

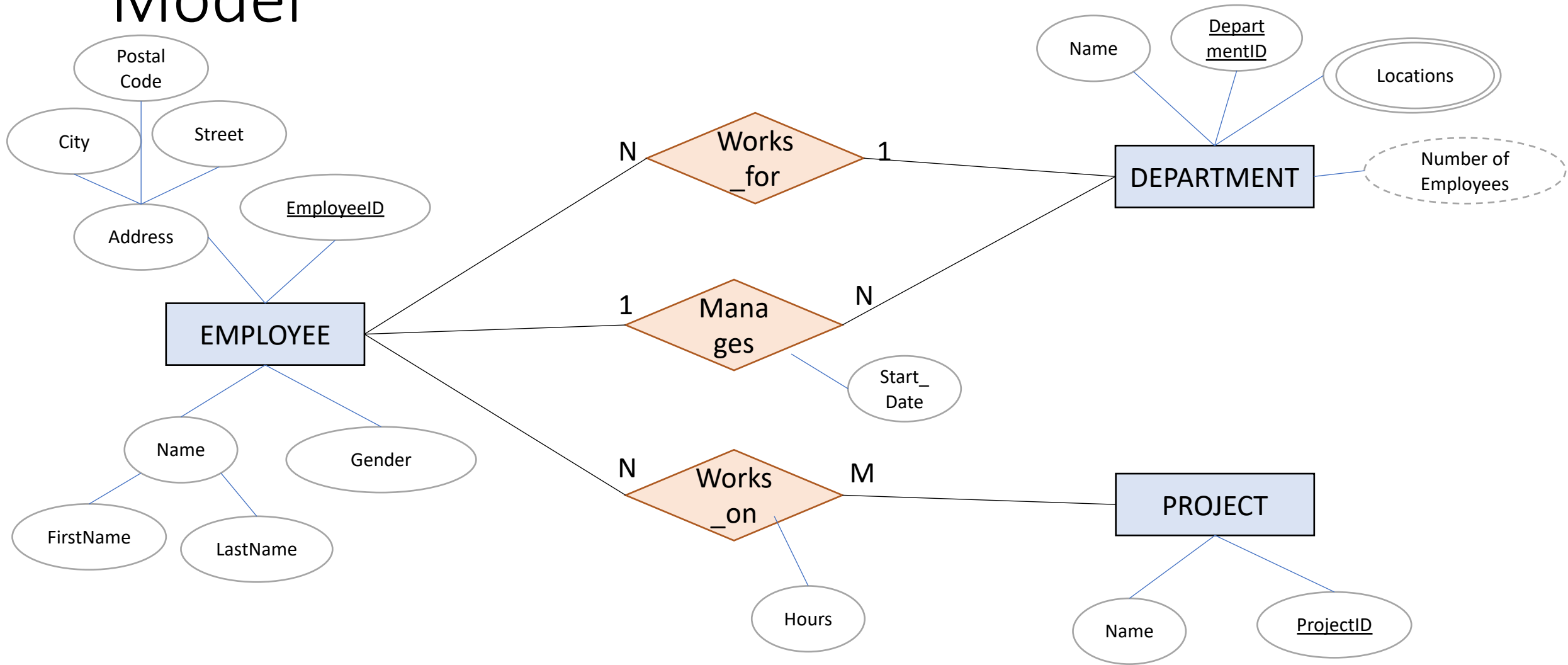
# Lecture 4. Relational Data Model (Chapter 5)

[alisa.lincke@lnu.se](mailto:alisa.lincke@lnu.se)

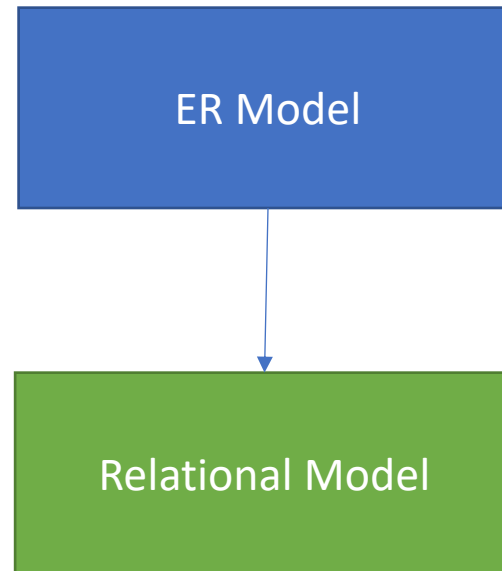
# Outline

- From Conceptual Models to Physical Models (ER to Relational Model)
- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- Update Operations and Dealing with Constraint Violations

# From ER (Conceptual model) to Relational Model



# From ER (Conceptual model) to Relational Model



# From ER to the Relational Model (Conventions)

- **Step 1 – Strong Entities**

A light blue rectangular box with a thin black border, containing the word "EMPLOYEE" in black, uppercase letters.

- Create a **table** that includes all the *simple* attributes
- Include only the *simple components* of composite attributes
- Choose a **primary key**
  - **Primary key** is the column or columns that contain values that uniquely identify each row in table. Primary keys must contain UNIQUE values, and cannot contain NULL values. The table can not have more then one primary key.

- **Step 2 – Weak Entities**

A light blue rectangular box with a thin black border, containing the word "DEPENDENT" in black, uppercase letters.

- Create a table that includes all the simple attributes
- Include the *primary key* from the **owner** entity, this will become a **foreign key**
  - **Foreign key** is a column (or columns) in one table that refers to the primary key in another table.

# From ER to the Relational Model (Conversion Guide)



- **Step 3 – 1:1 Relations**

- Choose one of the entities (right or left). It is better to choose the entity type with total participation (double line)
- Copy the **primary key** from the first table into second table – it becomes a *foreign key* in the second table
- Any attributes tied to the relationship goes to the table with the foreign key



- **Step 4 – 1:N relations**

- Choose the entity on the N-side and include copy of the primary key of the entity on the 1-side. The column becomes a foreign key in the table on the N-side
- Any attributes tied to the relationship goes to the table with the foreign key

# From ER to the Relational Model (Conversion Guide)

- **Step 5 – N:M Relations**



- Create a new table and include the primary keys from the two participating entities
- Both of these fields become foreign keys in the new table
- Any attributes tied to the relationship goes to the new table

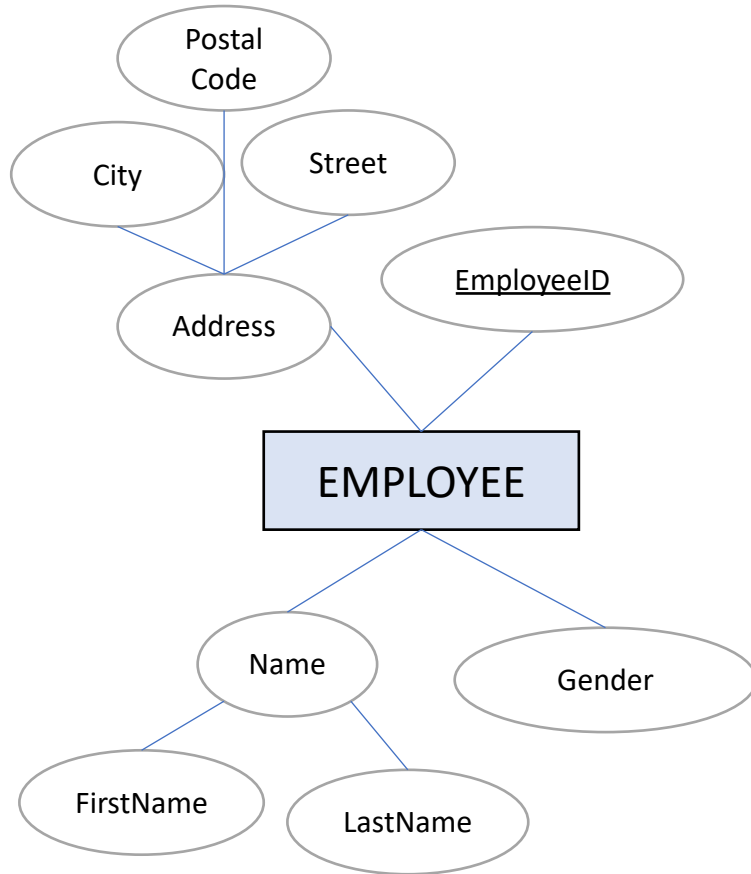
- **Step 6 – Multivalued Attributes**

- For each multivalued attribute (A) create a new table and move the attribute (A) to this table
- Include the primary key from the entity that originally had (A) as an attribute
- The combination of these attributes will be the primary key of the new table

- **Step 7 – Derived Attributes**

- Coded separately in SQL as a view. They are **not an attributes** in a relation table.

# Step 1 - Strong Entities (1)



Employee	
PK	<u>EmployeeID</u>
	Gender
	Birthday
	City
	Street
	Postal Code City
	FirstName
	LastName



# Step 1 - Strong Entities (2)

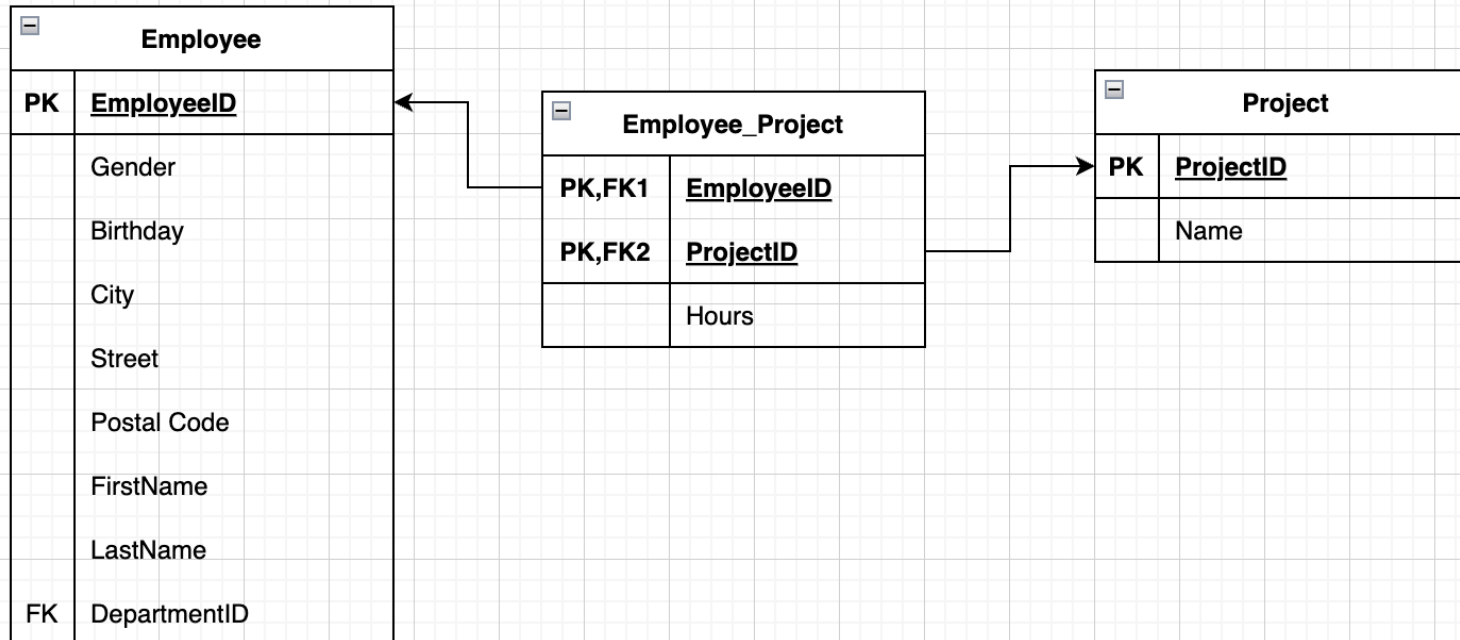
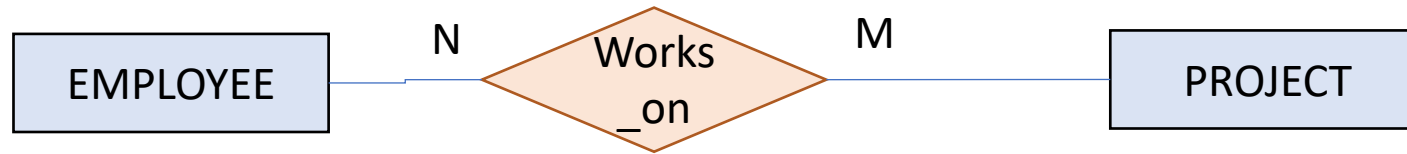
Employee	
PK	<u>EmployeeID</u>
	Gender
	Birthday
	City
	Street
	Postal Code City
	FirstName
	LastName

Department	
PK	<u>Department ID</u>
	Name
FK	LocationID

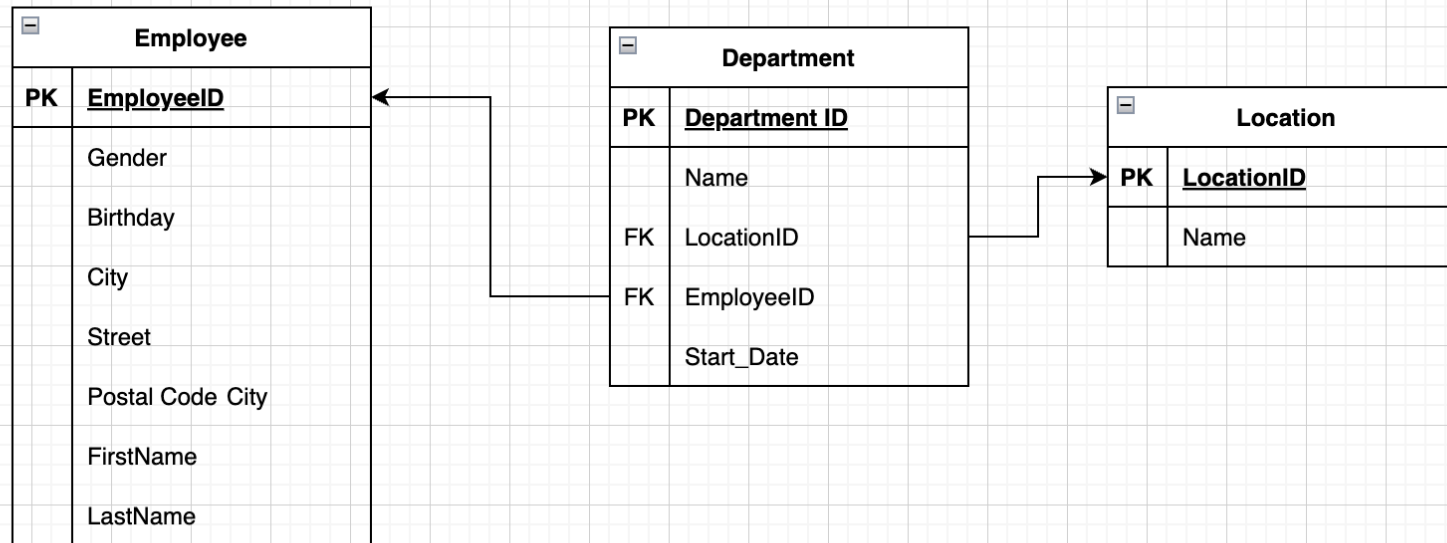
Location	
PK	<u>LocationID</u>
	Name

Project	
PK	<u>ProjectID</u>
	Name

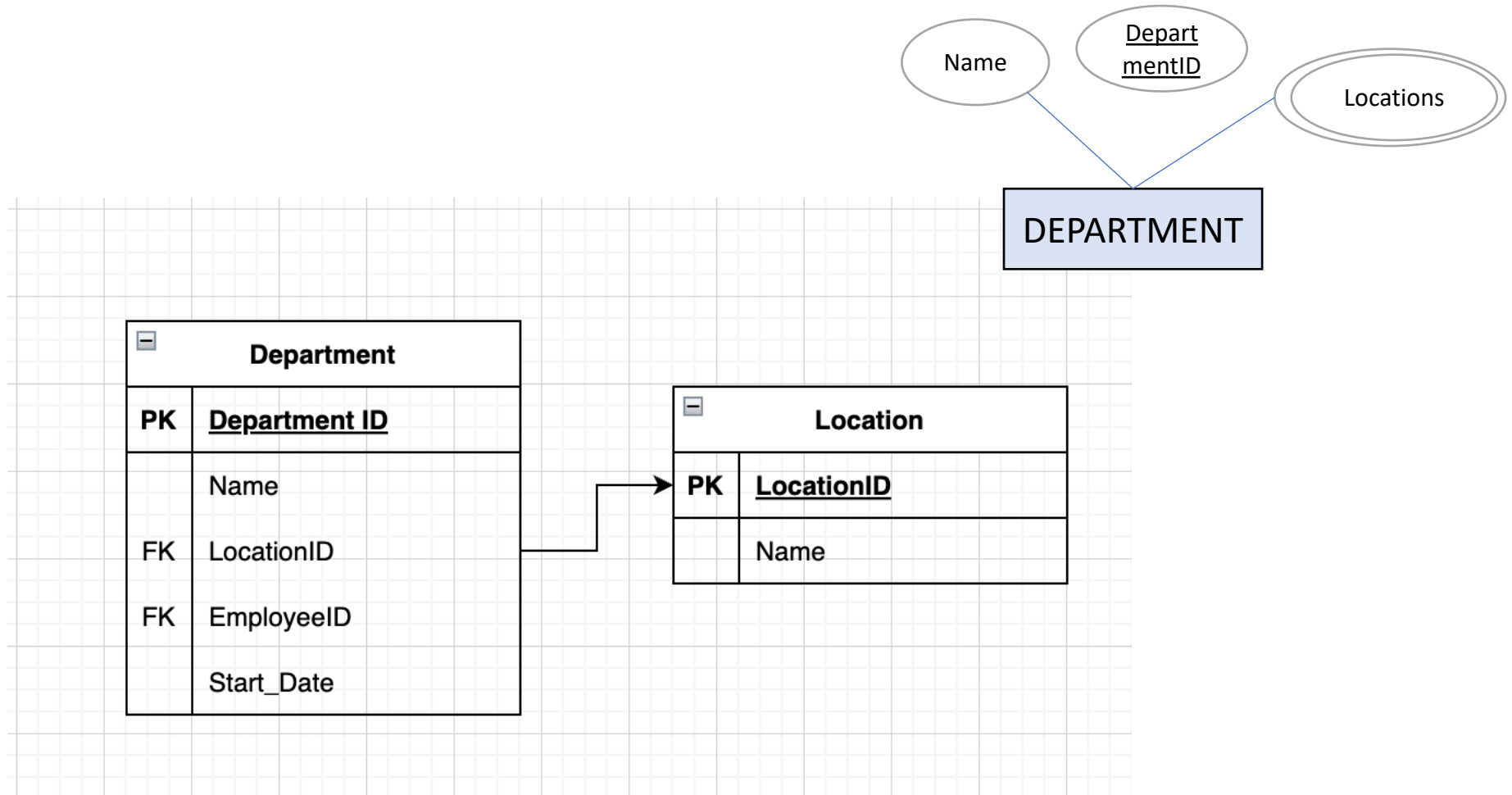
## Step 2 – Relations (N:M)



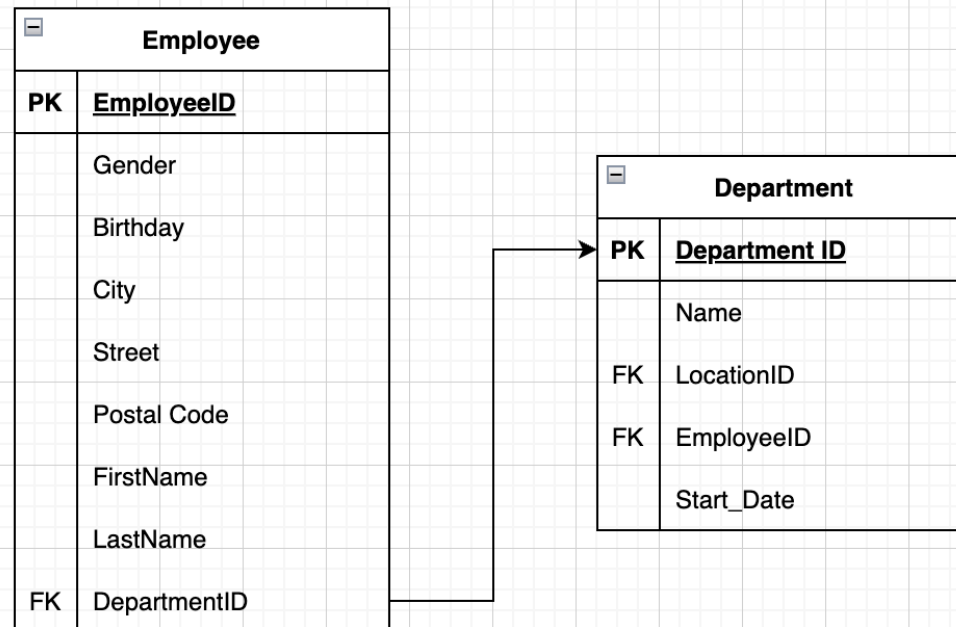
# Step 3 –Relation (1:N)



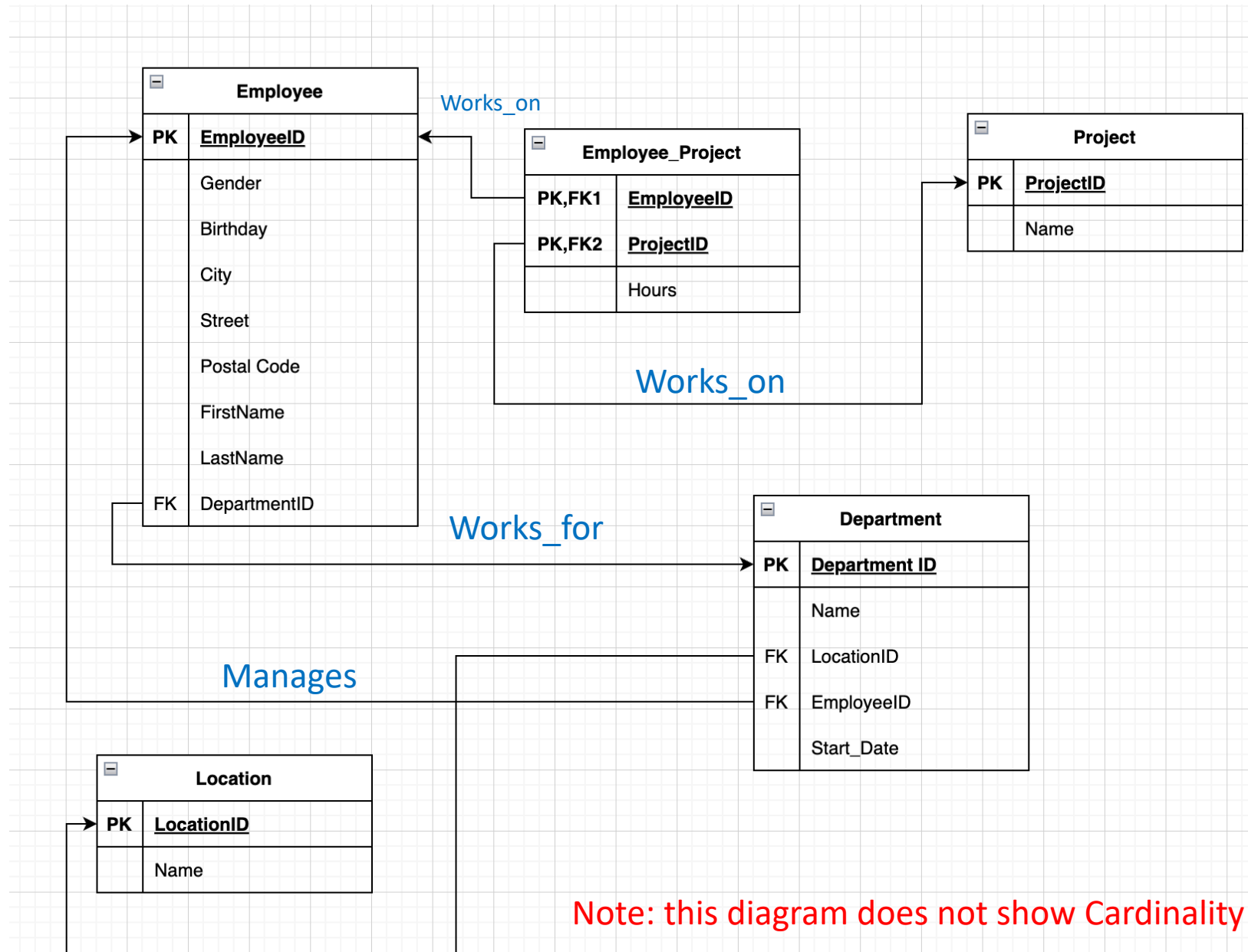
# Step 4 – Multivalued Attributes



# Step 3 – Relation (N:1)



# Relation Model



# Terminology

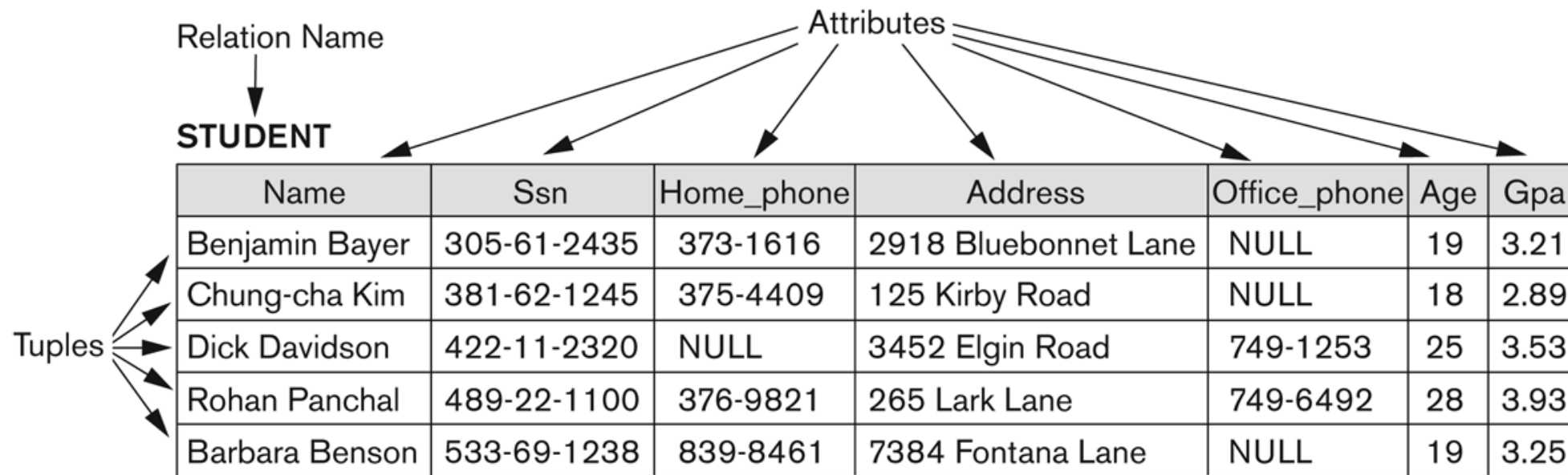
<b>Informal (Casual)</b>	<b>Formal (DBMS)</b>
Table	Relation
Columns/fields/column header	Attributes
All possible column values	Domain
Row/Rows	Tuple/Tuples
Table definition	Schema of relation
Populated Table	State of relation

# Relational Data Model Concepts

- Relation data model is used for data storage and processing.
- A relation represented as a **table** with **columns** (or *attributes*) and **rows** (or *tuples*)
  - It is based on the mathematical concept of relation based on the idea of sets proposed by Dr. E.F. Codd of IBM Research in 1970
- **Relationships**, defined as the associations or interactions between entities
- **The degree of relation** refers to number of columns/attributes in the table/relation
- Properties of a relation/table:
  - Should have a distinct relation name (or table name)
  - Each value attribute should contain exactly one value
  - The attribute names should be distinct names
  - Each tuple is distinct (no duplicate tuples)



# Example of Relation



**Figure 5.1**

The attributes and tuples of a relation STUDENT.

Values inside of the table, called **domains**  $D$

# 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 (or table) is a 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 tuple (row) as it has 4 values
  - A tuple (row) in the CUSTOMER relation (table).
- A relation (table) is a **set** of such tuples (rows)

# Formal Definitions - Domain

- A **domain** is a set of acceptable values that an attribute is allowed to contain

Examples:

- “Swedish\_phone\_numbers” are the set of 11 digit phone numbers including country code.
  - Swedish Personal Number. The set of valid 12 digits personal numbers
  - Grade\_points. Possible values for grade A (95-100 points), for grade B (86-94), etc.
  - Marital Status has a set of possible values: {Married, Single, Divorced..}
- 
- A domain also has a *data-type* or a *format* defined for it.
    - The Swedish\_phone\_numbers may have a format: “(+dd)dd-ddd dd dd” 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.
  - It is possible for several attributes to have the same domain:
    - Example: Invoice\_Date, Order\_Date, Shipment\_Date can have the same domain type (Date) and the same format (yy-mm-dd)

# NULL values

- A **NULL** is a special symbol, independent of data type, which means either unknown or inapplicable. It does not mean zero or blank.
- Can represent:
  - An unknown attribute value
  - A known, but missing, attribute value
  - A not applicable value
- Issues with NULL values:
  - Can create problems when functions such as COUNT, AVERAGE, SUM are used
  - Can create logical problems when relational tables are linked/connected

NOTE: The result of a comparison operation is null when either argument is null. The result of an arithmetic operation is null when either argument is null (except functions that ignore nulls).

# 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**.
    - The ordering of values in a tuple in relation schema is important
    - 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:  $t = \{ \langle \text{name}, \text{"John"} \rangle, \langle \text{SSN}, 123456789 \rangle \}$
- **Values** in a tuple:
  - All values are considered **atomic** (indivisible) or **Simple** type. Composite and multi-valued attributes are not allowed in relation model. Multivalued attributes must be presented by separate relations ( and/or weak entities), and composite attributes are represented only by their simple component attributes.
  - 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.
  - During database design, it is best to avoid NULL values as much as possible

# 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.

*For example:* relational model does not allow a list as a value for any attribute, the table names are unique, no duplicated tuples in a table, the primary key can not be NULL.

2. **Schema-based or Explicit Constraints:** They are expressed in the schema by using the facilities provided by the model.

*For example:* max. cardinality ratio constraint in the ER model, key constraints, constraints on NULLs

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.

# 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:
  - **Domain** constraints
  - **Key** constraints
  - **Entity integrity** constraints
  - **Referential integrity** constraints



# Domain Constraints

- Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)
- Domain constraints specify that within each tuple, the value of each attribute  $A$  must be an atomic value from the domain  $dom(A)$ :
  - Standard numeric data types for **integers** (short integer, integer, long integer)
  - And **real** numbers (float and double-precision float)
  - Characters, Booleans, fixed-length strings, date, time, timestamp, enumerated data type, a subrange of values, etc.

# Key Constraints (1)

- **Superkey** of R:
  - Is a set of attributes SK of R with the following condition:
    - No two tuples in any valid relation state  $r(R)$  will have the same value for SK
    - That is, for any distinct tuples  $t1$  and  $t2$  in  $r(R)$ ,  $t1[SK] \neq t2[SK]$
    - This condition must hold in *any valid state*  $r(R)$
- **Key** of R:
  - A "minimal" superkey 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)
- Any superkey formed from a single attribute is also a key. A key with multiple attributes must require *all* its attributes together to have the uniqueness property.

# Key Constraints (2)

- **Example:** Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - CAR has two keys:
    - Key1 = {SerialNo, Reg#}
    - Key2 = {SerialNo}.
  - Both are also superkeys of CAR
  - {SerialNo, Reg#} 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 (3)

- If a relation has several superkeys they called a **candidate keys**, one is chosen arbitrarily to be the **primary key**.
  - The primary key attributes are underlined.
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to *uniquely identify* each tuple in a relation
  - Provides the tuple identity
- Also used to *reference* the tuple from another tuple
  - General rule: Choose as primary key the smallest of the candidate keys (in terms of size of attributes)
  - Not always applicable – choice is sometimes subjective

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

**CAR**

<u>License_number</u>	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

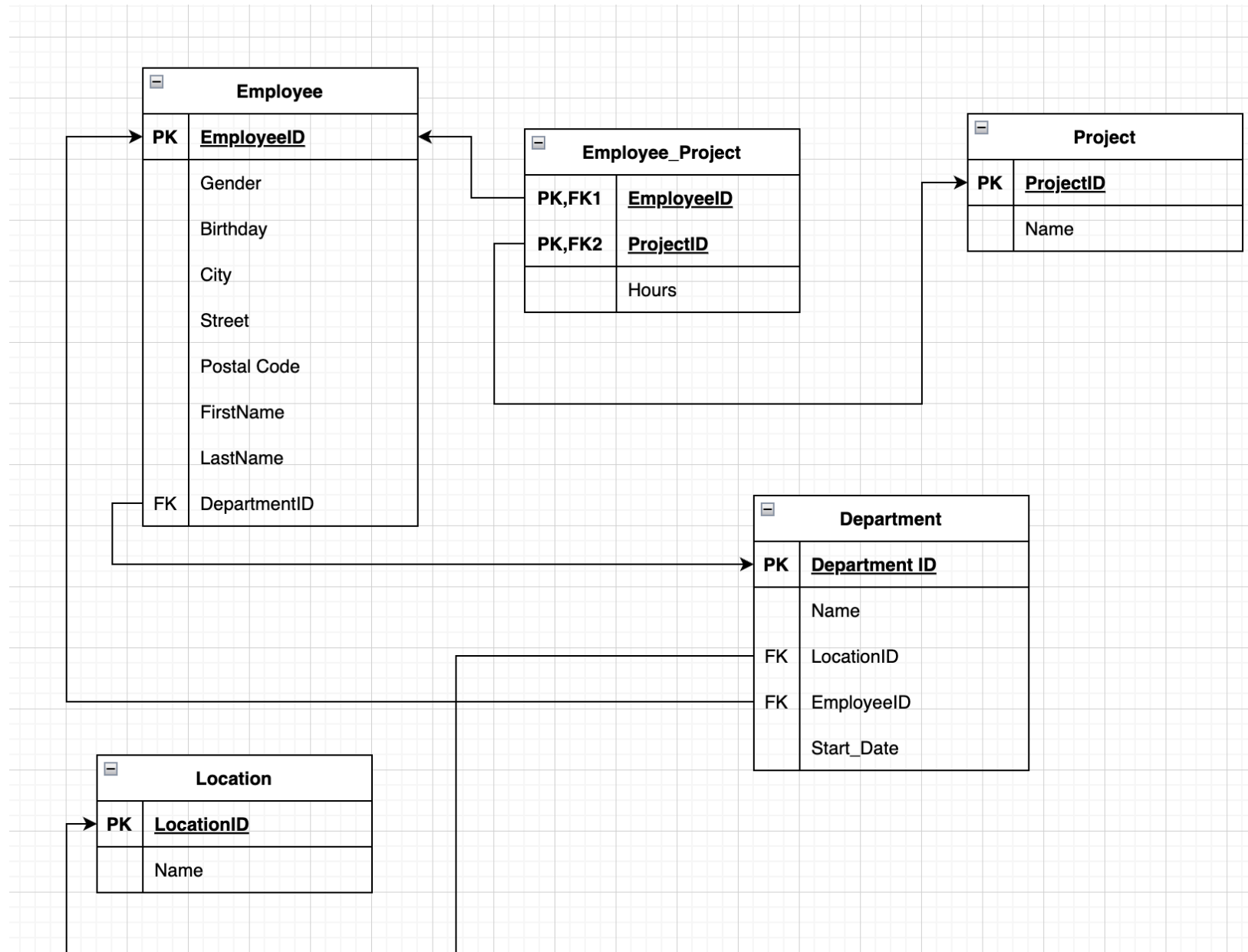
**Figure 5.4**

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

# Referential Integrity

- The referential integrity is specified between the two relations (tables) is used to maintain the consistency among tuples in the two relations (tables).
- A set of attributes FK in relation schema R1 is a *foreign key* of R1 that references relation R2 if it satisfies the following rules:
  - The attributes in FK have the same domain(s) as the primary key attributes PK of R2;
  - A value of FK in a tuple t1 can occur as a value of PK for some tuple t2 or is NULL.

# PK and FK



# 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.
- **Note**: whenever we apply the above operations to the relational table, the constraints should not get violated



# Update Operations on Relations (Tables)

- 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 (e.g., we insert value 15 for employee's age attribute which should not be less than 16)
  - 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

# Examples INSERT operation violated

- Operation 1:
  - Insert <'Cecilia', 'F', 'Kolonsky', NULL, '2000-04-05', 'Linnea gatan 1, 355 63 Växjö'>
  - *Result*: this insertion violates the entity integrity constrain (NULL for the personal number), so it is rejected.
- Operation 2:
  - Insert <"Alicia", 'J', 'Zelaya', '123456789', 'Linnea gatan 1, 355 63 Växjö'>
  - *Result*: This insertion violates the key constrain because another tuple (row) with the same personal number value already exists in the EMPLOYEE table, so it is rejected
- Operation 3
  - Insert <'Cecilia', 'F', 'Kolonsky', NULL, '2000-04-05', 'Linnea gatan 1, 355 63 Växjö', 7>
  - *Result*: This insertion violates the referential integrity constrain specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7. So it is rejected
- Operation 4:
  - Insert <'Cecilia', 'F', 'Kolonsky', '20000334-2389', '2000-04-05', 'Linnea gatan 1, 355 63 Växjö'>
  - *Result*: This insertion violates the domain constrain on personal number because the birth day 34 can not be specified in personal number format "yyyymmdd-dddd"

# Company Database Example

**Figure 5.6**

One possible database state for the COMPANY relational database schema.

**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>	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	<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

# 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 6 for more details)
      - RESTRICT option: reject the deletion
      - CASCADE option: propagate the new primary key value into the foreign keys of the referencing tuples
      - SET NULL option: set the foreign keys of the referencing tuples to NULL
  - One of the above options must be specified during database design for each foreign key constraint

# Examples DELETE Violations

- Operation 1:
  - Delete the WORKS\_ON tuple with Esnn = '999887777' and Pno=10
  - *Result*: This deletion is acceptable and deletes exactly one tuple
- Operation 2:
  - 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. **Solution**: use CASCADE option
- Operation 3:
  - 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. **Solution**: use CASCADE option

# 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. [Solution](#): use CASCADE option on UPDATE
  - Updating an ordinary attribute (neither PK nor FK):
    - Can only violate domain constraints.

# Summary

- Presented Relational Model Concepts
  - Definitions
  - Characteristics of relations
- Discussed Relational Model Constraints and Relational Database Schemas
  - Domain constraints
  - Key constraints
  - Entity integrity
  - Referential integrity
- Described the Relational Update Operations and Dealing with Constraint Violations



# Additional Video Tutorials

- Relation Data Model (<https://www.youtube.com/watch?v=-CuY5ADwn24> )
- Relational Database Concepts (<https://www.youtube.com/watch?v=NvrpuBAMddw> )

# Lecture 4 Key Terms

- A **tuple** (or row) is an ordered set of values in relation (table)
- **Constraints** determine which values are permissible and which are not in the database. There are three main types: (1) Inherent or Implicit Constraints, (2) schema-based, and (3) application-based
- **NULL value**: a special value, independent of data type, which means either unknown or inapplicable; it does not mean zero or blank
- A **domain** is a set of acceptable values that an attribute is allowed to contain
- **integrity constraints**: logical statements that state what data values are or are not allowed and which format is suitable for an attribute
- **referential integrity**: requires that a foreign key must have a matching primary key or it must be null
- **Inherent or Implicit Constraints**: constraints that are inherent in the definition/assumptions of a particular data model hold in every database having that data model as its underpinning. Example: In the relational model, no two tuples in a relation can be duplicates.
- **Schema-based constraints**: constraints that can be directly expressed in the schemas of the data model. They are domain constraints, key constraints, constraints on NULL, integrity and referential integrity constraints
- **Application-based constraints**: Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs

# Lecture 4 Key Terms

- A **Superkey** is a single key or a group of multiple keys that can uniquely identify tuples in a table. Super keys can contain redundant attributes that might not be important for identifying tuples
- A **key**: an attribute or group of attributes whose values can be used to uniquely identify an individual entity in an entity set
- A **composite key**: composed of two or more attributes, but it must be minimal
- A **foreign key (FK)**: an attribute in a table that references the primary key in another table OR it can be null
- A **primary key** is a minimal set of attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation
- Possible violations for UPDATE operation: key constrains, domain constrains
- Possible violations for INSERT operation: domain, key, integrity, and referential integrity
- Possible violations for DELETE operation: key constrains