relational Model 46

## Displaying a 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

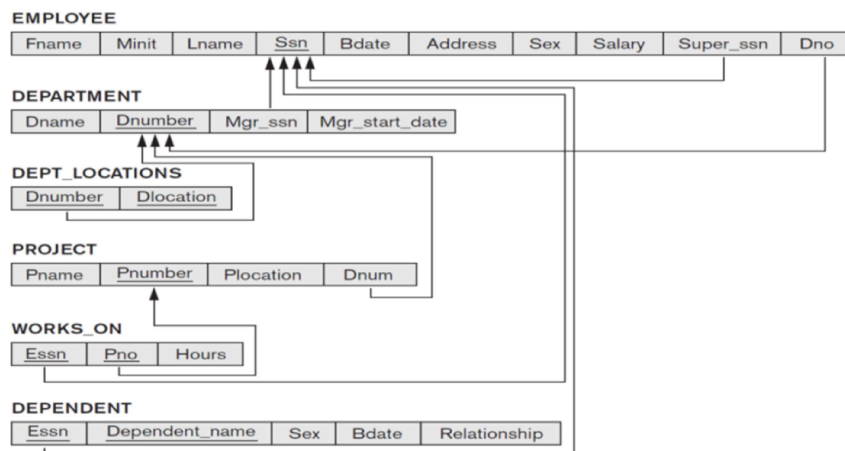
CSIE30600/CSIEB0290 Database Systems

Relational Model 47

## Relational Schema Diagram for COMPANY database

**Figure 5.7**

Referential integrity constraints displayed on the COMPANY relational database schema.



CSIE30600/CSIEB0290 Database Systems

Relational Model 48



## 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 is 56 hrs per week”
- A **constraint specification** language may have to be used to express these
- SQL allows **triggers** and **assertions** to express for some of these
  - More common to check for these types of constraints within the **application programs**

CSIE30600/CSIEB0290 Database Systems

Relational Model 49

## Other Types of Constraints

- **Functional dependency** constraint
  - Establishes a **functional relationship** among two sets of attributes  $X$  and  $Y$
  - Value of  $X$  determines a unique value of  $Y$
- **State constraints**
  - Define the constraints that a valid state of the database must satisfy
- **Transition constraints**
  - Define to deal with state changes in the database

CSIE30600/CSIEB0290 Database Systems

Relational Model 50

## Specification of a Relational Schema

- Select the relations, with a **name for each table**
- Select **attributes** for each relation and give the **domain** for each attribute
- Specify the **key(s)** for each relation
- Specify all appropriate **foreign keys** and **integrity constraints**
- **Database schema** is the set of schemas for the relations in a design

CSIE30600/CSIEB0290 Database Systems

Relational Model 51

## Update Operations on Relations

- **Operations** of the relational model can be categorized into **retrievals** and **updates**
- Basic operations that change the states of relations in the database:
  - **Insert, Delete, Update** (or **Modify**)
- 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.

CSIE30600/CSIEB0290 Database Systems

Relational Model 52

## 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**

CSIE30600/CSIEB0290 Database Systems

Relational Model 53

## The Insert Operation

- Provides a list of attribute values for a new tuple  $t$  that is to be inserted into a relation  $R$
- **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

CSIE30600/CSIEB0290 Database Systems

Relational Model 54

## The Delete Operation

- **DELETE** may violate only referential integrity:
  - If the primary key value of the tuple being deleted is referenced from other tuples
    - Can be remedied by several actions: **RESTRICT**, **CASCADE**, **SET NULL** or **SET DEFAULT** (will discuss)
      - **RESTRICT** option: reject the deletion
      - **CASCADE** option: propagate the new primary key value into the foreign keys of the referencing tuples
      - **SET** option: set the foreign keys of the referencing tuples to NULL or default value
  - One of the above options must be specified during database design for each foreign key constraint

CSIE30600/CSIEB0290 Database Systems

Relational Model 55

## The Update Operation

- Necessary to specify a condition on attributes of relation
  - Select the tuple (or tuples) to be modified
- **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

CSIE30600/CSIEB0290 Database Systems

Relational Model 56

## The Transaction Concept

- **Transaction**
  - Executing a designated function
  - Includes a sequence of operations
  - Considered as a single composite operation
  - Must leave the database in a valid or consistent state
- **Online transaction processing (OLTP)** systems
  - Execute transactions at rates that reach several hundred per second

## Summary

- Relational **model** concepts
  - Definitions (informal and formal)
  - Characteristics of relations
- Relational model **constraints** and relational database **schemas**
  - Inherent model-based constraints, explicit schema-based constraints, and application-based constraints
  - Domain constraints, Key constraints, Entity integrity, Referential integrity
- Relational **update operations** and dealing with **constraint violations**

## In-Class Exercise

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.**