

CS4.301 Data & Applications

Ponnurangam Kumaraguru ("PK")
#ProfGiri @ IIIT Hyderabad



pk.profgiri



/in/ponguru



@ponguru



Ponnurangam.kumaraguru

Weak Entity Types

An entity that does not have a key attribute and that is identification-dependent on another entity type.

A weak entity must participate in an identifying relationship type with an owner or identifying entity type

Entities are identified by the combination of:

- A partial key of the weak entity type

- The particular entity they are related to in the identifying relationship type

Example:

A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related

Name of DEPENDENT is the *partial key*

DEPENDENT is a *weak entity type*

EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Attributes of Relationship types

A relationship type can have attributes:

For example, HoursPerWeek of WORKS_ON

Its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.

A value of HoursPerWeek depends on a particular (employee, project) combination

Most relationship attributes are used with M:N relationships

Example Attribute of a Relationship Type: Hours of WORKS_ON

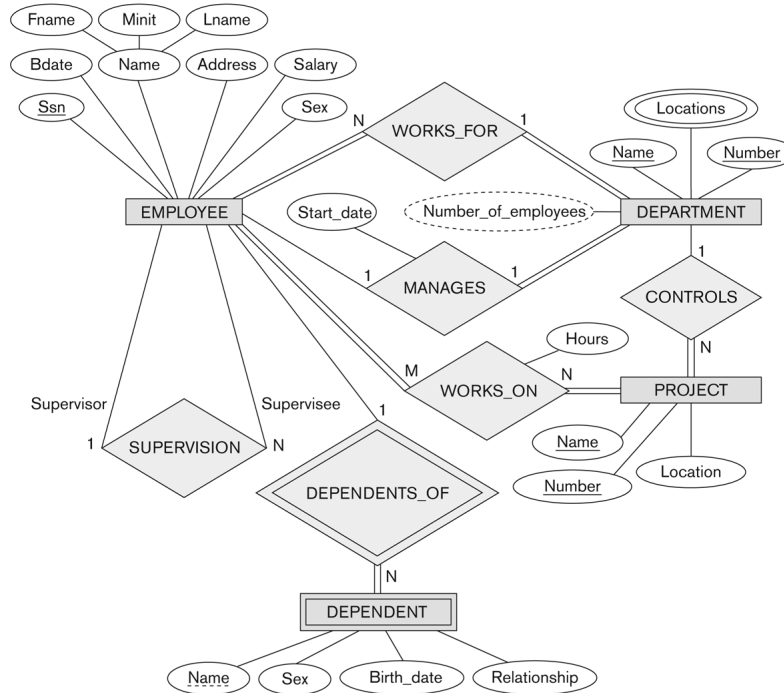


Figure 3.2

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Notation for Constraints on Relationships

Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N

Shown by placing appropriate numbers on the relationship edges.

Participation constraint (on each participating entity type): total (called existence dependency) or partial.

Total shown by double line, partial by single line.

Alternative (min, max) notation for relationship structural constraints:

Specified on each participation of an entity type E in a relationship type R

Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R

Default(no constraint): min=0, max=n (signifying no limit)

Must have $\min \leq \max$, $\min \geq 0$, $\max \geq 1$

Derived from the knowledge of mini-world constraints

Cardinality & Participation taken together called structural constraints; (m,n); m = 0 is partial, m = 1 total

Examples:

A department has exactly one manager and an employee can manage at most one department.

Specify (0,1) for participation of EMPLOYEE in MANAGES

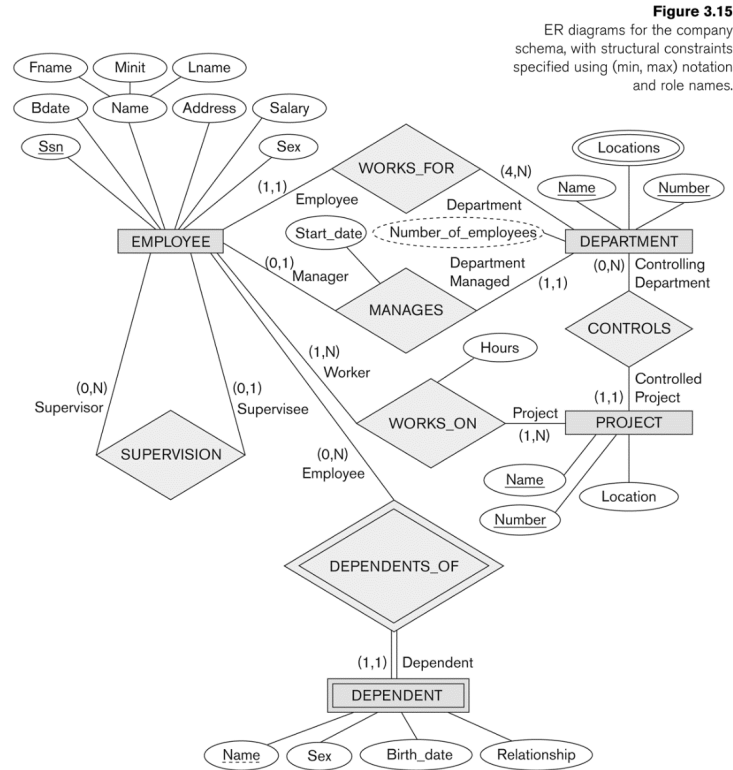
Specify (1,1) for participation of DEPARTMENT in MANAGES

An employee can work for exactly one department but a department can have any number of employees.

Specify (1,1) for participation of EMPLOYEE in WORKS_FOR

Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

COMPANY ER Schema Diagram using (min, max) notation



Alternative diagrammatic notation

ER diagrams is one popular example for displaying database schemas

Many other notations exist in the literature and in various database design and modeling tools

UML class diagrams is representative of another way of displaying ER concepts that is used in several commercial design tools

UML class diagrams

Represent classes (similar to entity types) as large rounded boxes with three sections:

- Top section includes entity type (class) name

- Second section includes attributes

- Third section includes class operations (operations are not in basic ER model)

Relationships (called associations) represented as lines connecting the classes

- Other UML terminology also differs from ER terminology

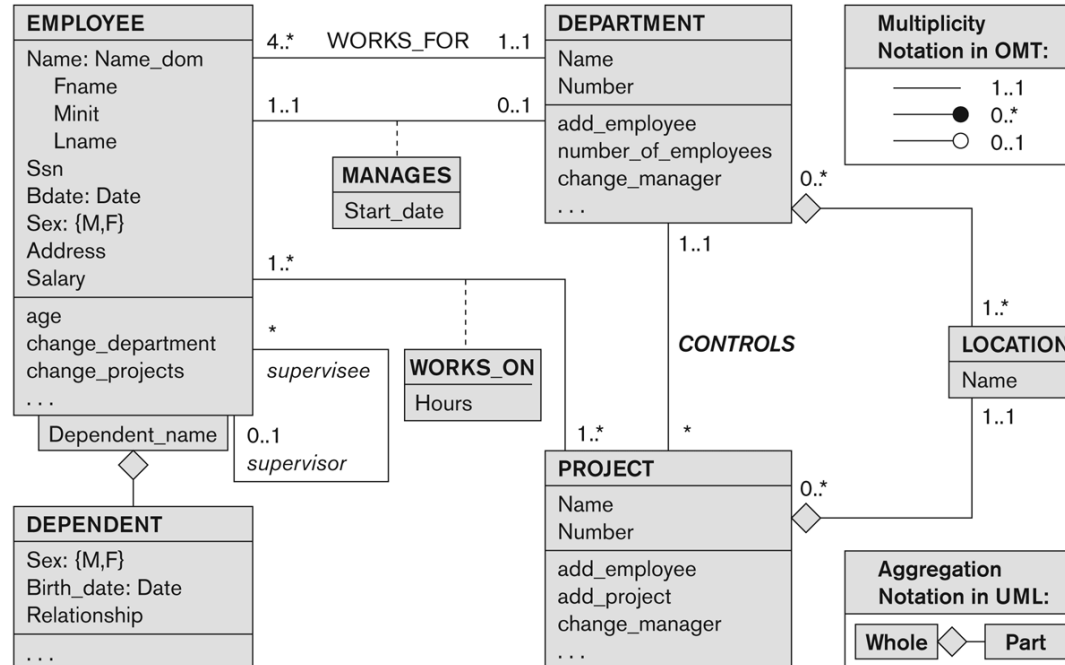
Used in database design and object-oriented software design

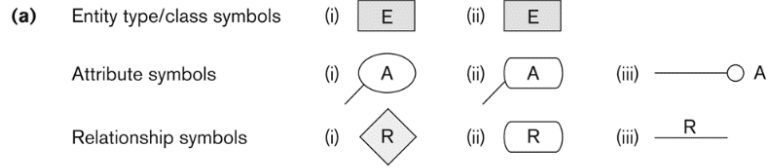
UML has many other types of diagrams for software design

UML class diagram for COMPANY database schema

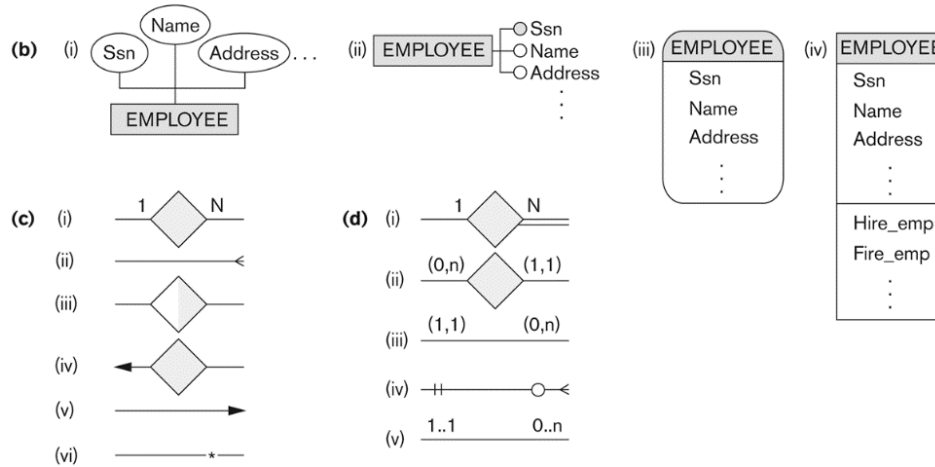
Figure 3.16

The COMPANY conceptual schema in UML class diagram notation.





Other alternative
diagrammatic notations



Some of the Automated Database Design Tools (Note: Not all may be on the market now)

COMPANY	TOOL	FUNCTIONALITY
Embarcadero Technologies	ER Studio	Database Modeling in ER and IDEF1X
	DB Artisan	Database administration, space and security management
Oracle	Developer 2000/Designer 2000	Database modeling, application development
Popkin Software	System Architect 2001	Data modeling, object modeling, process modeling, structured analysis/design
Platinum (Computer Associates)	Enterprise Modeling Suite: Erwin, BPWin, Paradigm Plus	Data, process, and business component modeling
Persistence Inc.	Pwertier	Mapping from O-O to relational model
Rational (IBM)	Rational Rose	UML Modeling & application generation in C++/JAVA
Resolution Ltd.	Xcase	Conceptual modeling up to code maintenance
Sybase	Enterprise Application Suite	Data modeling, business logic modeling
Visio	Visio Enterprise	Data modeling, design/reengineering Visual Basic/C++

The Relational Data Model and Relational Database Constraints

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 module

In *practice*, there is a *standard model* based on SQL – We will see this as next module

Example of a Relation

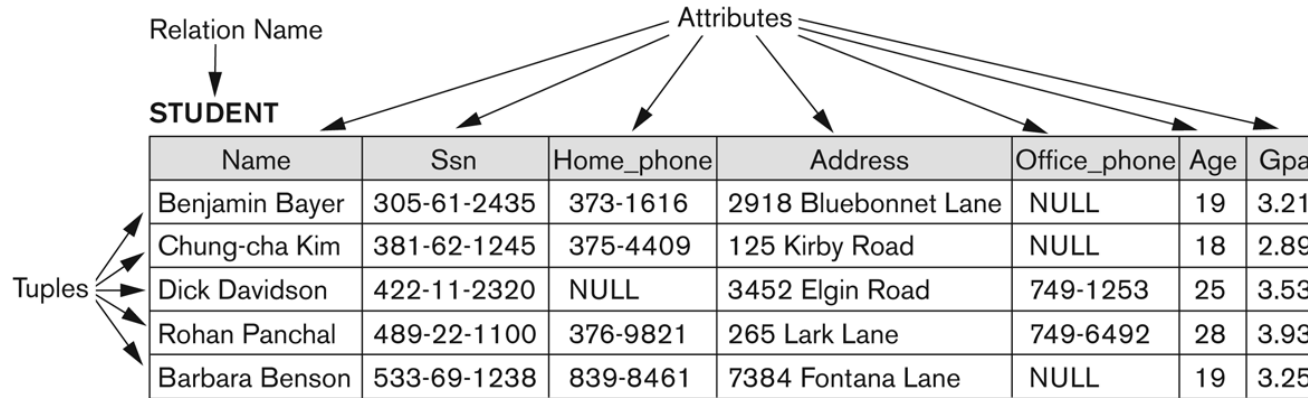


Figure 5.1

The attributes and tuples of a relation STUDENT.

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 '< ... >')

Each value is derived from an appropriate *domain*.

A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:

<632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">

This is called a 4-tuple as it has 4 values

A tuple (row) in the CUSTOMER relation.

A relation is a **set** of such tuples (rows)

Formal Definitions- Domain

A **domain** has a logical definition:

Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.

A domain also has a data-type or a format defined for it.

The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.

Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.

The attribute name designates the role played by a domain in a relation:

Used to interpret the meaning of the data elements corresponding to that attribute

Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings

Formal Definitions- State

The **relation state** is a subset of the Cartesian product of the domains of its attributes

each domain contains the set of all possible values the attribute can take.

Example: attribute Cust-name is defined over the domain of character strings of maximum length 25

dom(Cust-name) is varchar(25)

The role these strings play in the CUSTOMER relation is that of the *name of a customer*.

Formal Definitions- Summary

Formally,

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

$$r(R) \subset \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$$

$R(A_1, A_2, \dots, A_n)$ 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 [All Rows in a table]

$t_i = \langle v_1, v_2, \dots, v_n \rangle$ where each v_j *element-of* $\text{dom}(A_j)$ [Single Row in the table]

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

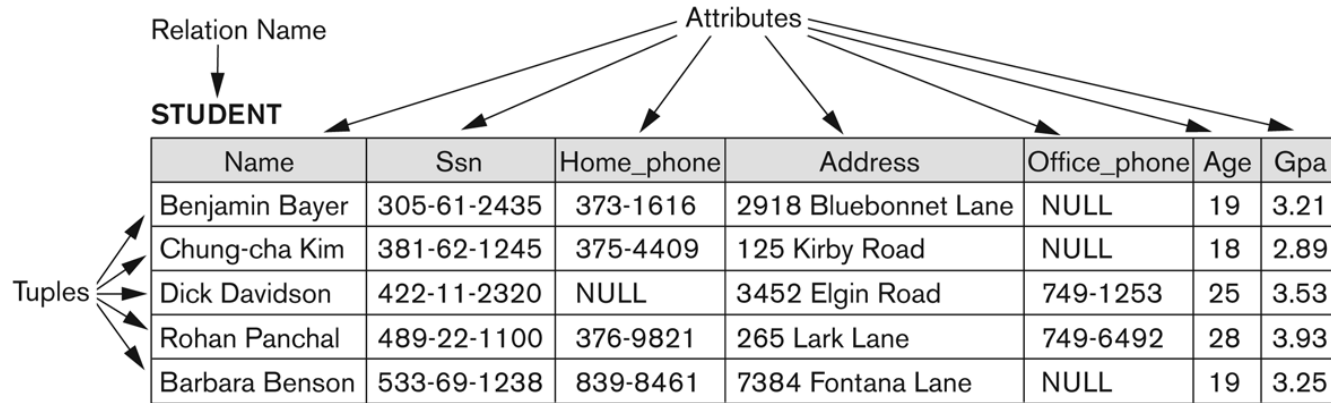


Figure 5.1

The attributes and tuples of a relation STUDENT.

Characteristics Of Relations

Ordering of tuples in a relation $r(R)$:

The tuples are *not considered to be ordered*, even though they appear to be in the tabular form.

Ordering of attributes in a relation schema R (and of values within each tuple):

We will consider the attributes in $R(A_1, A_2, \dots, A_n)$ 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. It includes both the name and the value for each of the attributes).

Example: $t = \{ \langle \text{name}, \text{"John"} \rangle, \langle \text{SSN}, 123456789 \rangle \}$

This representation may be called as “self-describing”.

Same state as previous Figure (but with different order of tuples)

Figure 5.2

The relation STUDENT from Figure 5.1 with a different order of tuples.

STUDENT

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

Characteristics Of Relations

Values in a tuple:

All values are considered atomic (indivisible).

Each value in a tuple must be from the domain of the attribute for that column

If tuple $t = \langle v_1, v_2, \dots, v_n \rangle$ 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 not available or inapplicable in 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

<u>Formal Terms</u>
Relation
Attribute
Domain
Tuple
Schema of a Relation
State of the Relation

This lecture

CONSTRAINTS

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

They are of three main types:

1. **Inherent or Implicit Constraints:** These are based on the data model itself. (E.g., relational model does not allow a list as a value for any attribute)
2. **Schema-based or Explicit Constraints:** They are expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model)
3. **Application based or semantic constraints:** These are beyond the expressive power of the model and must be specified and enforced by the application programs. (E.g. Bookmytickets allowing only 4 tickets to be booked for a user at a given point in time)

Relational Integrity Constraints

Constraints are **conditions** that must hold on **all** valid relation states.

There are three *main types* of (explicit schema-based) constraints that can be expressed in the relational model:

- Key** constraints

- Entity integrity** constraints

- Referential integrity** constraints

Another schema-based constraint is the **domain** constraint

- Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

Key Constraints

Superkey of R:

Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation.

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

A candidate key (or key) is a set of attributes (or attributes) that uniquely identify the tuples in relation to or table.

Key Constraints (continued)

Example: Student{ID, First_name, Last_name, Age, Sex, Phone_no}

Two candidate keys: ID & {First_name, Last_name, Age, Sex, Phone_no}

Consider the STUDENT relation schema:

Attribute Ssn is a key, because no 2 students tuples can have the same Ssn

Any set of attributes that include Ssn, like {Ssn, Name, Age} is a superkey it uniquely identifies all attributes in a relation

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

Candidate key & Super key difference

Super Key	Candidate Key
Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation.	Candidate Key is a subset of a super key.
All super keys can't be candidate keys.	But all candidate keys are super keys.
Various super keys together makes the criteria to select the candidate keys.	Various candidate keys together makes the criteria to select the primary keys.
In a relation, number of super keys is more than number of candidate keys.	While in a relation, number of candidate keys are less than number of super keys.
Super key attributes can contain NULL values.	Candidate key attributes can also contain NULL values.

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:

License_number & Engine_serial_number

We chose Engine_serial_number 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

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

Super & Primary key difference

S.NO	Super Key	Primary Key
1.	Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation.	Primary Key is a minimal set of attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation.
2.	All super keys can't be primary keys.	Primary key is a minimal super key.
3.	Various super keys together makes the criteria to select the candidate keys.	We can choose any of the minimal candidate key to be a primary key.
4.	In a relation, number of super keys are more than number of primary keys.	While in a relation, number of primary keys are less than number of super keys.
5.	Super key's attributes can contain NULL values.	Primary key's attributes cannot contain NULL values.

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**

$S = \{R_1, R_2, \dots, R_n\}$ and a set IC of integrity constraints.

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

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.

Relational Database State

A **relational database state** DB of S is a set of relation states $DB = \{r_1, r_2, \dots, r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy the integrity constraints specified in IC.

A relational database *state* is sometimes called a relational database *snapshot* or *instance*.

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

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

Populated DB state for COMPANY

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

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 **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 with referential integrity constraints**

Referential integrity constraints for Company

Figure 5.7

Referential integrity constraints displayed on the COMPANY relational 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

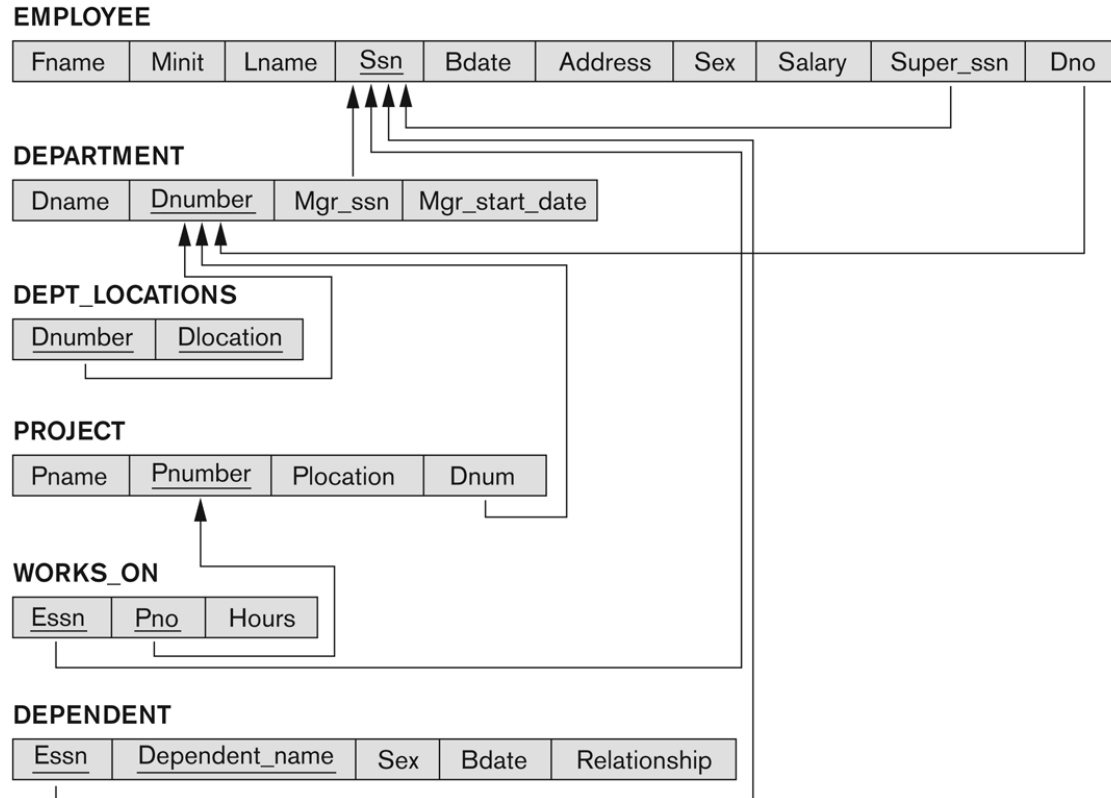
Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

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

DEPENDENT

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



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
- 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

Referential integrity constraints for Company

Figure 5.7

Referential integrity constraints displayed on the COMPANY relational 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

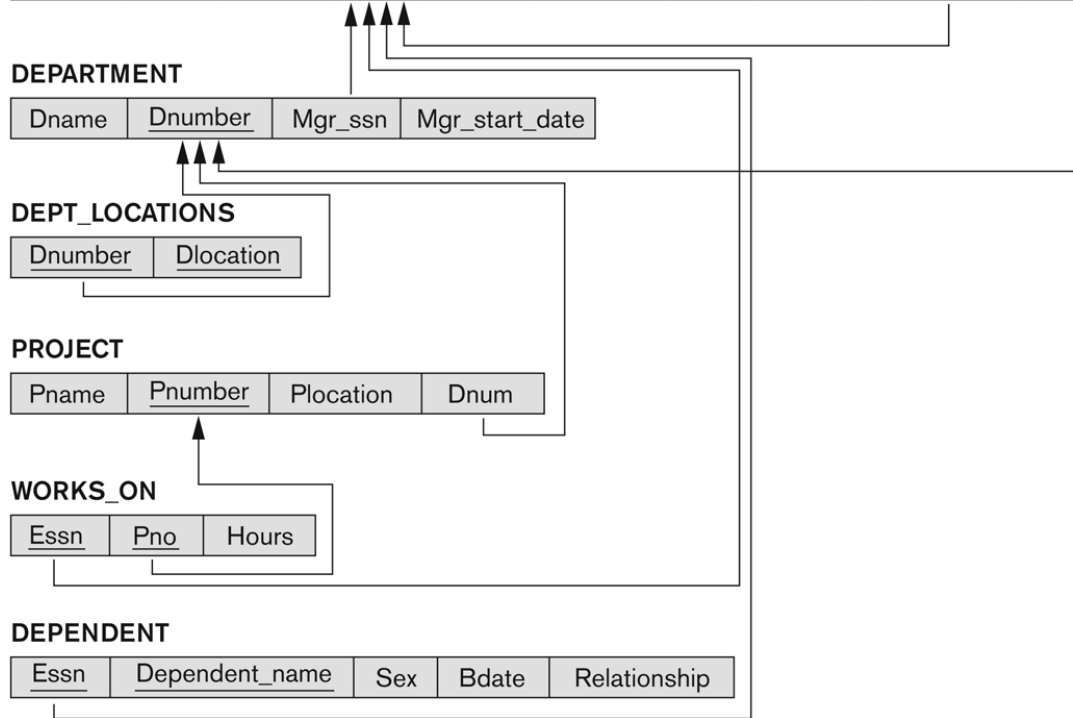
Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

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

DEPENDENT

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



Operation:

Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.

Result: This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected.

Operation:

Insert <'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.

Result: This insertion violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.

Operation:

Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> into EMPLOYEE.

Result: This insertion violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.

What are the results?

Figure 5.6

One possible database state for the COMPANY relational database schema.

Figure 5.7

Referential integrity constraints displayed on the COMPANY relational 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
-------------	-----------------------	-----	-------	--------------

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	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	<u>Dnumber</u>	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

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

<u>Essn</u>	<u>Pno</u>	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	<u>Pnumber</u>	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

<u>Essn</u>	<u>Dependent_name</u>	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

Operation:

Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.

Result: This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected.

Operation:

Insert <'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.

Result: This insertion violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.

Operation:

Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> into EMPLOYEE.

Result: This insertion violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.

What are the results?

Operation:

Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.

Result: This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected.

Operation:

Insert <'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.

Result: This insertion violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.

Operation:

Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> into EMPLOYEE.

Result: This insertion violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.

What are the results?

Operation:

Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.

Result: This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected.

Operation:

Insert <'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.

Result: This insertion violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.

Operation:

Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> into EMPLOYEE.

Result: This insertion violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.

What are the results?

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

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

Operation:

Delete the WORKS_ON tuple with Essn = '999887777' and Pno = 10.

Result: This deletion is acceptable and deletes exactly one tuple.

Operation:

Delete the EMPLOYEE tuple with Ssn = '999887777'.

Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

Operation:

Delete the EMPLOYEE tuple with Ssn = '333445555'.

Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations.

What are the results?

Figure 5.6

One possible database state for the COMPANY relational database schema.

Figure 5.7

Referential integrity constraints displayed on the COMPANY relational 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
-------------	-----------------------	-----	-------	--------------

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	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	<u>Dnumber</u>	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

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

<u>Essn</u>	<u>Pno</u>	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	<u>Pnumber</u>	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

<u>Essn</u>	<u>Dependent_name</u>	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

Operation:

Delete the WORKS_ON tuple with Essn = '999887777' and Pno = 10.

Result: This deletion is acceptable and deletes exactly one tuple.

Operation:

Delete the EMPLOYEE tuple with Ssn = '999887777'.

Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

Operation:

Delete the EMPLOYEE tuple with Ssn = '333445555'.

Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations.

What are the results?

Operation:

Delete the WORKS_ON tuple with Essn = '999887777' and Pno = 10.

Result: This deletion is acceptable and deletes exactly one tuple.

Operation:

Delete the EMPLOYEE tuple with Ssn = '999887777'.

Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

Operation:

Delete the EMPLOYEE tuple with Ssn = '333445555'.

Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations.

What are the results?

Operation:

Delete the WORKS_ON tuple with Essn = '999887777' and Pno = 10.

Result: This deletion is acceptable and deletes exactly one tuple.

Operation:

Delete the EMPLOYEE tuple with Ssn = '999887777'.

Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

Operation:

Delete the EMPLOYEE tuple with Ssn = '333445555'.

Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations.

What are the results?

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e

 pk.profgiri

 Ponnurangam.kumaraguru

 /in/ponguru

 ponguru

 pk.guru@iiit.ac.in

Thank you
for attending
the class!!!