RV College of Engineering®

Go, change the world

Unit 2 (**Refining the ER Design for the COMPANY Database and Relational Model**)

Original Content:
Ramez Elmasri and Shamkant B. Navathe

Dr. Pratiba D
Assistant Professor, Department of CSE
RV College of Engineering, Bengaluru - 59

Refining ER Diagram for Company Database

The following are the binary relationship types observed:

1. Manages : 1:1 relationship type between EMPLOYEE and DEPARTMENT. Employee participation is partial. Department participation is not clear from requirements

2. WORKS-FOR : 1:N relationship type between DEPARTMENT and EMPLOYEE. Both participation are total.

3. SUPERVISION : 1:N relationship between employee and employee. Both participants are determined to be partial.

4. WORKS-ON : M:N relationship type with attributes Hours. Both participants are determined to be total.

5. CONTROLS : 1:1 relationship type between DEPARTMENT and PROJECT. The participation of the project is total and department is partial.

6. DEPENDENTS-OF : 1:N relationship type between EMPLOYEE and DEPENDENT . The participation of employee is partial and dependent is total.

ER-to-Relational Mapping Algorithm

- **Step 1: Mapping of Regular Entity Types**
  - For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
  - Choose one of the *key attributes* of E as the primary key for R.
  - If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.
- **Example:** We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram.
  - SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.

ER-to-Relational Mapping Algorithm (cont.)

- **Step 2: Mapping of Weak Entity Types**
  - For each weak entity type W in the ER schema with owner entity type E, *create* a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R.
  - Also, include as foreign key attributes of R the *primary key* attribute(s) of the relation(s) that correspond to the *owner entity* type(s).
  - The primary key of R is the *combination of the primary key(s) of the owner(s) and the partial key of the weak entity type* W, if any.
- **Example:** Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT.
  - Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN).
  - The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT

ER-to-Relational Mapping Algorithm (cont.)

- **Step 3: Mapping of Binary 1:1 Relation Types**
  - For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
  - There are three possible approaches:
    - *Foreign Key approach*: Choose one of the relations-S, say-and include as a foreign key in S the *primary key* of T. It is better to choose an entity type with *total* participation in R in the role of S.
      - **Example:** 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.
    - *Merged relation option*: An alternate mapping is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are *total*.
    - *Cross-reference or relationship relation option*: The third alternative is to set up a third relation R for the purpose of cross-referencing the *primary keys* of the two relations S and T representing the entity types.

ER-to-Relational Mapping Algorithm (cont.)

- **Step 4: Mapping of Binary 1:N Relationship Types**
  - For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the *N-side* of the relationship type.
  - Include as *foreign key* in S the *primary key* of the relation T.
  - Include any simple attributes of the 1:N relation type as attributes of S.
  - An alternative is use cross-reference like Step 3.
    - *Create a new relationship R* whose attributes are the *keys of S and T*, and whose primary key is the same as the *key of S*.
- **Example:** 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure.
  - For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.

ER-to-Relational Mapping Algorithm (cont.)

- **Step 5: Mapping of Binary M:N Relationship Types**
  - For each regular binary M:N relationship type R, *create a new relation* S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; *their combination will form the primary key* of S.
  - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.
- **Example:** The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema.
  - The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively.
  - Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

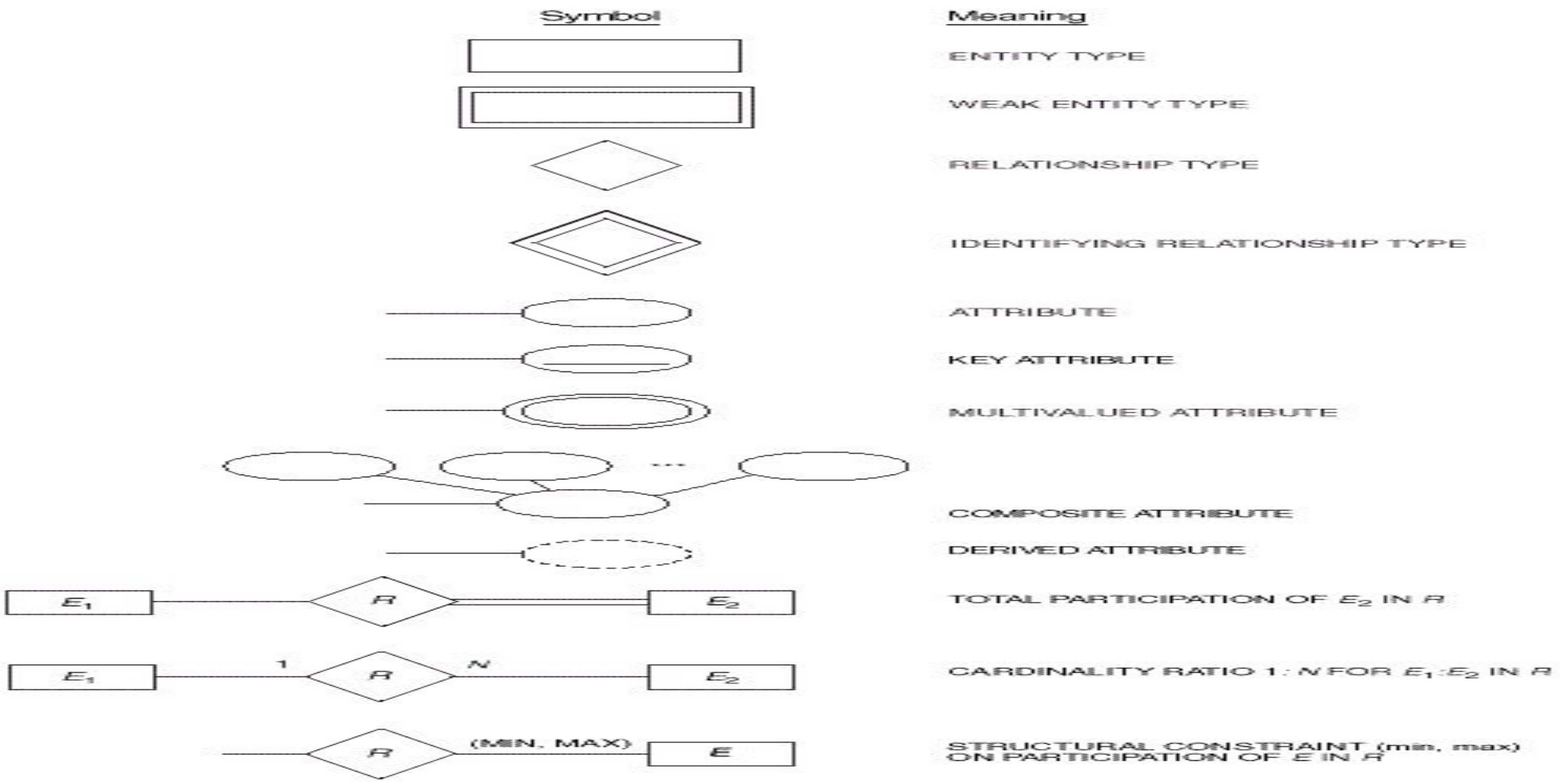
ER-to-Relational Mapping Algorithm (cont.)

- **Step 6: Mapping of Multivalued Attributes**
  - For each multivalued attribute A, *create a new relation R*.
  - This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
  - The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.
- **Example:** The relation DEPT_LOCATIONS is created.
  - The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation.
  - The primary key of R is the combination of {DNUMBER, DLOCATION}.

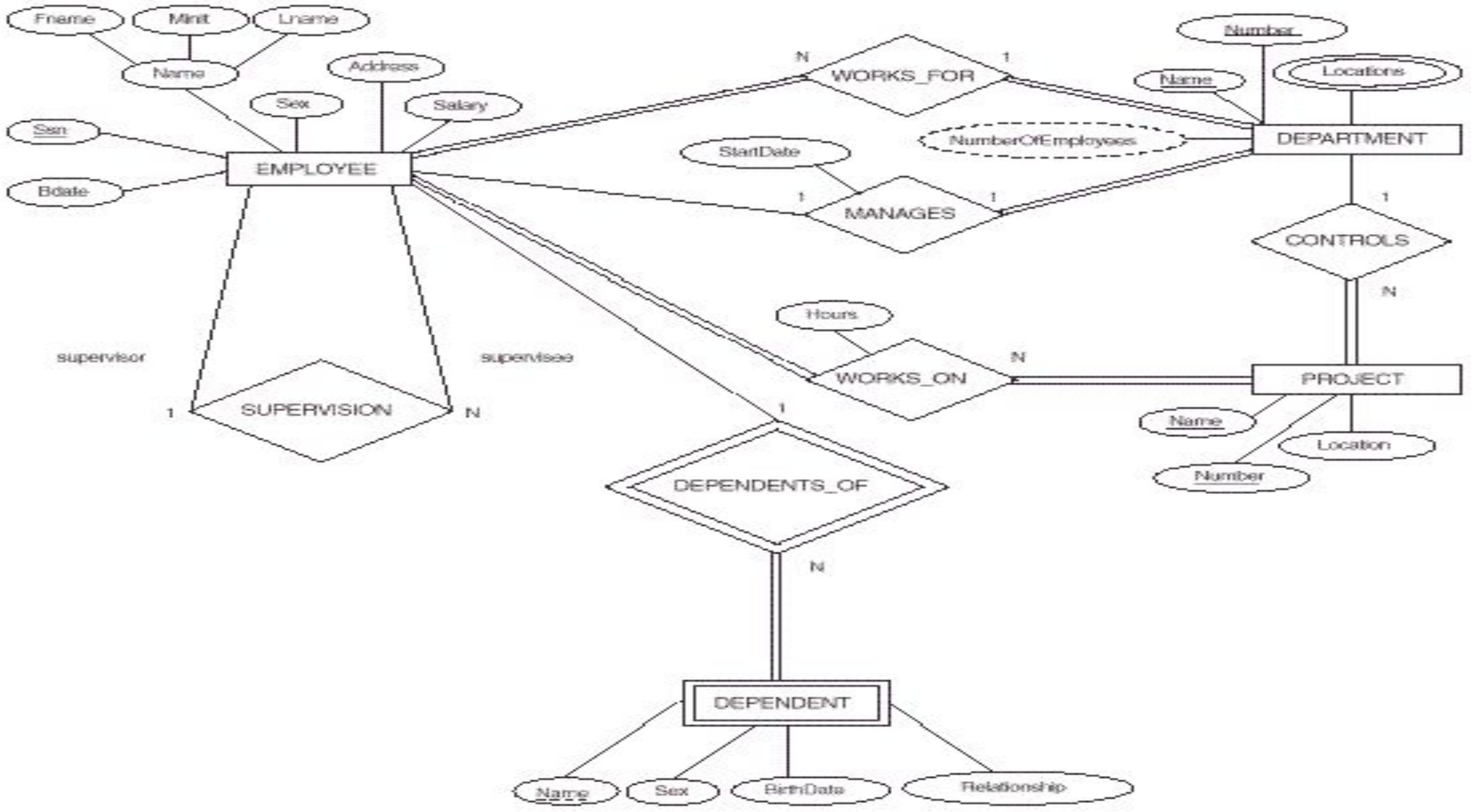
ER-to-Relational Mapping Algorithm (cont.)

- **Step 7: Mapping of N-ary Relationship Types**
  - For each n-ary relationship type R, where n>2, *create a new relationship* S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
    - The primary key of S is usually a combination of all the foreign keys.
  - Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.
- **Example:** The relationship type SUPPY in the ER on the next slide
  - This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}

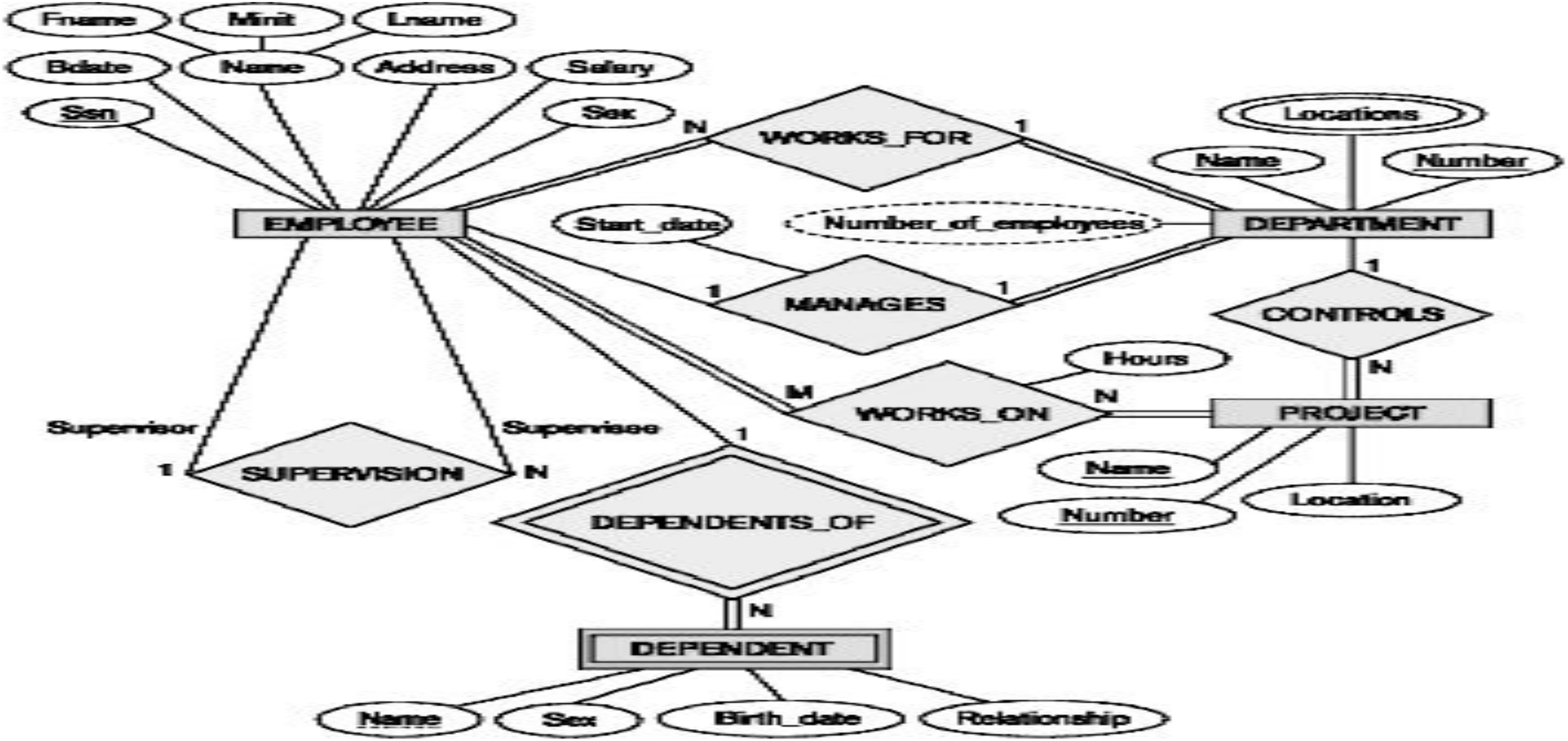
Summary of notation for ER Diagrams



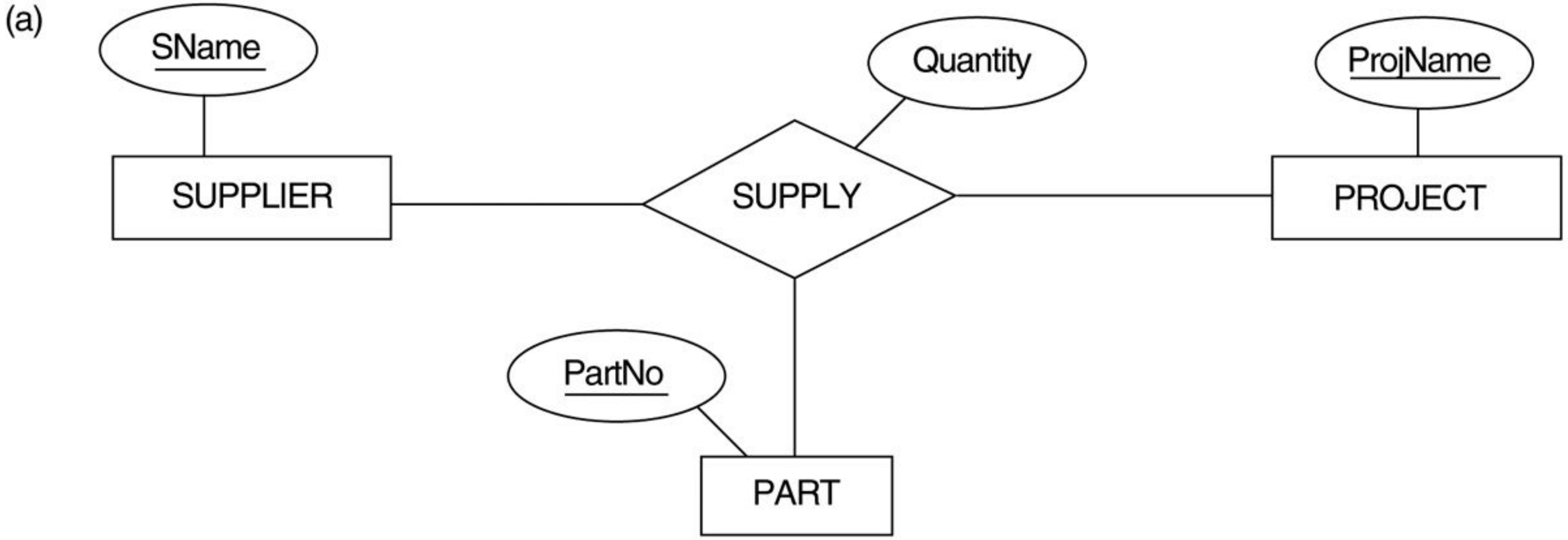
ER Diagram for Company Database



ER Diagram with Structural Constraints



Ternary relationship types.
(a) The SUPPLY relationship.



Mapping the *n*-ary relationship type SUPPLY

SUPPLIER

SNAME	. . .

PROJECT

PROJNAME	. . .

PART

PARTNO	. . .

SUPPLY

SNAME	PROJNAME	PARTNO	QUANTITY

Summary of Mapping Constructs and Constraints

Correspondence between ER and Relational Models

ER Model	Relational Model
Entity type	"Entity" relation
1:1 or 1:N relationship type	Foreign key (or "relationship" relation)
M:N relationship type	"Relationship" relation and two foreign keys
n-ary relationship type	"Relationship" relation and n foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

Relational Data Model

1. Relational Model Concepts

BASIS OF THE MODEL

The relational Model of Data is based on the concept of a **Relation.**

A Relation is a mathematical concept based on the ideas of sets.

The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.

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

RELATION: A table of values

A relation may be thought of as a **set of rows.**

A relation may alternately be thought of as a **set of columns.**

Each row of the relation may be given an identifier.

Each column typically is called by its column name or column header or attribute name.

.

FORMAL DEFINITIONS

A **Relation** may be defined in multiple ways.

The **Schema** of a Relation:

$R\ (A_1, A_2, \ldots..A_n)$

Relation R is defined over **attributes** $A_{1,}\ A_{2,}\ \ldots..A_n$

For Example -

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

Here, CUSTOMER is a relation defined over the four attributes Cust-id, Cust-name, Address, Phone#, each of which has a domain or a set of valid values. For example, the **domain** of Cust-id is 6 digit numbers.

.

FORMAL DEFINITIONS

A **tuple** is an ordered set of values
Each value is derived from an appropriate domain.
Each row in the CUSTOMER table may be called as a tuple in the table and would consist of four values.

<632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000"> is a triple belonging to the CUSTOMER relation.

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 - STATE

- Formally,
  - Given R(A1, A2, ..., An)
  - r(R) ⊂ 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, …, tm} where each ti is an n-tuple
  - ti = <v1, v2, …, vn> where each vj *element-of* dom(Aj)

FORMAL DEFINITIONS - STATE

- 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) ⊂ 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") of the relation R, defined over A1 and A2.
  - It has three 2-tuples: <0,a> , <0,b> , <1,c>

The attributes and tuples of a relation STUDENT

Relation name

Attributes

Tuples

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

A relation may be regarded as a set of tuples (rows). Columns in a table are also called as attributes of the relation.

The relation is formed over the cartesian product of the sets; each set has values from a domain; that domain is used in a specific role which is conveyed by the attribute name.

For example, attribute Cust-name is defined over the domain of strings of 25 characters. The role these strings play in the CUSTOMER relation is that of the name of customers.

Formally,

Given $R(A_1, A_2, \ldots\ldots\ldots, A_n)$

$r(R)$ <u>subset-of</u> $\text{dom}\,(A_1)\ X\ \text{dom}\,(A_2)\ X \ldots X\ \text{dom}(A_n)$

R: schema of the relation

r of R: a specific "value" or population of R.

R is also called the **intension** of a relation

r is also called the **extension** of a relation

Let S1 = {0,1}

Let S2 = {a,b,c}

Let R <u>subset-of</u> S1 X S2

for example: r(R) = {<0.a> , <0,b> , <1,c> }

DEFINITION SUMMARY

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

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

Values in a tuple: All values are considered *atomic* (indivisible). A special **null** value is used to represent values that are unknown or inapplicable to certain tuples.

Relation STUDENT with different order of tuples

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

Two identical tuples when order of attributes and values is not part of the definition of a relation

t = < (Name, Dick Davidson),(SSN, 422-11-2320),(HomePhone, null),(Address, 3452 Elgin Road),
(OfficePhone, 749-1253),(Age, 25),(GPA, 3.53)>

t = < (Address, 3452 Elgin Road),(Name, Dick Davidson),(SSN, 422-11-2320),(Age, 25),
(OfficePhone,749-1253),(GPA, 3.53),(HomePhone, null)>

Notation:

- We refer to **component values** of a tuple t by $t[A_i] = v_i$ (the value of attribute A_i for tuple t).

Similarly, $t[A_u, A_v, ..., A_w]$ refers to the subtuple of t containing the values of attributes A_u, A_v, ..., A_w, respectively.

Characteristics Of Relations (cont.)

- 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 inapplicable to certain tuples

Characteristics Of Relations (cont.)

- 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
  - For example
    - In STUDENT relation, t=<‘Barbara Benson’, ‘533-69-1238’, ‘839-8461’, ‘7384 Fontana Lane’, NULL, 19, 3.25>
    - t[Name]=<‘Barbara Benson’>, t[Ssn, Gpa, Age]=<‘533-69-1238’, 3.25, 19>

3 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

3.1 Key Constraints

Superkey of R: A set of attributes SK of R such that no two tuples *in any valid relation instance r(R)* will have the same value for SK. That is, for any distinct tuples t_1 and t_2 in r(R), $t_1[SK] <> t_2[SK]$.

Key Constraints

- **Superkey (SK) 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] ≠ 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)

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 relation with 2 candidate keys License Number and Engine Serial Number

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

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}
  - 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

Schema diagram for the company relational DB

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

One possible relational DB state corresponding to the COMPANY DB

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John		Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin		Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia		Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer		Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh		Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce		English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad		Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James		Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
		Houston
		Stafford
		Bellaire
		Sugarland

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

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

3.2 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.
    - $t[PK] \neq null$ 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.

3.3 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**.

3.3 Referential Integrity(cont.)

- 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.

3.3 Referential Integrity(or Foreign Key)

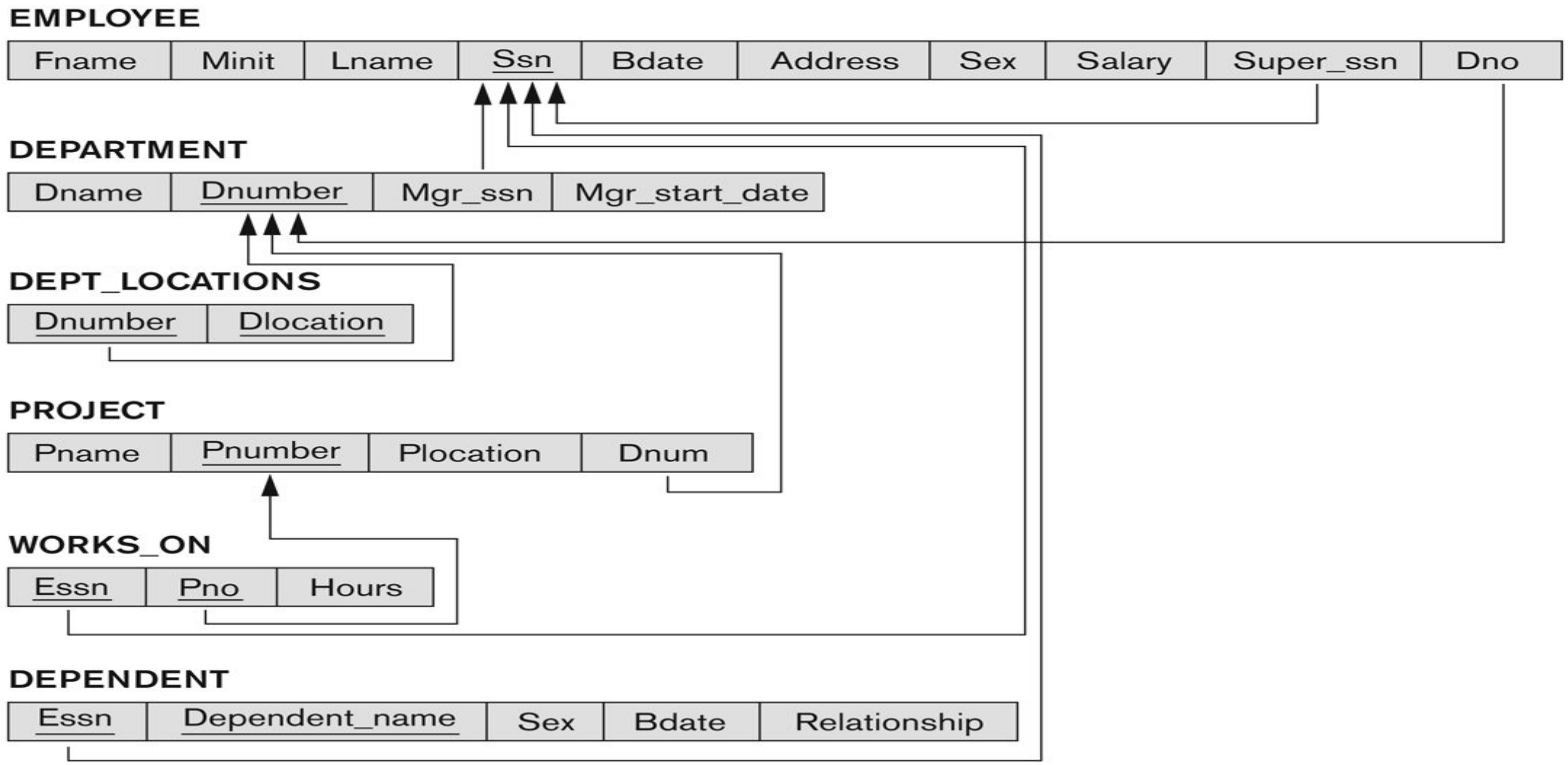
- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the the **referencing relation** R1 can be **either**:
    - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, <u>or</u>
    - (2) a **null**.
- In 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 the the primary key of the referenced relation for clarity
- Next slide shows the COMPANY **relational schema diagram**

Referential integrity constraints displayed on the COMPANY relational DB schema diagram

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

4 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 deletion by deleting the tuples that are referencing primary key that is being deleted.
      - 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

Possible violations for each operation

- 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.

Thank YOU