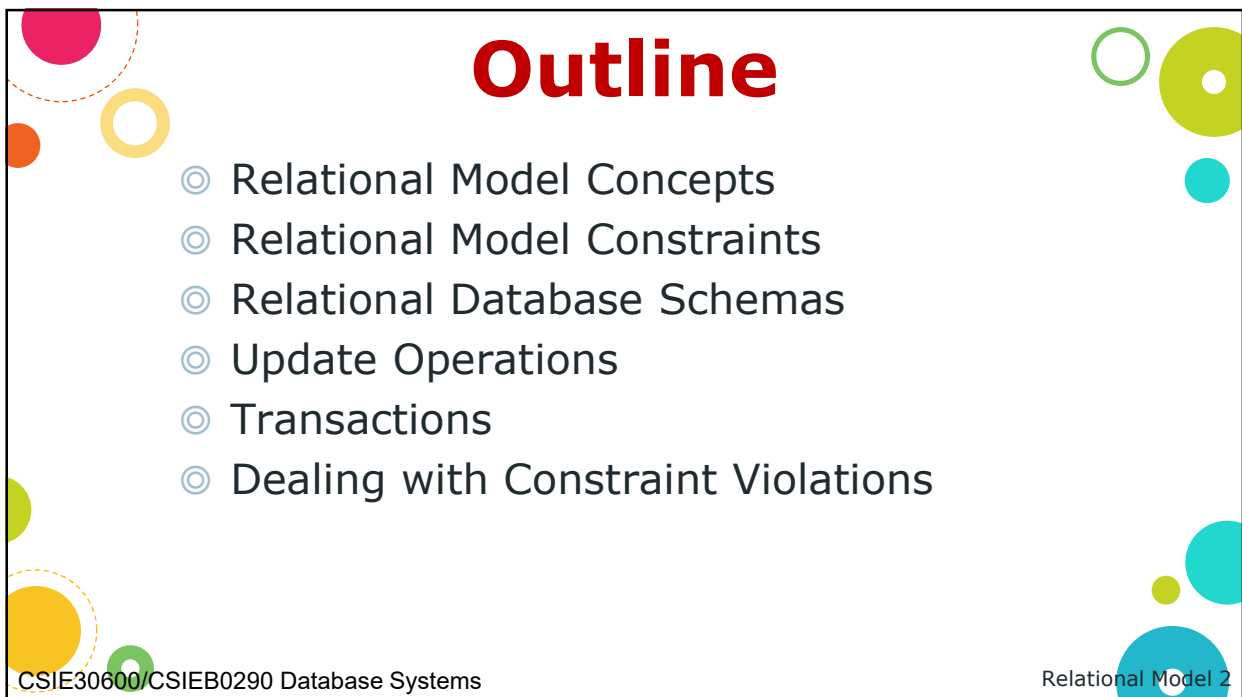


CSIE30600/CSIEB0290
Database Systems
Lecture 3:
Relational Model

The slide features a decorative background with various colored circles (teal, green, orange, yellow) and a dashed line. In the bottom right corner, there is an icon of a database consisting of three blue cylinders.



Outline

- ⦿ Relational Model Concepts
- ⦿ Relational Model Constraints
- ⦿ Relational Database Schemas
- ⦿ Update Operations
- ⦿ Transactions
- ⦿ Dealing with Constraint Violations

CSIE30600/CSIEB0290 Database Systems Relational Model 2

The slide features a decorative background with various colored circles (pink, orange, yellow, green, teal) and a dashed line.

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 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(DB2, Informix), Microsoft(SQL Server, Access), etc.
- ⊙ Recent competitors: **object-relational model, semi-structured model, document, key-value, NoSQL, NewSQL, Distributed SQL**
 - ⊙ MongoDB(document), Redis(key-value), Cassandra(NoSQL), HBase(NoSQL), Spanner(Distributed SQL)
 - ⊙ Object-oriented aspects of SQL:1999

DB Engine Ranking

417 systems in ranking, February 2024

Rank			DBMS	Database Model	Score		
Feb 2024	Jan 2024	Feb 2023			Feb 2024	Jan 2024	Feb 2023
1.	1.	1.	Oracle	Relational, Multi-model	1241.45	-6.05	-6.08
2.	2.	2.	MySQL	Relational, Multi-model	1106.67	-16.79	-88.78
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	853.57	-23.03	-75.52
4.	4.	4.	PostgreSQL	Relational, Multi-model	629.41	-19.55	+12.90
5.	5.	5.	MongoDB	Document, Multi-model	420.36	+2.88	-32.41
6.	6.	6.	Redis	Key-value, Multi-model	160.71	+1.33	-13.12
7.	7.	8.	Elasticsearch	Search engine, Multi-model	135.74	-0.33	-2.86
8.	8.	7.	IBM Db2	Relational, Multi-model	132.23	-0.18	-10.74
9.	9.	12.	Snowflake	Relational	127.45	+1.53	+11.80
10.	11.	9.	SQLite	Relational	117.28	+2.08	-15.38
11.	10.	10.	Microsoft Access	Relational	113.17	-4.50	-17.86
12.	12.	11.	Cassandra	Wide column, Multi-model	109.27	-1.77	-6.95
13.	13.	13.	MariaDB	Relational, Multi-model	97.23	-2.00	+0.42
14.	14.	14.	Splunk	Search engine	91.65	-1.07	+4.57
15.	16.	15.	Amazon DynamoDB	Multi-model	82.90	+1.96	+3.21
16.	15.	16.	Microsoft Azure SQL Database	Relational, Multi-model	79.56	-1.51	+0.81
17.	17.	19.	Databricks	Multi-model	76.91	-3.62	+16.58
18.	18.	17.	Hive	Relational	65.81	-1.15	-6.31
19.	19.	22.	Google BigQuery	Relational	63.63	+0.15	+11.17
20.	20.	18.	Teradata	Relational, Multi-model	51.24	-1.94	-11.79

CSIE30600/CSIEB0290 Database Systems Relational Model 5

- # Relational Model Concepts
- The relational model of data is based on the concept of a *relation*.
 - The strength of the relational approach 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 model used by 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 6

Relational Model Concepts

- ⦿ A **relation** is a mathematical concept based on sets.
- ⦿ The model was first proposed by Dr. E. F. Codd of IBM Research in 1970 in the following paper: (next slide)
 - ⦿ "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.

Dawn of the Relational Model

Information Retrieval

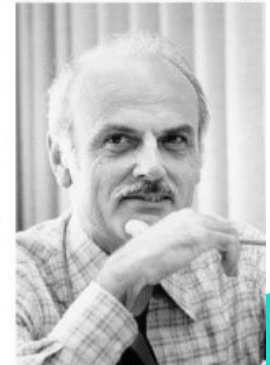
A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network

The relational view (or model Section 1 appears to be superior in graph or network model [3, 4] pre



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 indicates the **meaning** of the data in that column
 - In the formal model, the column header is called an **attribute name** (or just **attribute**)

Example of a *INSTRUCTOR* Relation

INSTRUCTOR

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

attributes
(or columns)

tuples
(or rows)

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 INSTRUCTOR table, ID is the key
- ⊙ Sometimes row-ids or sequential numbers are assigned as keys to identify the rows
 - Called **artificial key** or **surrogate key**

Formal Defs - 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)
 - ⊙ A relation instance r defined over schema R is denoted by $r(R)$.
- ⊙ Example: **CUSTOMER(Cid, Cname, Address, Phone#)**
 - ⊙ CUSTOMER is the relation name
 - ⊙ Defined over the 4 attributes: Cid, Cname, Address, Phone#
- ⊙ Each attribute has a **domain** (a set of valid values)
 - ⊙ For example, the domain of Cid is 6 digit numbers.
 - ⊙ Denoted as $dom(A_1), dom(A_2), \dots$

Formal Defs - 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 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 formats such as year, month, date formatted as **yyyy-mm-dd**, or as **dd mm,yyyy** etc.

Formal Defs - Tuple

- ⊙ An **n-tuple (row)** 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 (discussed later)
 - ⊙ Attribute values are **atomic**; that is, indivisible.
 - ⊙ A row in the CUSTOMER relation is a 4-tuple with 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.

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 “**Invoice-date**” and “**Payment-date**”
- ⦿ More example: attribute **Cname** is defined over the domain of char strings of max length 25
 - ⦿ $\text{dom}(\text{Cname})$ 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. (example in next slide)

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 Defs - 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 $v_j \in \text{dom}(A_j)$

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** 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):
 - ⊙ Attributes in $R(A_1, A_2, \dots, A_n)$ and the values in $t = \langle v_1, v_2, \dots, v_n \rangle$ are **ordered**.

Alternative Definition

- ⊙ Tuple considered as a set of (<attribute>, <value>) pairs
- ⊙ Each pair gives the value of the mapping from an attribute A_i to a value v_i from $dom(A_i)$
- ⊙ Use the first definition of relation
 - ⊙ Attributes and the values within tuples are ordered
 - ⊙ Simpler notation

More Example: STUDENT Relation

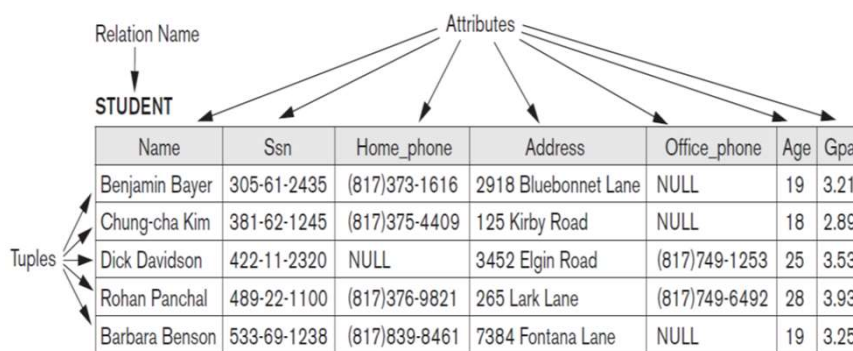


Figure 5.1
The attributes and tuples of a relation STUDENT.

Figure (with diff ordering 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

NULL

- **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**)
- The NULL value causes complications in the definition of many operations.

Interpretation 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

Model Notations

- ⊙ 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

Model Notations

- ⊙ 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 .

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

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

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

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 (can **uniquely identify** tuples)
 - ⊙ 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
 - ⊙ 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 (cont.)

- ⊙ 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. (why?)
- ⊙ 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

Examples of Keys

- ⊙ **Example:** Consider the STUDENTS schema
 STUDENTS(SSN, StuID, Name, Major, Bdate)
- ⊙ What are the candidate keys of STUDENTS?
- ⊙ What is the primary key?
- ⊙ Is (Name, Major) a superkey?
- ⊙ What about (SSN, Name)?
- ⊙ Is (SSN, Name) a candidate key? Why(not)?

Key Constraints (cont.)

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

Key Constraints (cont.)

- ⊙ 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 relation
 - ⊙ 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

Figure 5.4

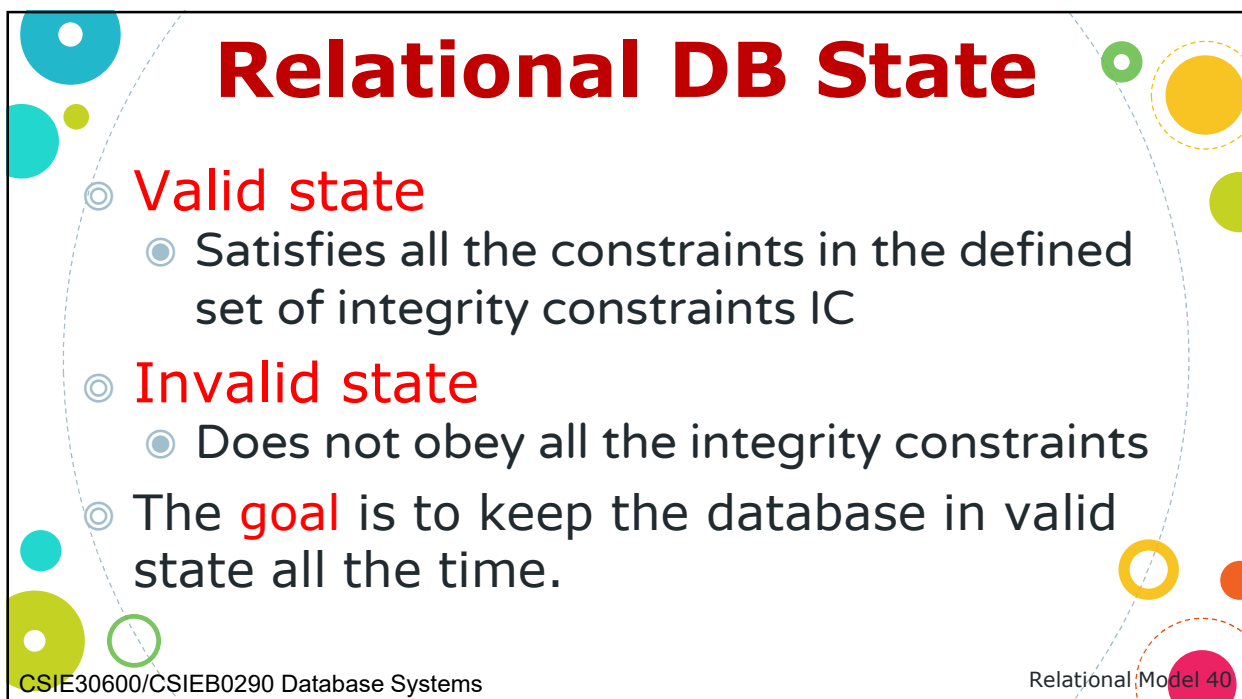
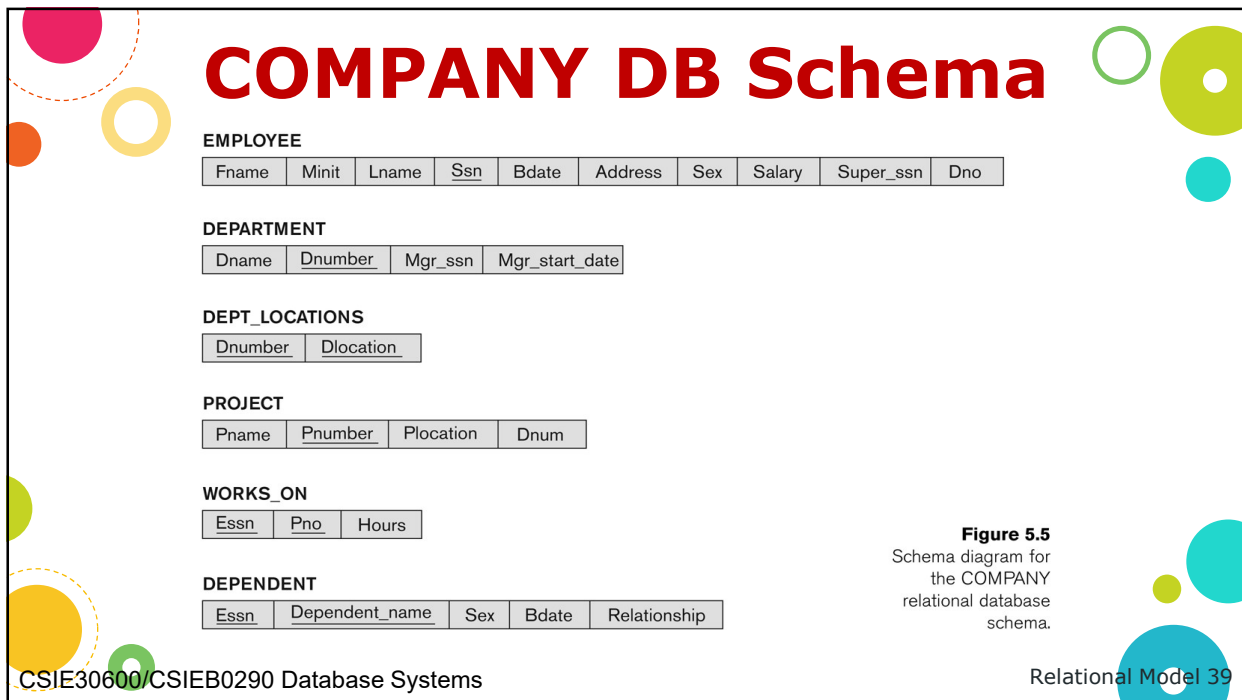
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

CAR

<u>License_number</u>	<u>Engine_serial_number</u>	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

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.
 - ⊙ A 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



Populated DB State

- ⦿ Each **relation** has 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 DB State for COMPANY

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

Empname	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
Ahmed	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

Empname	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

Empname	Pro	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

Esan	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

Entity Integrity

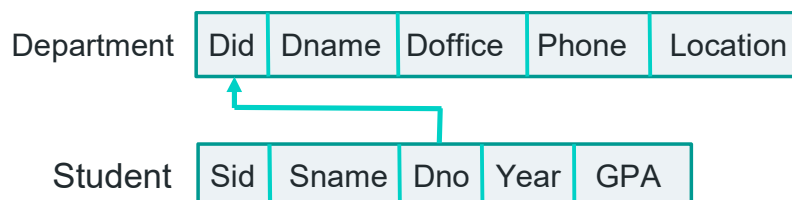
- ⦿ The **primary key attributes** PK of each relation schema R in S **cannot have null values** in any tuple of $r(R)$.
 - ⦿ This is because primary key values are used to *identify* the individual tuples.
 - ⦿ $t[PK] \neq \text{null}$ for any tuple t in $r(R)$
 - ⦿ If PK has several attributes, null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

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** R1 have attributes FK (called **foreign key**) that reference the primary key attributes PK of the **referenced relation** R2.
 - ⦿ Value of FK in a tuple t1 of the current state r1(R1) either occurs as a value of PK for some tuple t2 in the current state r2(R2) or is NULL.
 - ⦿ t1 is said to **reference** t2 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

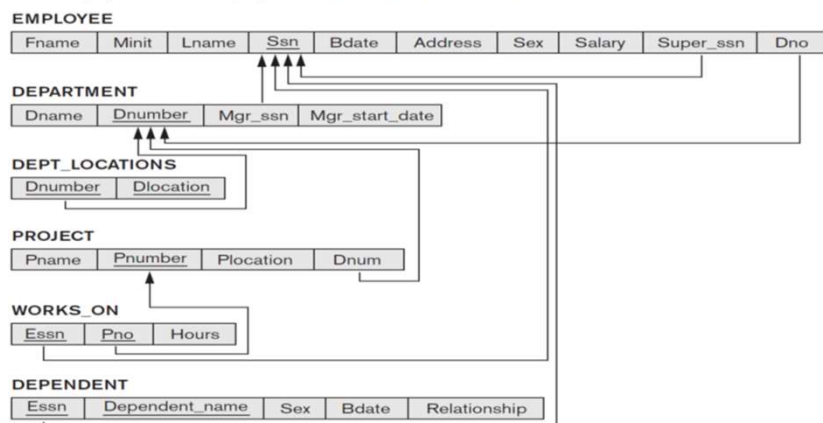
- ⦿ Statement of the constraint
 - ⦿ The value in the foreign key column (or columns) FK of the referencing relation R1 can be either:
 - 1) a value of an **existing primary key value** of a corresponding primary key PK in the referenced relation R2, or
 - 2) a **null**.
- ⦿ In case (2), the FK in R1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

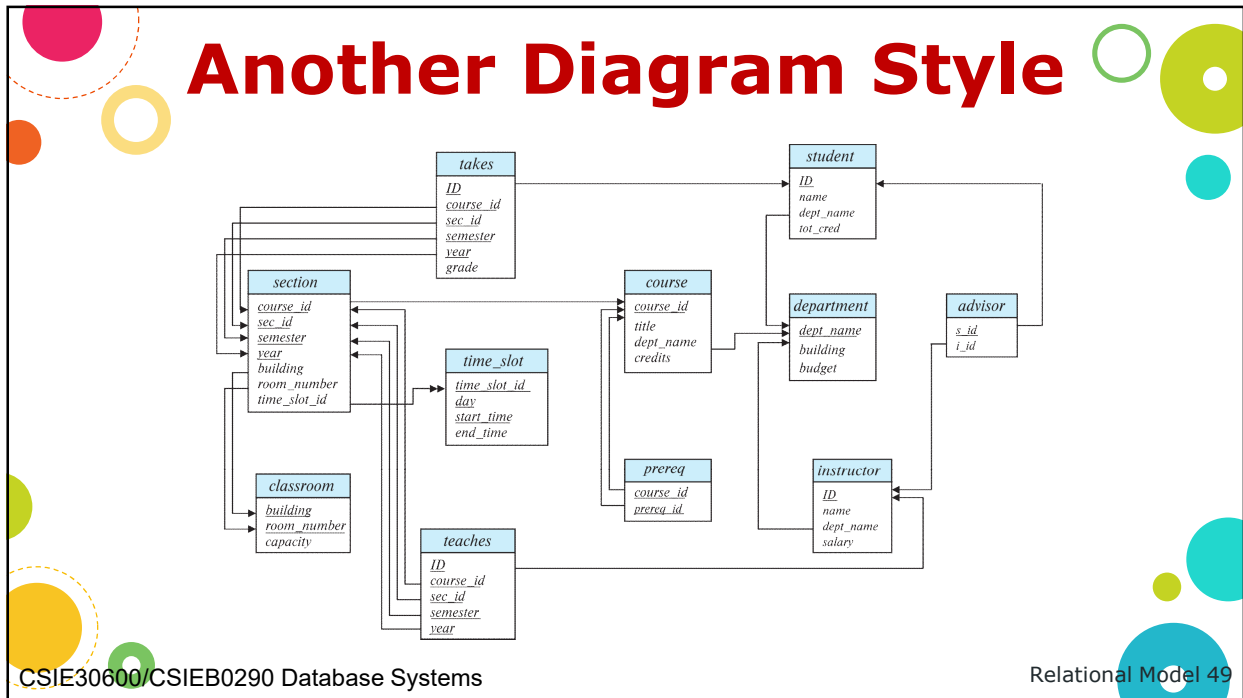
Displaying a Schema and Constraints

- ⦿ Each relation schema is 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**.

Relational Schema Diagram 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 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 constraints.
 - ⊙ More common to check for these types of constraints within the **application programs**.

CSIE30600/CSIEB0290 Database Systems Relational Model 50

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.

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.

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.

Update Operations on Relations

- Actions on integrity violation:**
 - 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**

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

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.

The Update Operation

- Need to specify a **condition** on attributes of relation
 - To 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

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 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**

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.