

Database Design and SQL

Chapter 2: Data Modeling Using ERD Diagram

FSDM-CPCM-2023S

Sagara Samarawickrama

Chapter Objectives

- Explain what entity relationship diagrams(ERD) are
- Identify and explain the steps for developing and ERD
- Develop ERD Diagrams.

Introduction to Data Modeling

The first step in data modelling is the development of a conceptual data modeling is the development of a conceptual data using entity relationship diagram. The second step is logical data modelling, which includes a process called normalization to evaluate the entity relationship diagram for alternative designs.

Data modelling consists of two steps: *conceptual* and *logical data modelling*.

A data model represents the data used in an organization and the relationship between the data. A data model specifies the following items of interest to the organization:

Introduction to Data Modeling

- **Entities** (people, places, things or concepts about which data must be recorded, such as customers or employees)
- **Events** (such as placing an order, approve a loan)
- **Attributes** for entities , events, and relationships (such as customer's names, dates on which order was placed)
- **Relationships** among entities and events(for example , a customer places an order , a loan officer approves a loan)
- **Business rules** or policies (for example, an employees' hours rate must be between 11.75 and 28.00)

Introduction to Data Modeling

Data model serves as a contract between end users and the designers. As with any contract, both parties should have no question about its meaning. The data model , as a contract , specifies the minimum requirements for the database design.

Data modelling is the process of developing a graphical representation of the proposed database using an entity relationship diagram. An ERD is a graphical representation of the proposed database showing the relation between entities and attributes or characteristics of the entities

Introduction to Data Modeling

There are four goals of ER diagrams:

- Capture all required data
- Ensure data appears only once in the database design
- Do not include any data that is derived from other data in the data model
- Arrange data in the data model in a logical manner

Entities in the ERD **represents a table** in the relational database. The entities are connected with **lines** that represent the **relationship** between the two entities.

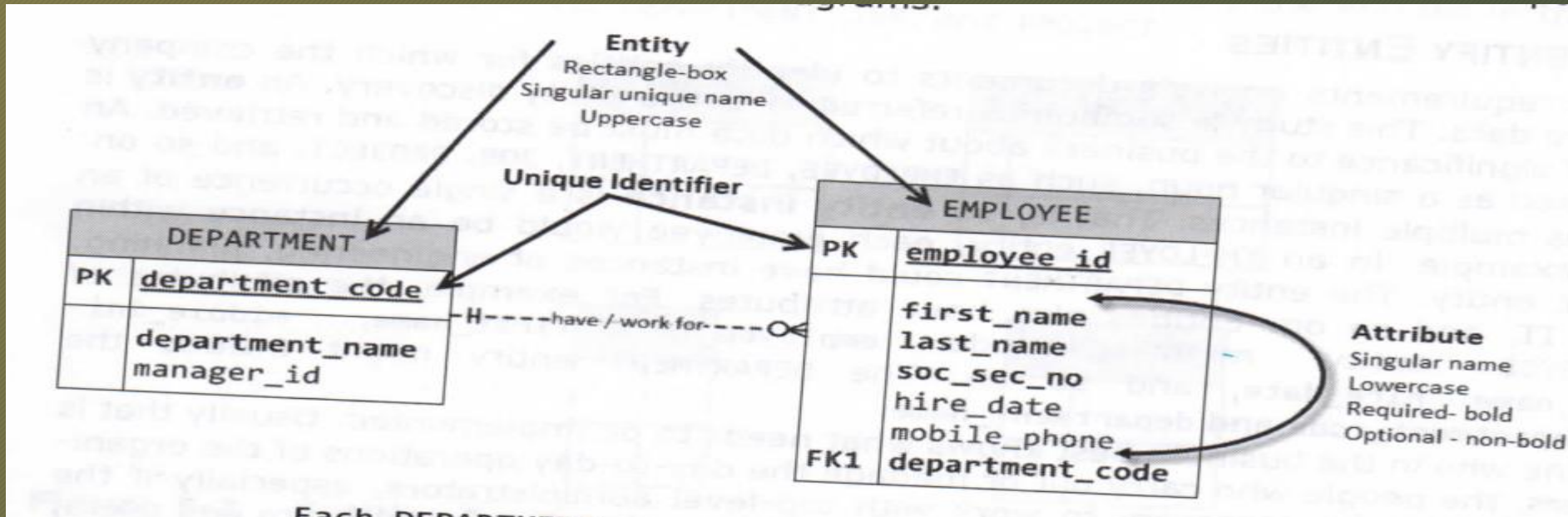
Introduction to Data Modeling

The steps involved in data modelling are shown in the following figure

Step	Task	Description
1	Identify entities	Identify the people, places or events
2	Identify attributes	Define attributes with description
3	Specify unique identifiers(UID)	Identify attributes that uniquely identify one occurrence of each entity
4	Establish Relationships	Identify relationships between entities
5	Define relationships Optionality and Cardinality	Determine the number of occurrence of a one entity for a single occurrence of the related entity
6	Eliminate Many to Many Relationships	With an associate entity that includes a unique identifier (UID) and foreign keys in each parent entity
7	Name Relationships	Name each relationship between entities
8	Normalize the database	is the process of organizing the columns (attributes) and tables (relations) of a relational database to reduce data redundancy.
9	Define Custom Data types	Define the data types and sizes of each column in the physical database.

Introduction to Data Modeling

Data model represents the entities, attributes and relationships of the proposed database. It is the model from which the physical design is created. Following is a one example of a ER diagram.



Developing an Entity Relationship Diagram

Step 1 : Identify Entities:

The first step in data modelling is to identify entities. An entity is a real world items for each data are collect and stored.

- An entity is something of significance to the business about which data must be stored and retrieved.
- An entity is named as a singular noun, such as EMPLOYEE,DEPARTMENT,JOB and so on
- An Entity has multiple instances (entity instance is a single occurrence of an entity e.g. EMPLOYEE)
- Entities have attributes.(e.g. employee_id,first_name,hire_date)

Determine who is in the business best knows what needs to be implemented. Usually that is end users. It might also be necessary to work with top level administrators. While speaking to the end users collect manuals,documents,sample reports or anything else that will help to document the data the organization uses

Developing an Entity Relationship Diagram

An **object**, such as customer , and an **event** , such as placing an order, are two types of entities or things about which facts should be recorded; **these objects and events can be represented as entities.**

When entity is discovered , be sure to enter it in the data dictionary. Here are some of the things to record about an entity.

- ✓ The official name for the entity as a singular noun, as used in the rest of the data model(e.g. EMPLOYEE)
- ✓ Synonyms (e.g. WORKER)
- ✓ A Short Description of the entity(e.g. An individual who works part-time or full-time under a contract of employment)
- ✓ Approximate or estimated yearly volumes(e.g. “Approximately 100 employees are hired each year)

Developing an Entity Relationship Diagram

Step 2 : Identify Attributes:

An attribute is a characteristic that identifies or describes an entity. Attributes are represented as columns in a database table. Attributes are single-valued. That is, each attribute can have only one value for each instance of the entity. For example first_name, last_name and hire_date are attribute of the EMPLOYEE entity. Moreover, each attributes, such as first_name and last_name, can have only one value for each instance.

Attribute names are listed inside an entity's rectangle

e.g. employee_id, first_name, last_name, birth_date, gender, soc_sec_no, hire_date and salary

EMPLOYEE	
	employee_id
	first_name
	last_name
	birth_date
	gender
	soc_sec_no
	hire_date
	salary

Developing an Entity Relationship Diagram

Volatile Attributes:

Attributes that change over time are called **volatile attributes**. For example, the attribute age would be a volatile attribute in the entity EMPLOYEE if the entity also contain the attribute *birth_date*. Another example is years_of_service

Required and Optional Attributes:

Each instance of an entity contains attributes that can hold a potential value. An attribute that must contain a value for each instance of the entity is called a **required or mandatory attribute**. For example entity would contain customer name attribute that would be a required attribute. If an attribute is identified as required, it will appear as bold in the entity rectangle. On the other hand , an attribute that may not have a value or is null is called an optional attribute. For example in a CUSTOMER entity , the phone number and e-mail address attributes may not be required to have a value. In this situation ,these attributes are considered to be optional attributes.

Developing an Entity Relationship Diagram

Identify the required and optional attributes of the EMPLOYEE entity !

Domains:

Attributes have domain. A domain defines **two aspects of an attribute**: Its **allowable values** and the **allowable operations** on the attribute. For Example , with clear definitions of the domains of hourly_wage and monthly_salary, one can assure that neither column ever contains negative values and that hourly _wage is not added to monthly_salary without conversion.

EMPLOYEE	
PK	<u>employee_id</u>
	first_name last_name soc_sec_no hire_date

Time Dependent Attributes:

Identify which attributes are time-dependent. For example , if a part's cost must be known different dates , cost is a time-dependent attribute.

Developing an Entity Relationship Diagram

Default Values:

Identify, for each attribute, whether there is a default value that should be used if no explicit value is supplied when a new instance is added.

Documenting Attributes:

- Official Name of the attribute (“annual_salary”)
- A short textual description (“The annual gross pay before taxes”)
- The source in the business process (“defined in the Employment Contract”)
- The allowable values (salary > 11.75)
- Whether a value is required or optional
- The default value
- Attribute is part of a unique key
- Whether the attribute is direct or derived attribute.

Developing an Entity Relationship Diagram

DEPARTMENT	
	department_id
	department_name

EMPLOYEE	
	employee_id
	first_name
	last_name
	birth_date
	gender
	soc_sec_no
	hire_date
	salary

JOB	
	job_id
	job_title
	minimum_salary
	maximum_salary

Each attribute must be matched with exactly one entity. Often , it seems as if an attribute should go with more than one entity(e.g. name). In that case, either add a modifier to the attribute name to make it unique - such as department_name, first_name and so on.

Developing an Entity Relationship Diagram

Step 3 :Define Unique Identifier(UID)

Every entity must have a unique identifier. A unique identifier(UID) is a single attribute or a combination of attributes that uniquely identifies one and only one instance of an entity. That is , the UID must be unique for each instance of the entity. When two or more attributes are used as the unique identifier , this is called the **concatenated key**.

Candidate Keys

During the identification of the entities, it might be determined that there are several attributes that could qualify as a unique identifier. For example , in an EMPLOYEE entity , the employee batch number, Social security number e-mail address , the telephone number attributes could be used to uniquely identify an EMPLOYEE instance. When this occurs , all of the attributes that can serve as the unique identifier are known as the **candidate key**. For these situations one candidate key should be selected as the unique key which is called the **primary key**

Developing an Entity Relationship Diagram

The desirable primary key characteristics are as follows:

- 1.Unique values:** The PK must uniquely identify each entity instance. A primary key must be able to guarantee unique values. It cannot contain nulls.
- 2.Non intelligent:** The PK should not have embedded semantic meaning other than to uniquely identify each entity instance
- 3.No change over time:** If an attribute has semantic meaning, it might be subject to updates. This is why names do not make good primary keys
- 4.Preferably single-attribute:** A primary key should have the minimum number of attributes possible (irreducible)
- 5.Preferably Numeric:** Unique values can be better managed when they are numeric, because the database can use internal routines to implement a counter-style attribute that automatically increments values with the addition of each new row
- 6.Security-Compliant:** The selected primary key must not be composed of any attribute(s) that might be considered a security risk or violation. For example, using a Social Security number as a PK in an EMPLOYEE table is not a good idea.

Developing an Entity Relationship Diagram

Types of UIDs (Primary Keys)

An entity's unique identifier (primary key) can be a **natural key** or a **surrogate key**.

Natural Key:

A natural key is a primary key that can be used to uniquely identify each row in the database table but also have some meaning to the users e.g. Customer Number

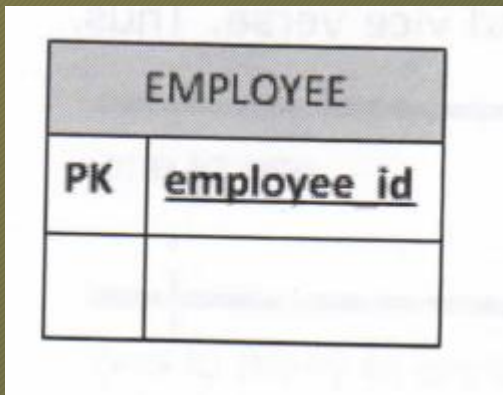
Surrogate Key :

A surrogate key is an arbitrarily assigned positive integer number that is system generated, never NULL, and is non-identifying or meaningless. A non-identifying value is a value that has no impotence to users

System generated surrogate keys avoid problems that can sometimes occur with natural identifiers as the descriptive meaning could become obsolete in the future.(Ex :Wholesale company uses Vendors product number but later vendor decides to change it)

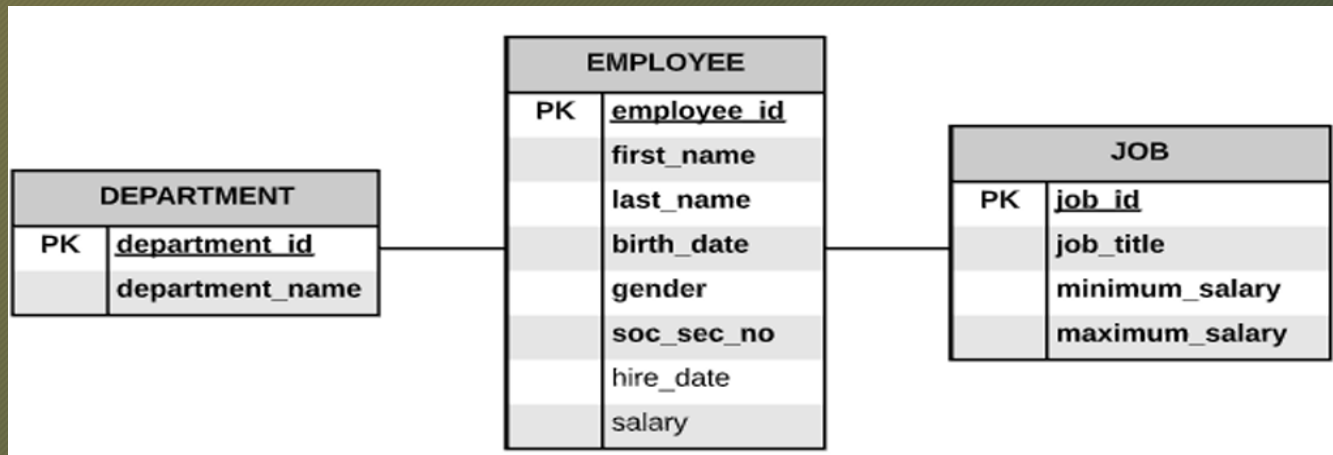
Developing an Entity Relationship Diagram

A unique identifier is one of the candidate keys that has been identified to uniquely identify each instance of an entity. The unique identifier for the shown EMPLOYEE entity is employee_id. The unique identifier is always the first attribute to be listed. In addition , the unique identifier is **underlined by the letters PK(primary key)**. Unique identifiers become primary keys in the physical database. Notice that employee_id is bold, indicating that it is a required attribute.



Developing an Entity Relationship Diagram

Step 4 : Identify Relationships: A relationship is like a verb that identifies a dependency or natural association between two entities; for example, a student takes courses, a course has students, a room is scheduled for classes, and a professor teaches courses. The nature of relationships in the ERD reveals a lot about the structure of an organizations data. A relationship between two entities is represented by a line joining the two entities. Relationship line indicates that each instance of an entity may have relationship with instances of the connected entity, and vice versa. Thus, relationship lines express how entities are mutually related.



Developing an Entity Relationship Diagram

Step 5 : Determine Optionality and Cardinality:


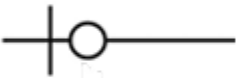

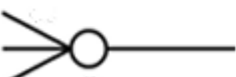
Symbols are placed at the ends of relationship lines to indicate the optionality and the cardinality of each relationship. Optionality expresses whether the relationship is optional or mandatory. For example ,when the department entity may not have any employees, a **minimum of zero employee**, the relationship is **optional**. On the other hand when the department entity **must have one or more employees**, the relationship is **mandatory**.






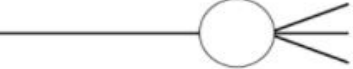
Cardinality defines **the number of occurrences of one entity for a single occurrence of related entity**.

- Optionality is the minimum number of times an instance in one entity can be associated with an instance in the related entity.
- Cardinality is the maximum number of times an instance in one entity can be associated with instances of related entity.

Developing an Entity Relationship Diagram

Relationships identify an association between two entities and are represented by stylized lines. There are several methods used to represent style lines on a relationship line , but **the crow's foot style** is one of the most popular for ER Diagrams

Symbol	Cardinality
	One and only one
	Zero or one
	One or many
	Zero, one or many

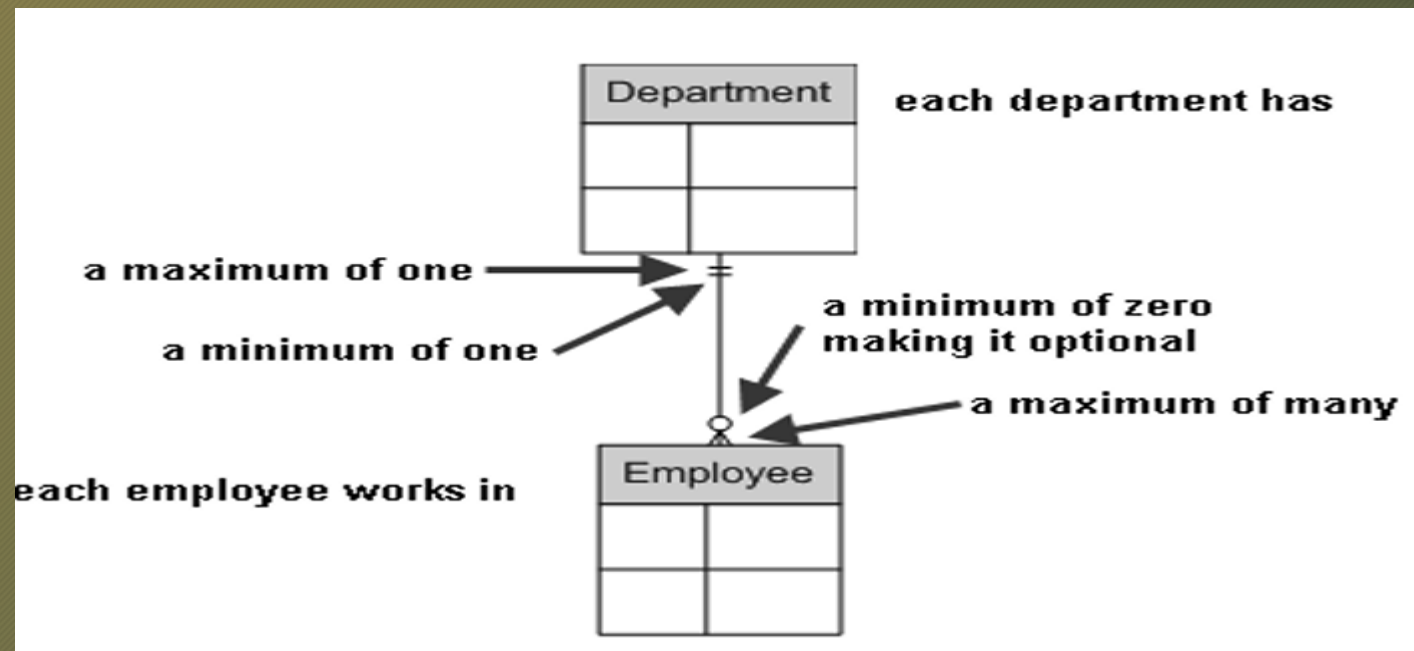
Symbol	Meaning	Number
	One	N/A
	Many	N/A
	Mandatory-One	Exactly one
	Optional-One	Zero or one
	Mandatory-Many	One or More
	Optional-Many	Zero or more

Developing an Entity Relationship Diagram

Following figure indicates all the possible optionality and cardinality combinations. Two symbols are specified at the end of the relationship line where it joints the entity. The first , or outer symbol is the optionality indicator. A circle (o) indicates that the relationship is optional. The second or inner symbol indicates cardinality. A stroke (|) indicates that the maximum number of relationship is one. A crow foot indicates that many relationships can exist between instances of the related entities

ERD notation	Meaning
	Each instance of TableA is related to a minimum of zero and a maximum of one instance of TableB .
	Each instance of TableB is related to a minimum of one and a maximum of one instance of TableA .
	Each instance of TableB is related to a minimum of one and a maximum of one instance of TableA .

Developing an Entity Relationship Diagram

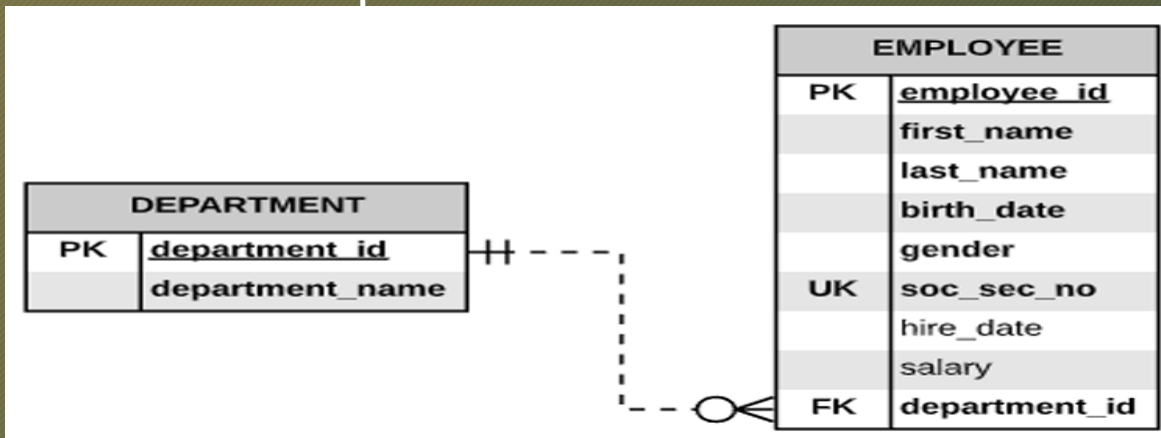


The department might not have any employees(a minimum of zero) and may have a maximum of one or many employees. In addition , each employee works in exactly one department. That is , an employee may work in a minimum of one department and a maximum of one department.

Developing an Entity Relationship Diagram

Foreign Keys :

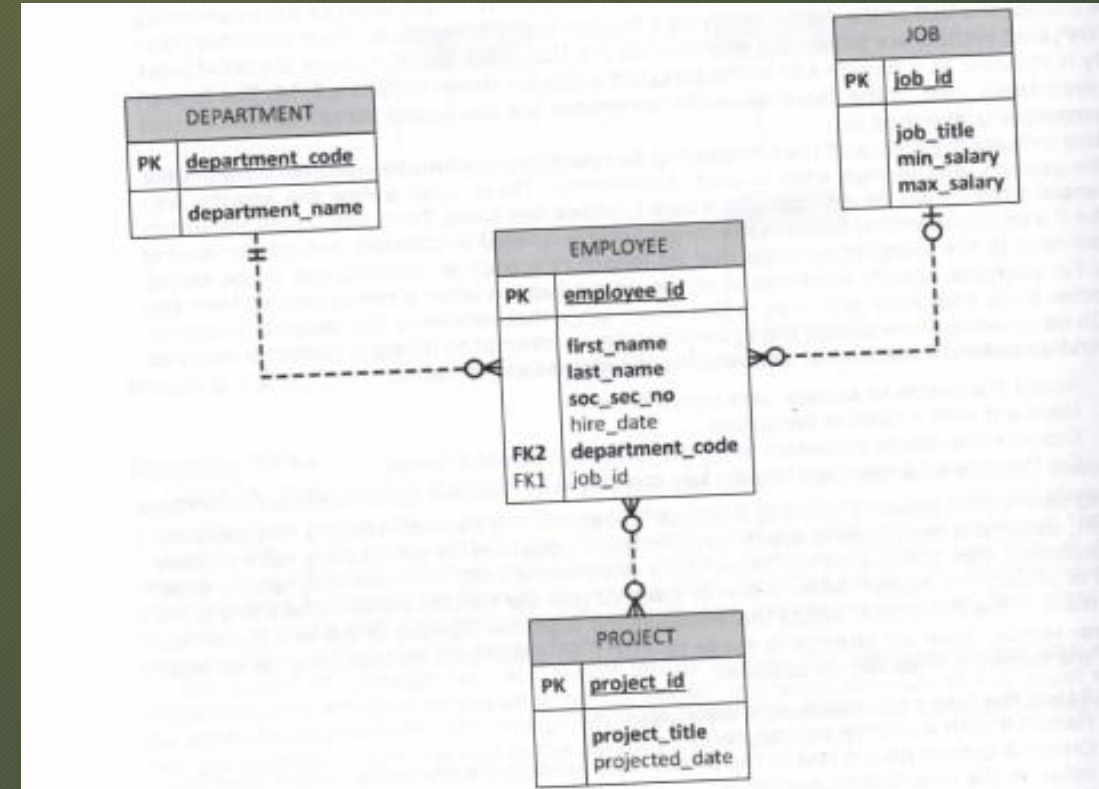
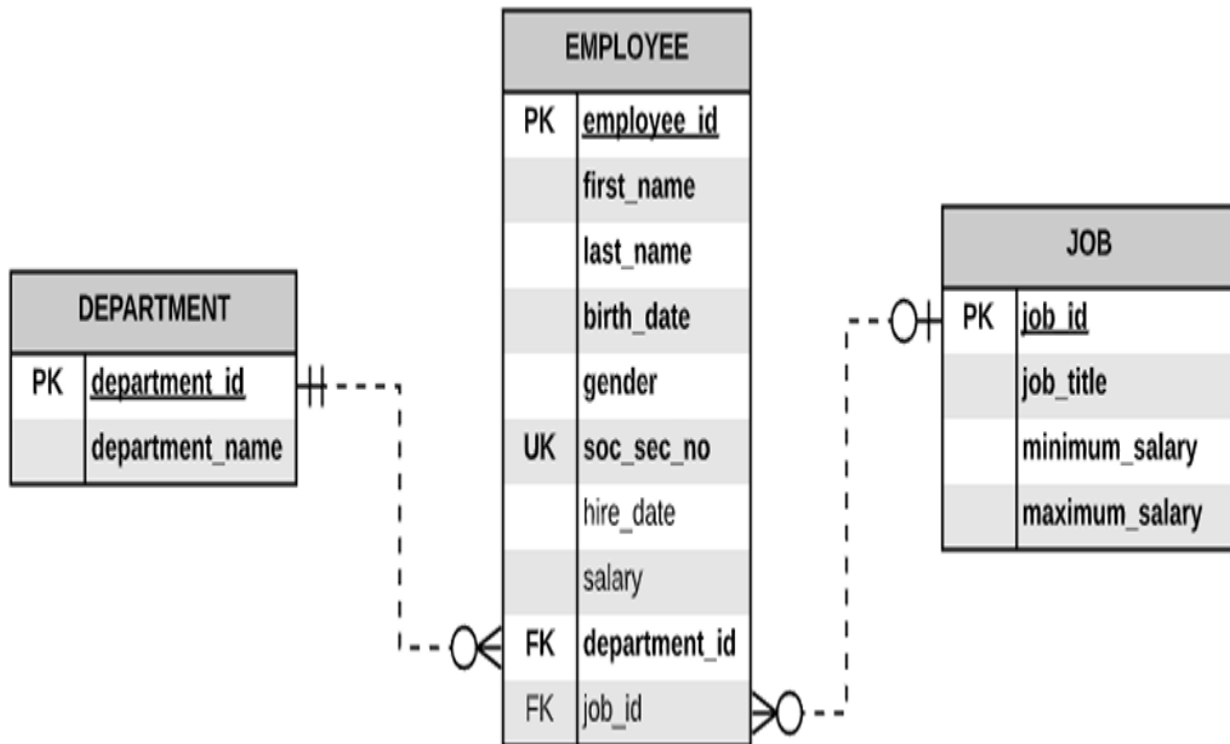
When a relationship exists between two entities, the unique identifier of one entry becomes a foreign key in the other entity. The purpose of the foreign key is to ensure referential integrity of the data by permitting only those values that are supposed to appear in the database. A relationship between two entities implies that one of them has one or more attributes that represent the foreign key values from other entity. The letter **FK(foreign key)** are placed beside foreign key attributes to indicate which relationship the attribute serves as a foreign for and which attribute in the target entity the attribute is paired with.



Each department may have zero ,one or more employees
Each employee must work for one and only one department.

Developing an Entity Relationship Diagram

Exercise : Determine the optionality, cardinality and foreign keys for the following ERD's



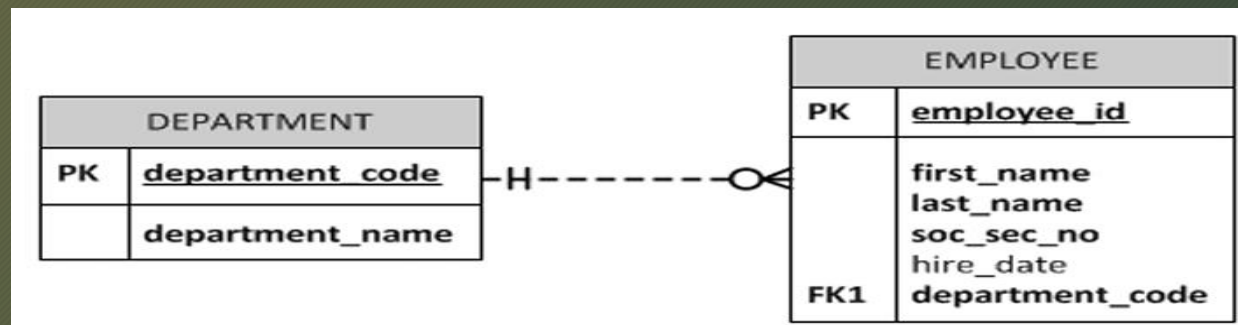
Developing an Entity Relationship Diagram

Relationship Strength :

Any time there is a relationship between entities, the unique identifier (primary key) of the parent entity appears as a foreign key attribute in the child entity. How that foreign key attribute is implemented into the child entity determines the strength of the relationship between the entities. There are two ways to determine that.

Weak or Non-identifying relationship:

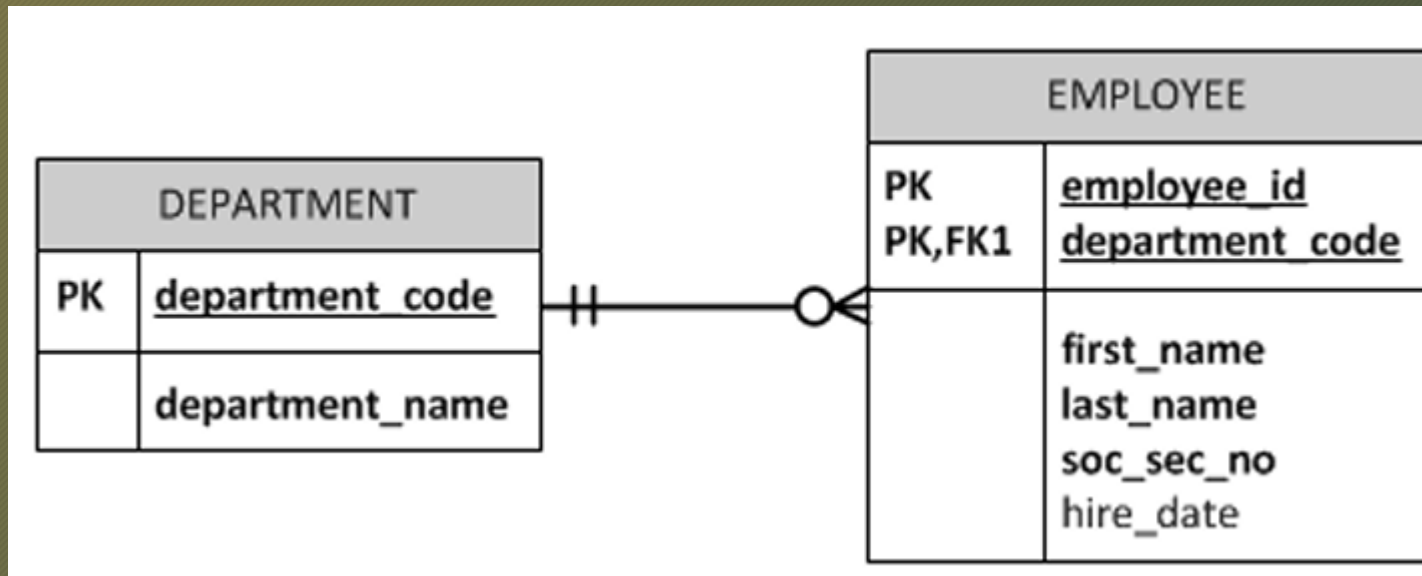
A weak or non-identifying relationship exists between two Entities when the unique key (PK) of the parent entity is **not a component** of the unique identifier of the child entity. The crow's foot notation represents a weak relationship **as a dashed relationship line** between the two entities.



Developing an Entity Relationship Diagram

Strong or Identifying relationship:

When a **strong** or identifying relationship exists between two entities , the unique identifier (primary key) of the parent entity **becomes a component** of the unique identifier of the child entity. The crow foot notation represents an strong relationship as a **solid line between two entities**

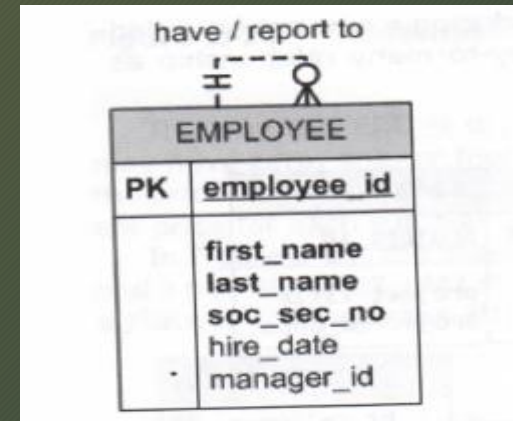
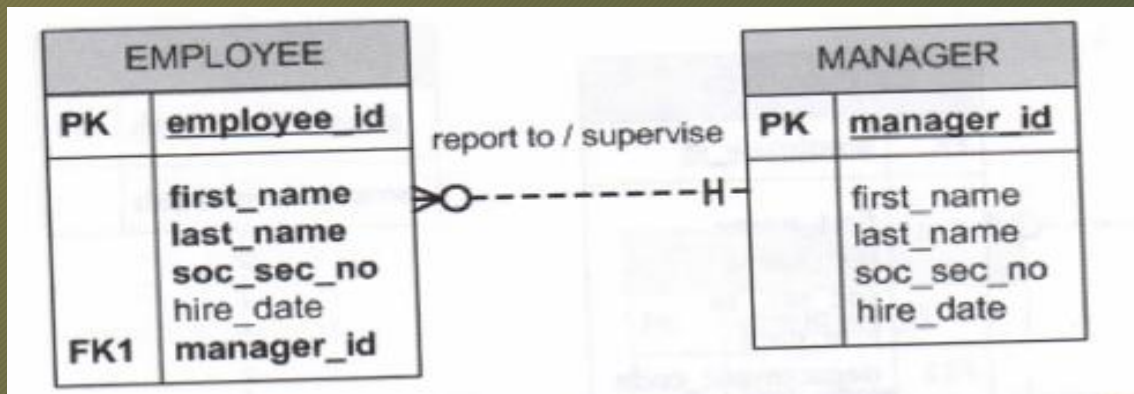


Developing an Entity Relationship Diagram

Recursive Relationship :

A recursive relationship occurs when an entry has a relationship with itself. Lets consider a situation where , as in many companies, each employee ,except the president , is supervised by one manager. This can be represented by the ERD in figure below.

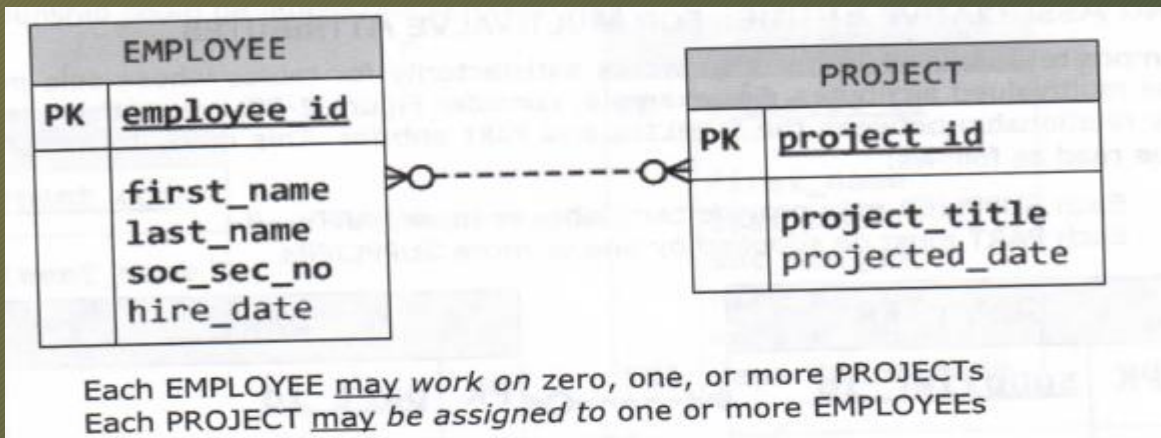
The problem in the ERD is that each manager is also an employee. Creating the second MANAGER Entity not only duplicates data from the EMPLOYEE Entity, but it virtually guarantees that there will be Mistakes and conflicts in the data. This problem can Be fixed by eliminating the redundant entity and re-drawing the association line.



Developing an Entity Relationship Diagram

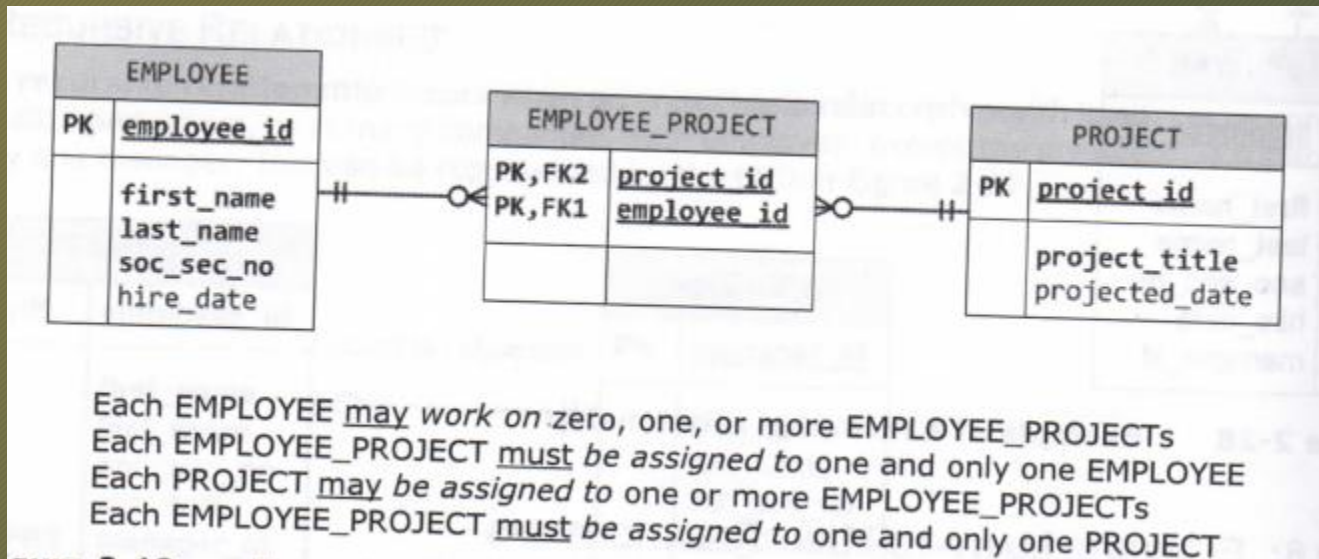
Step 6 : Eliminate Many to Many Relationships

Because of their simplicity, unique identifiers consists of just one attribute are preferred to use UID's with more than attribute. However there are situations in which a composite unique identifier works satisfactorily in place of a single-column unique identifier. Following figure shows a many-to-many relationship between EMPLOYEE and PROJECT. Many-to-Many relationships add complexity and confusion to the model and to the application development process. Such relationships will be a problem later when the related entities are implemented into database tables because two entities cannot be children of each other. There is no place to put the foreign keys



Developing an Entity Relationship Diagram

The many to many relationship problem can be solved by introducing a new entity called an associative, junction or intersection entity, for each many - to -many relationships as shown in the following figure. The name of the new intersection entity will be the hyphenation of the names of the two originating entities. In this example , the PROJECT_EMPLOYEE entity becomes the intersection entity. It will have a concatenated UID consisting of the UIDs from the PROJECT and EMPLOYEE entities.

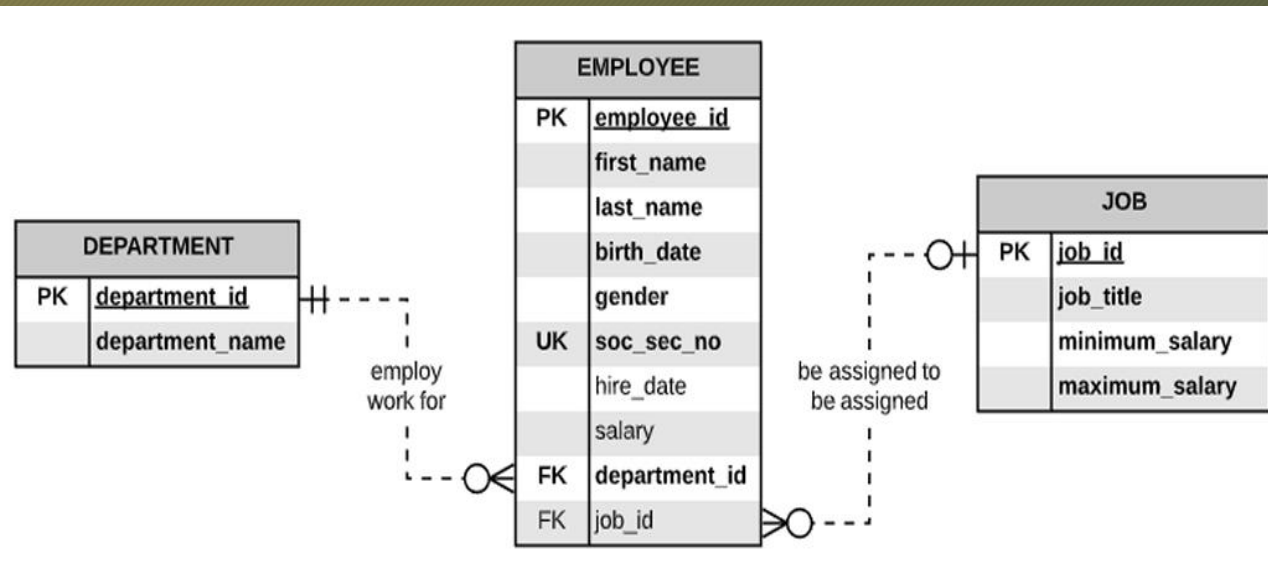


The EMPLOYEE_PROJECT entity will have a one-to-many relationship with each of its parent entities as shown in the figure.

Developing an Entity Relationship Diagram

Step 7 : Name Relationships

A relationship can be labeled to define the relationship .In this case, one can infer that a department has employees, or that an employee works for a department. Figure shows how this relationship could be labeled for clarification.



Each DEPARTMENT may employ zero, one or more EMPLOYEEs
Each EMPLOYEE must work for one and only one DEPARTMENT
Each JOB may be assigned to zero, one or more EMPLOYEEs
Each EMPLOYEE may be assigned zero or one JOB

Developing an Entity Relationship Diagram

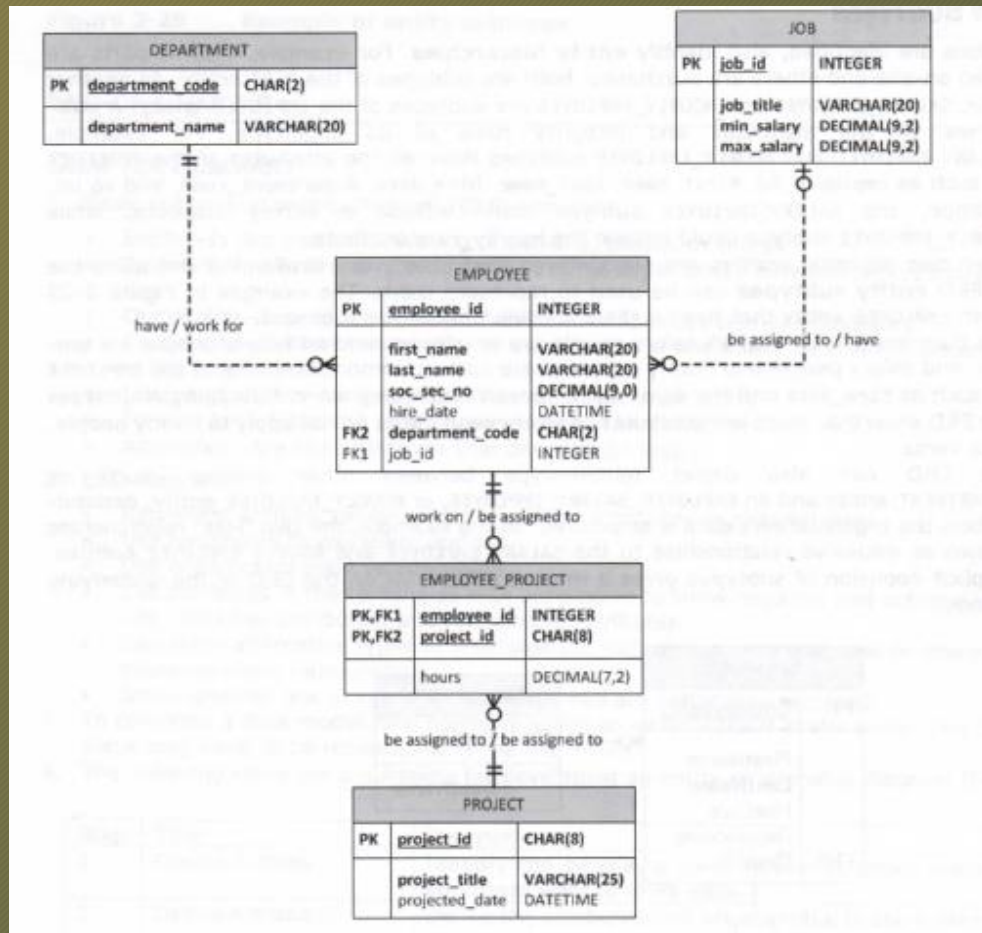
Relationship Matrix

A relationship matrix can be developed where entity relationships are listed down the left column and across the top, as shown in the figure. An active verb can be filled in at the intersection of two entities that are related. Each row and column should have at least one relationship listed, unless the associated with that row or column does not interact with the rest of the system.

	Department	Employee	Job
Department		employ	
Employee	work for		Be assigned
Job		be assigned to	

Once the entity relationships are determined , they are applied to the entity relationship diagram
The result is shown in the next slide

Developing an Entity Relationship Diagram

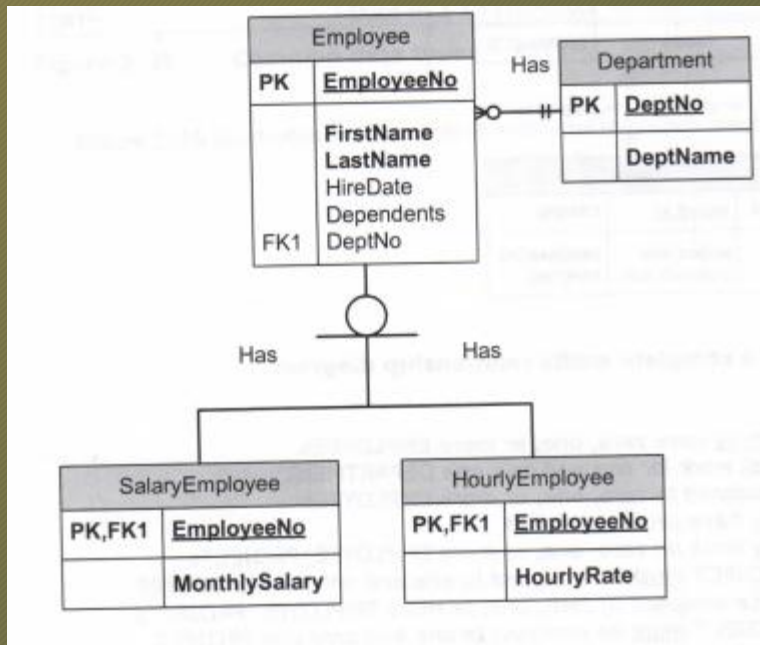


Each DEPARTMENT may have zero, one, or more EMPLOYEEs
Each EMPLOYEE must work for one and only one DEPARTMENT
Each JOB may be assigned to zero, one, or more EMPLOYEEs
Each EMPLOYEE may have zero or one JOB
Each EMPLOYEE may work on zero, one, or more EMPLOYEE_PROJECTs
Each EMPLOYEE_PROJECT must be assigned to one and only one EMPLOYEE
Each PROJECT may be assigned to zero, one, or more EMPLOYEE_PROJECTs
Each EMPLOYEE_PROJECT must be assigned to one and only one PROJECT

Developing an Entity Relationship Diagram

Entity Subtypes:

When two potential entities are encountered that seem very similar but not quite the same, ERD entity sub type can be used to represent them. The example in figure shows an EMPLOYEE entity that has SALARY and HOURLY sub types



The diagram tells us that all salary people are employees, and salary people and hourly people share common attributes of the EMPLOYEE entity, such as hire_date and the department_code in which they work. Subtyping employees lets the ERD show that there are attributes of people that do not apply to hourly people and vice versa.

Conclusion

