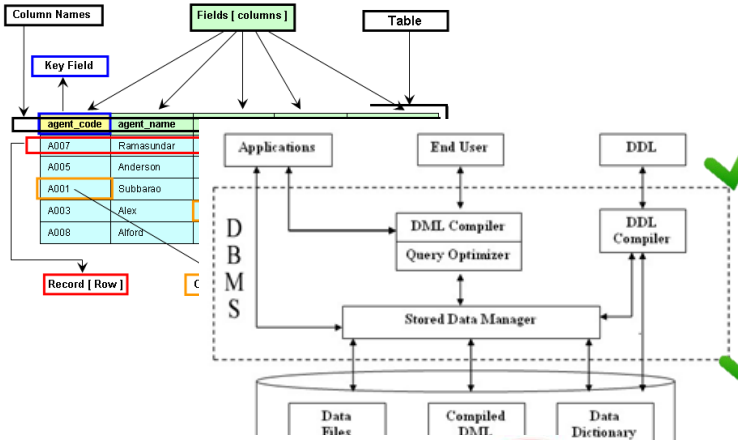
DATABASE CONCEPTS &
ER MODEL

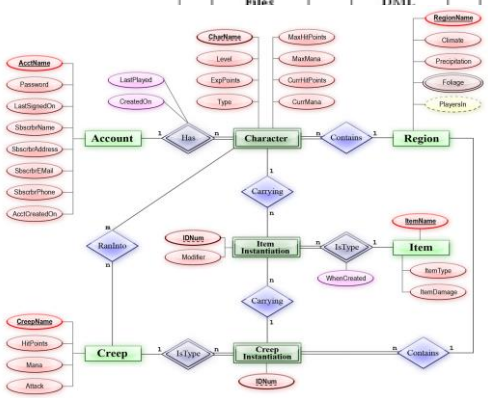
Instructor:

Learning Goals

By the end of this lecture students should be able to:

- Understand an overview of the basic RDBMS Concepts
- Understand an insight into the architecture and components of a Database System.
- Describe how entities, attributes and relationships are used to model data;
- Converting ER Model to relational schema





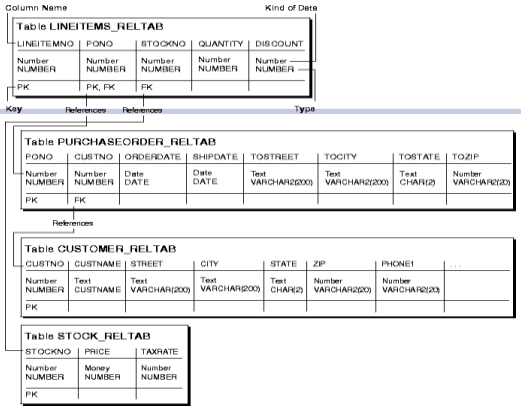


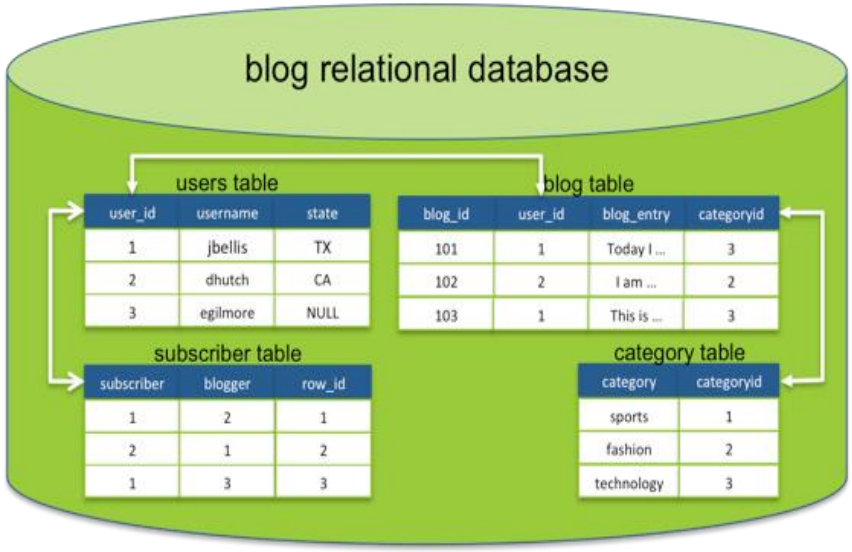
Table of contents

Section 1

DBMS TUTORIAL

What is Database?

- The **database** is a collection of inter-related data which is used to **retrieve**, **insert** and **delete** the data efficiently.
- It is also used to organize the data in the **form of a table**, **schema**, **views**, and **reports**, etc.
- **For example:** The college Database organizes the data about the admin, staff, students and faculty etc.

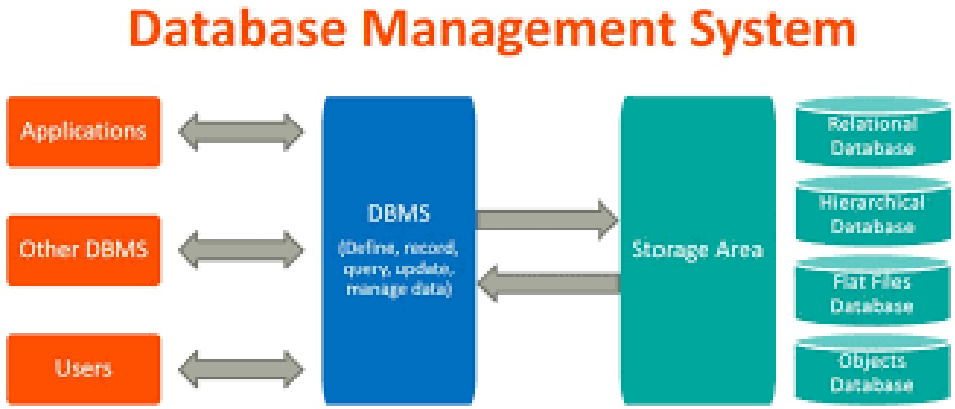


Database Management System

- **Database management system** is a software which is used to manage the database.
- **Example**: MySQL, Oracle, MS SQL Server, etc are a very popular commercial database which is used in different applications.
- DBMS provides an **interface** to perform various operations like database **creation**, **storing data** in it, **updating data**, **creating a table** in the database and a **lot more**.
- It provides **protection** and **security** to the database. In the case of multiple users, it also maintains data consistency.

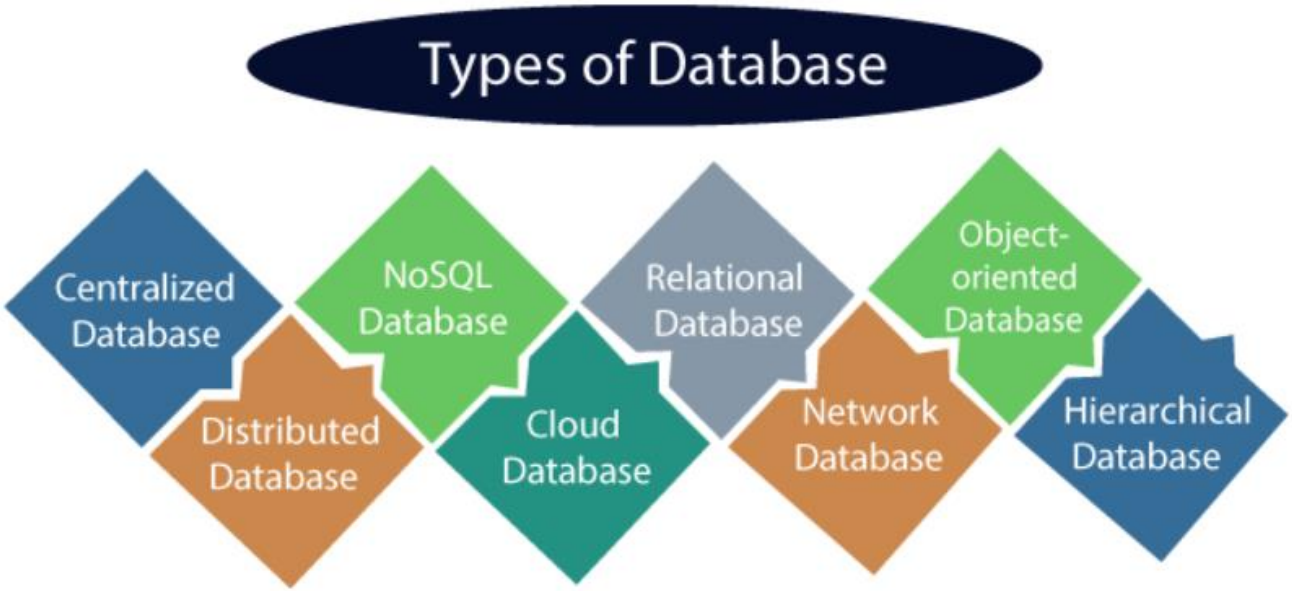
Database Management System

- **DBMS allows users the following tasks:**
  - ✓ **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
  - ✓ **Data Updation:** It is used for the insertion, modification, and deletion of the actual data in the database.
  - ✓ **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.
  - ✓ **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.



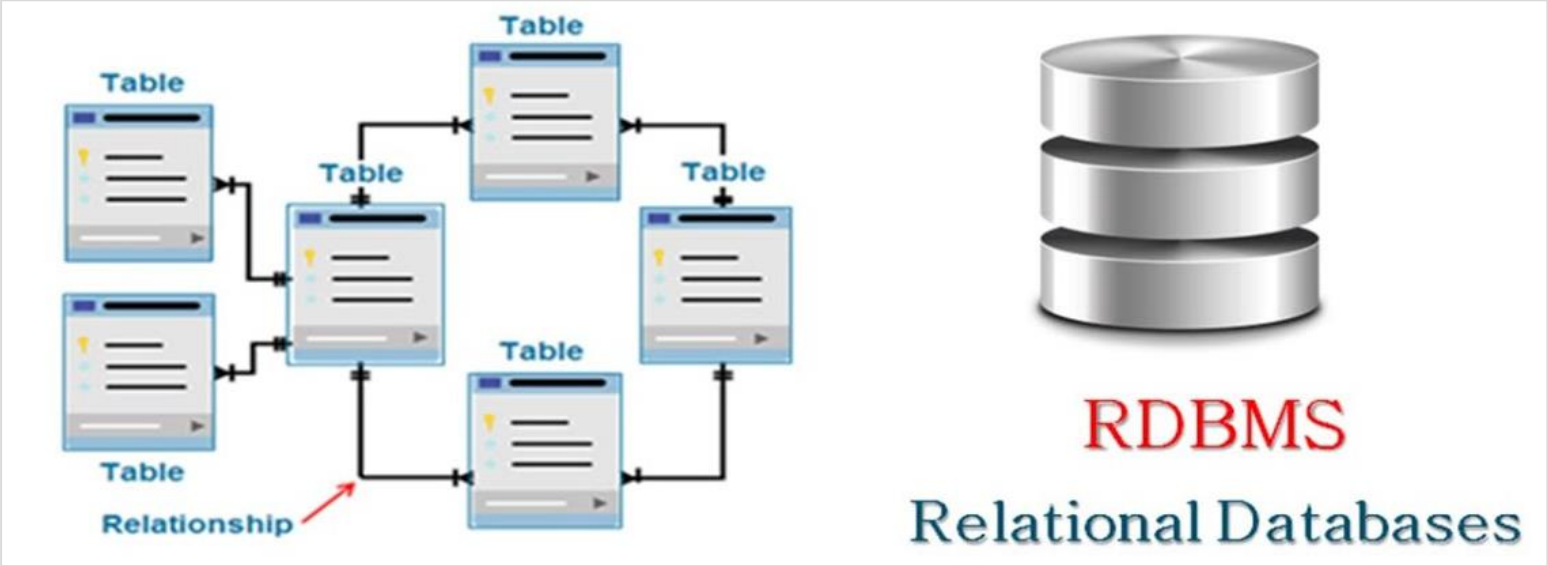
What is Data?

- **Data** is a collection of a distinct small unit of information.
- It can be used in a variety of forms like **text**, **numbers**, **media**, **bytes**, etc. it can be stored in pieces of paper or electronic memory, etc.
- **Types of Databases:**



What is RDBMS?

- **RDBMS** stands for *Relational Database Management Systems.*.
- All modern database management systems like **SQL**, **MS SQL Server**, **IBM DB2**, **ORACLE**, **My-SQL** and **Microsoft Access** are based on RDBMS.
- It is called **Relational Data Base Management System** (RDBMS) because it is based on relational model introduced by E.F. Codd.



What is table?

- The **RDBMS database** uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data.
- **Example:** the "**Customers**" table.

Customer ID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

What is field?

- **Field** is a smaller entity of the table which contains specific information about every record in the table.
- In the above example, the field in the **Customer** table consist of:
  - ✓ CustomerID,
  - ✓ CustomerName,
  - ✓ ContactName,
  - ✓ Address,
  - ✓ City,
  - ✓ PostalCode,
  - ✓ Country.

What is row or record?

- A **row** of a table is also called ***record***.
- It contains the specific information of each individual entry in the table.
- For example: The above table contains 5 records.

3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Table

Field

Record

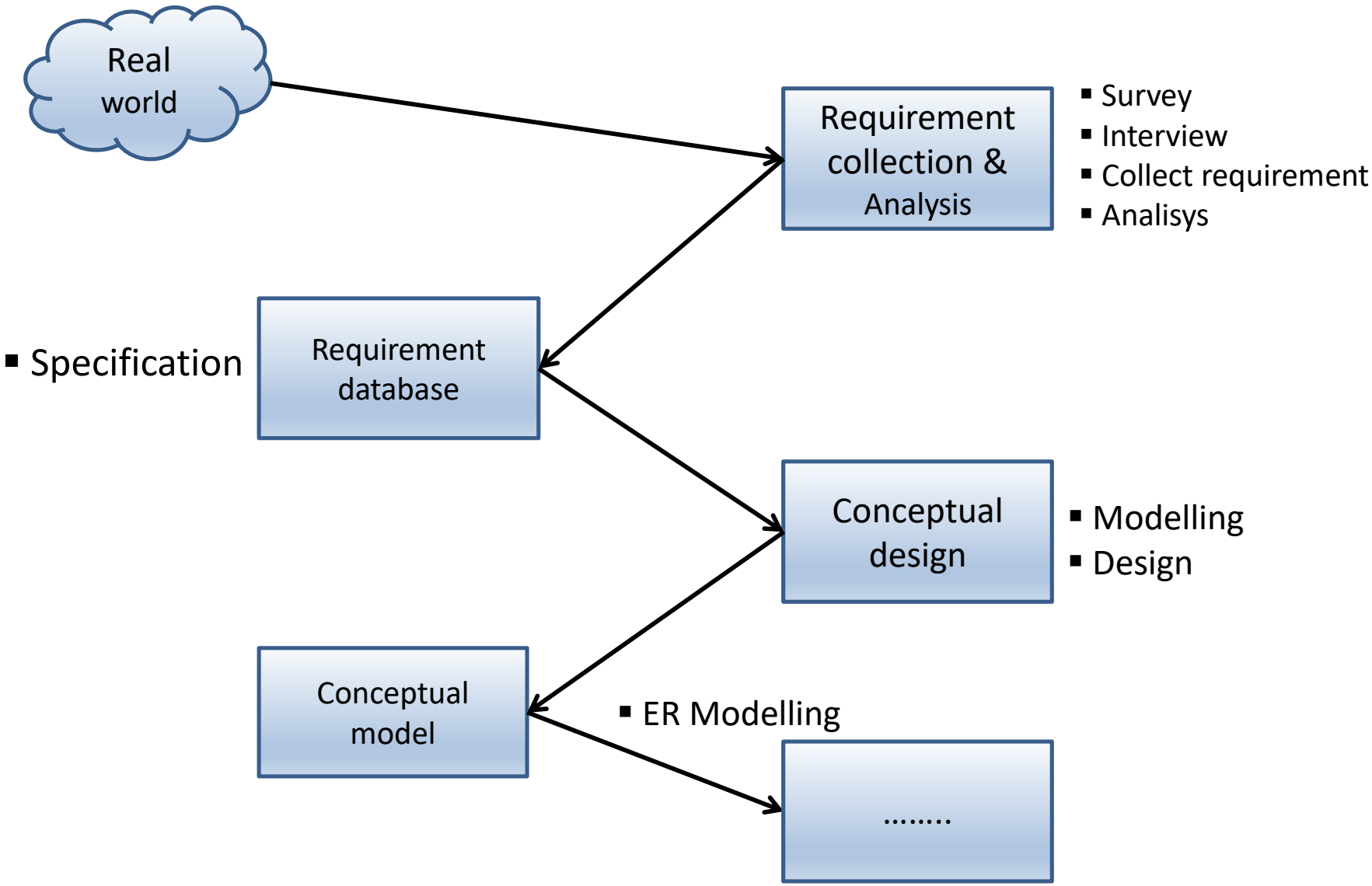
CD_ID	Title	Artist	Genre
1	The Wall	Pink Floyd	Rock
2	Blue Train	John Coltrane	Jazz
3	Requiem	W.A. Mozart	Classical

What is column?

- A **column** is a *vertical entity* in the table which contains all information associated with a specific field in a table.
- For example: “CustomerName" is a column in the above table which contains all information about student's name.

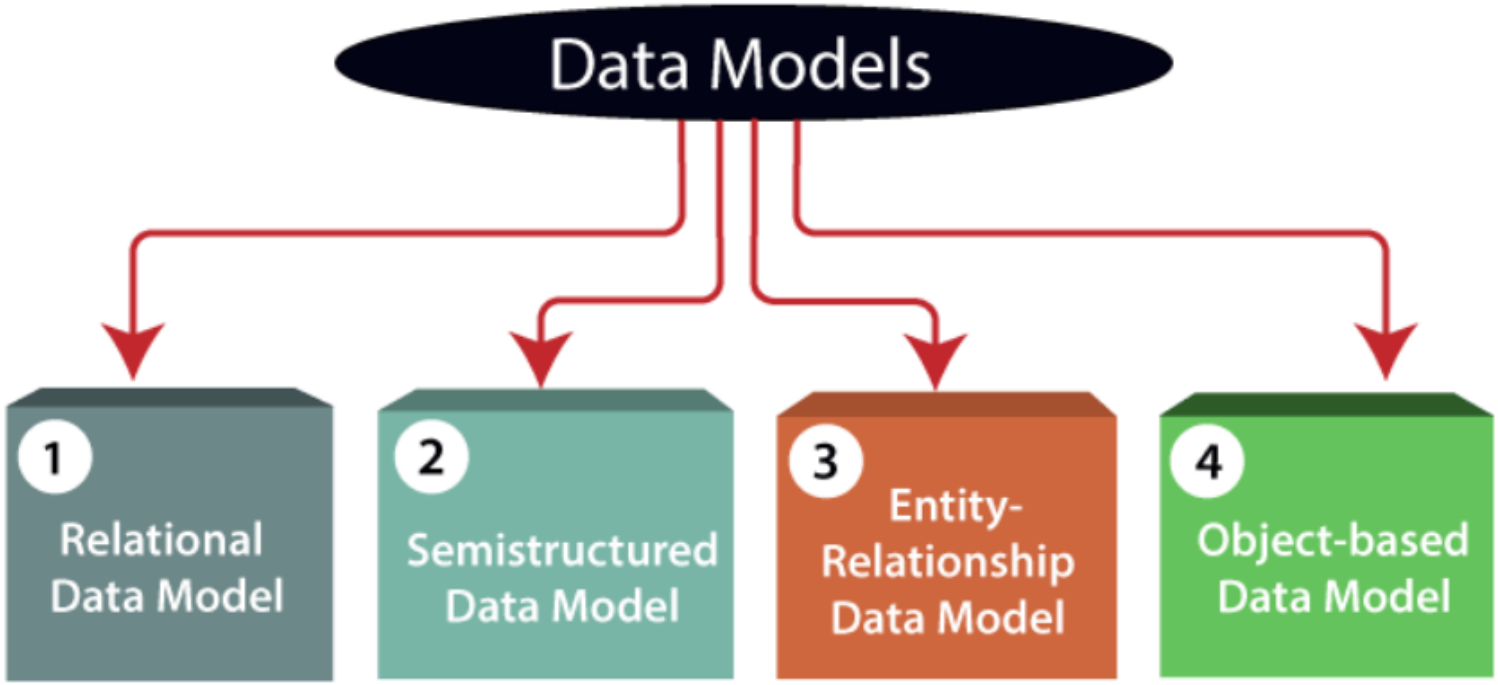
CustomerName
Alfreds Futterkiste
Ana Trujillo Emparedados y helados
Antonio Moreno Taquería
Around the Horn
Berglunds snabbköp

Design Process



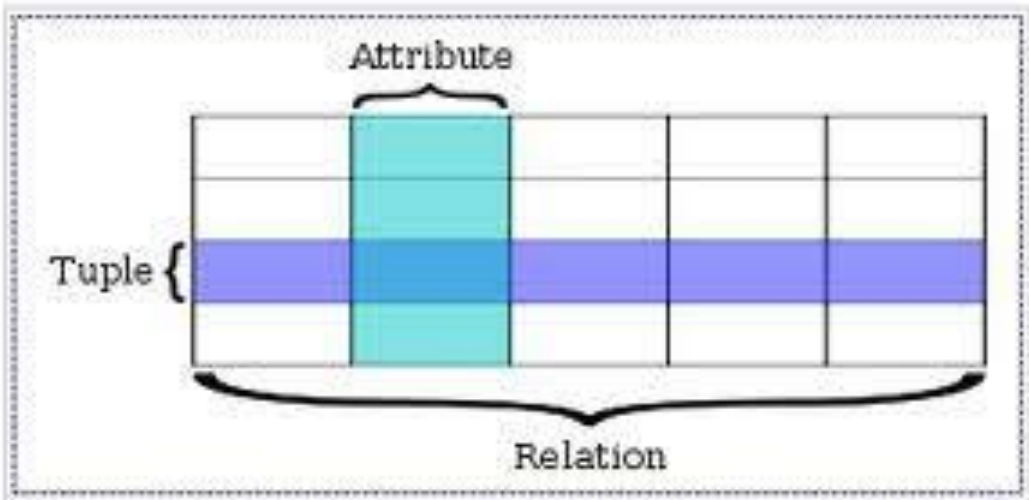
Data Models

- **Data Model** is the modeling of the data description, data semantics, and consistency constraints of the data.
- It provides the **conceptual tool**s for describing the design of a database at each level of **data abstraction**.



Data Models

- **Relational Data Model:**
  - ✓ This type of model designs the data in the form of **rows** and **columns** within a table.
  - ✓ **A relational model** uses **tables** for representing data and **in-between relationships**.
  - ✓ **Tables** are also called **relations**.
  - ✓ This model was initially described by Edgar F. Codd, in 1969



Relational database terminology.

Example Relation Instances

Emp Relation

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36

Proj Relation

pno	pname	budget	dno
P1	Instruments	150000	D1
P2	DB Develop	135000	D2
P3	Budget	250000	D3
P4	Maintenance	310000	D2
P5	CAD/CAM	500000	D2

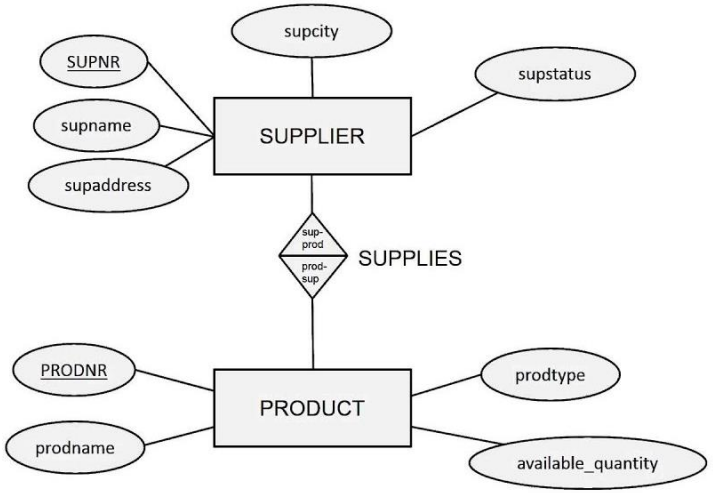
Dept Relation

dno	dname	mgreno
D1	Management	E8
D2	Consulting	E7
D3	Accounting	E5
D4	Development	null

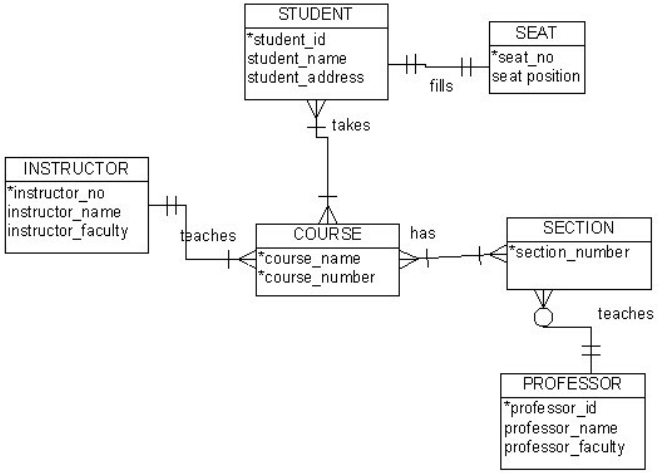
Data Models

- **Entity-Relationship Data Model:**
  - ✓ **An ER model** is the logical representation of data as objects and relationships among them.
  - ✓ These objects are known as **entities**, and **relationship** is an association among these entities.
  - ✓ This model was designed by Peter Chen and published in 1976 papers.
  - ✓ A set of attributes describe the entities.
  - ✓ For example, **student_name**, **student_id** describes the 'student' entity.

Peter Chen and published in 1976

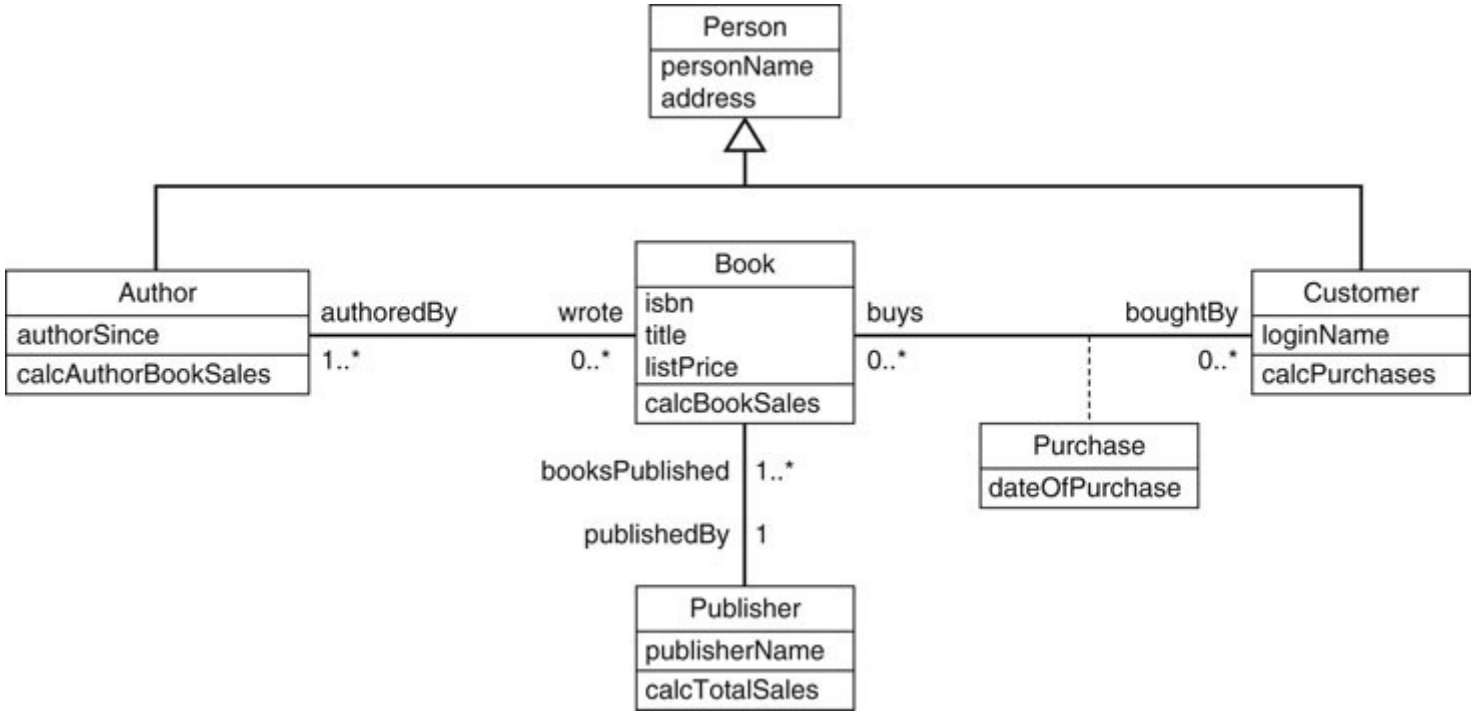


Using Crow's Foot Notation in an ERD



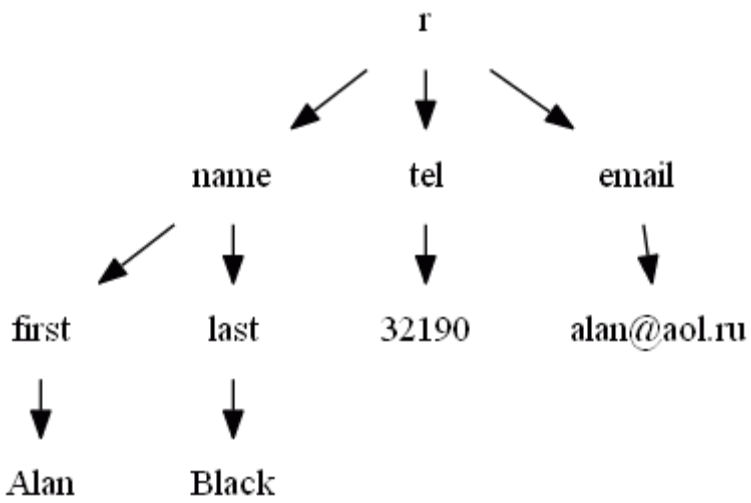
Data Models

- **Object-based Data Model:** An extension of the ER model with notions of **functions**, **encapsulation**, and **object identity**, as well.
- **Example**:



Data Models

- **Semistructured Data Model:**
  - ✓ This type of data model is different from the other three data models (explained above).
  - ✓ The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets.
  - ✓ The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data.



Section 2

ER MODEL

ER Model

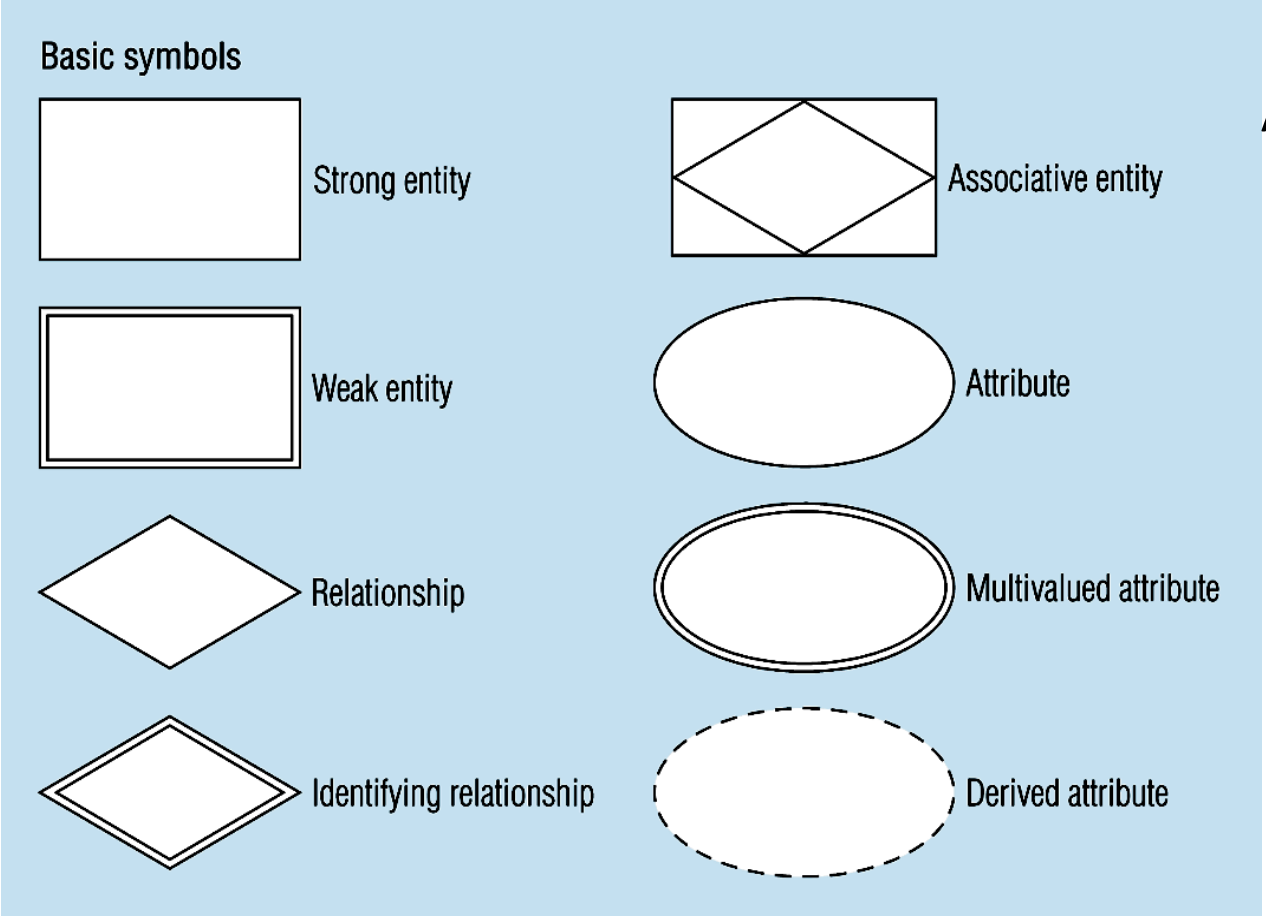
- **ER Notation (Chen's Notation vs Crow's foot Notation)**
- **Entity Types**
- **Relationship Types**
- **Attributes**
- **Keys**
- **Relationship Cardinalities**

1- ER Notation

Basic E-R Notation

The model was designed by Peter Chen (Chen's Notation)

Entity symbols



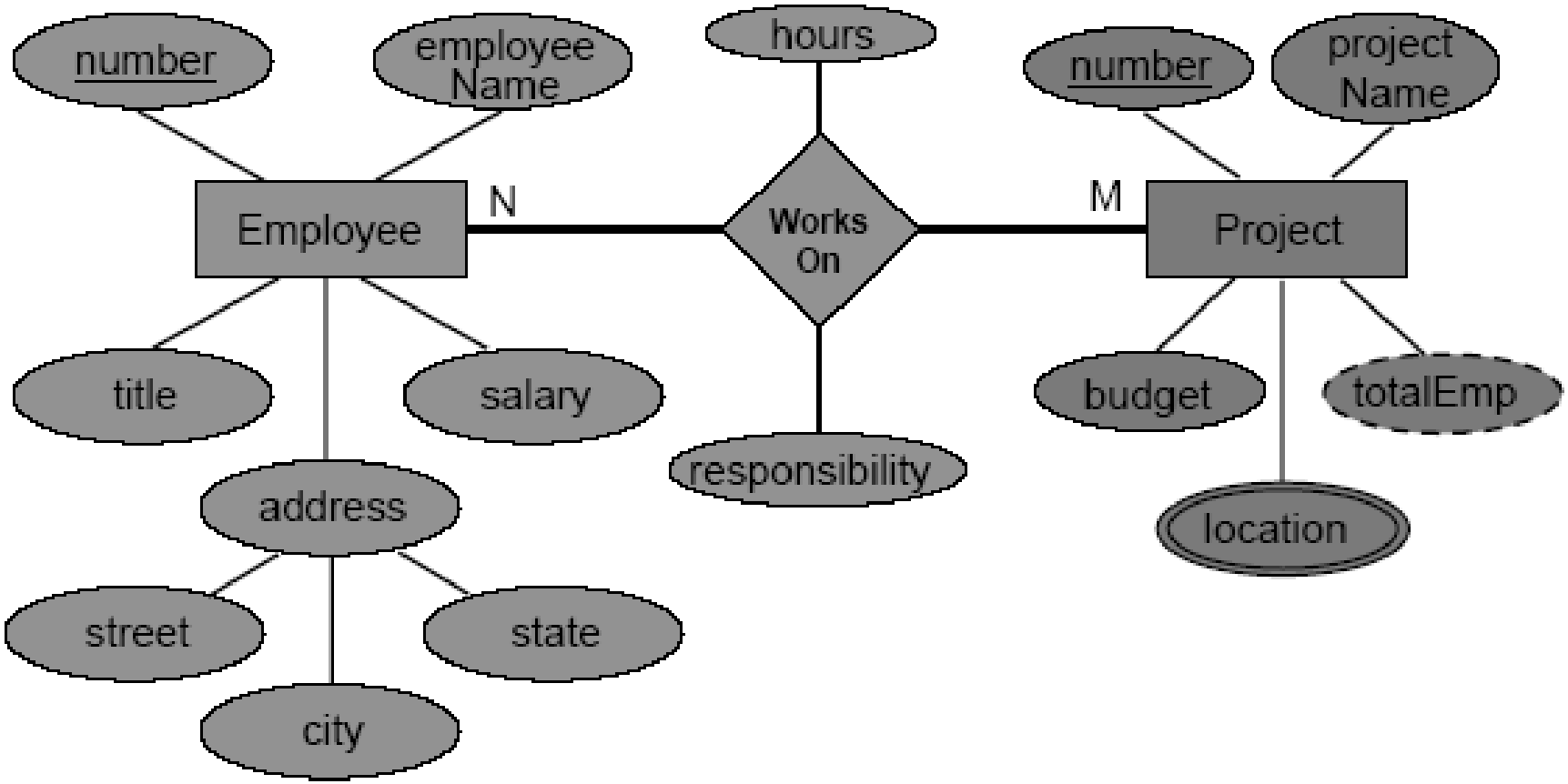
A special entity that is also a relationship

Relationship symbols

Attribute symbols

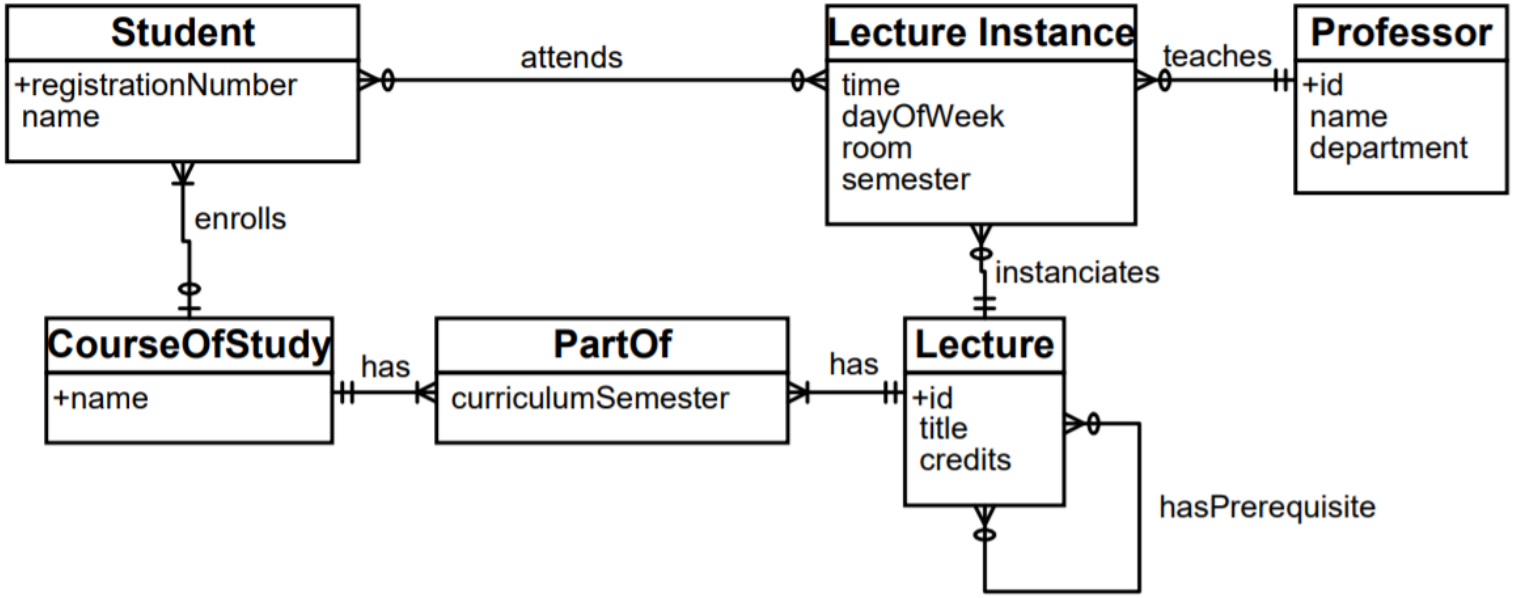
ER Model Example

The model was designed by Peter Chen (Chen's Notation)



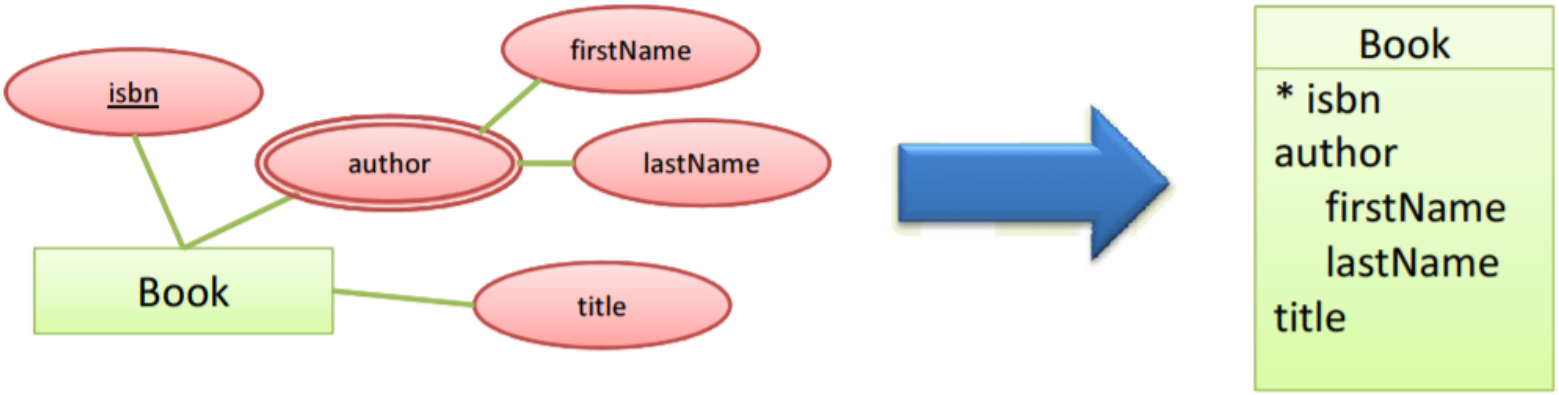
ER – Crow's Foot Notation

- **Crow's foot notation** was initially developed by **Gordon Everest**.
- **Main Goal:**
  - ✓ Consolidate graphical representation
  - ✓ Provide a better and faster overview
  - ✓ Allow for easier layouting
- **Example:**



ER – Crow's Foot Notation

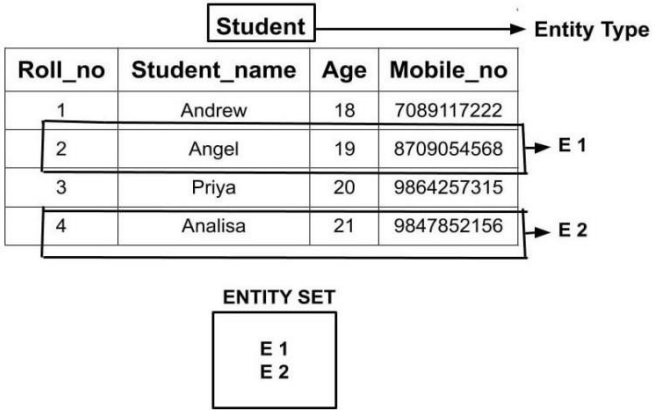
- **Entity Types Notation:**
  - ✓ Entity Types are modeled with a named box.
  - ✓ Attribute names are written inside the box separated by a line



2- Entity Types and Relationship Types

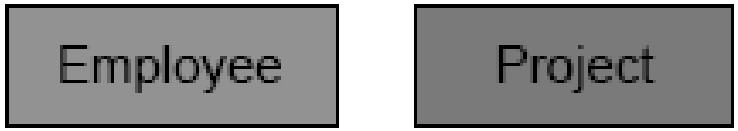
Entity Types

- An ***entity type*** is a group of objects with the same properties which are identified as having an independent existence.
  - ✓ An entity type is the basic concept of the ER model and represents a group of real-world objects that have properties.
    - Note that an entity type does not always have to be a physical real-world object such as a person or department, it can be an abstract concept such as a project or job.
- An ***entity instance*** is a particular example or occurrence of an entity type.
  - ✓ For example, an entity type is Employee. A entity instance is '**E1 - John Doe**'.
- An ***entity set*** is a set of entity instances.



Representing Entity Types

- In ER notation (and UML), entity types are represented by rectangles with the name of the entity type in the rectangle.
- **Examples**:



✓ An entity type name is normally a singular noun.

- That is, use Person instead of People, Project instead of Projects, etc.

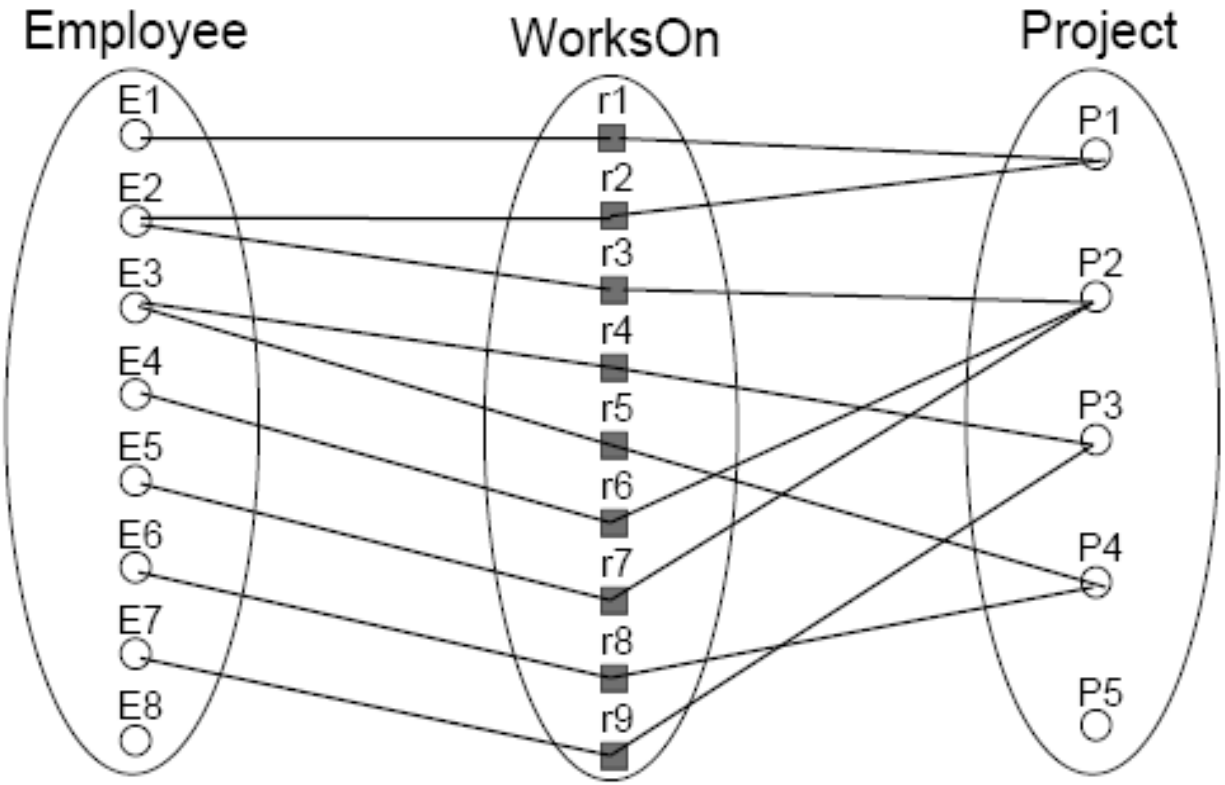
✓ The first letter of each word in the entity name is typically capitalized.

Relationship Types

- A ***relationship type*** is a set of meaningful associations among entity types. Each relationship type is given a name that describes its function.
- A ***relationship instance*** is a particular occurrence of a relationship type that relates entity instances.
  - ✓ For example, WorksOn is a relationship type. A relationship instance is that 'E1' works on project 'P1' or (E1,P1).
- A ***relationship set*** is a set of relationship instances.
- Note that there can be more than one relationship between two entity types.

Visualizing Relationships

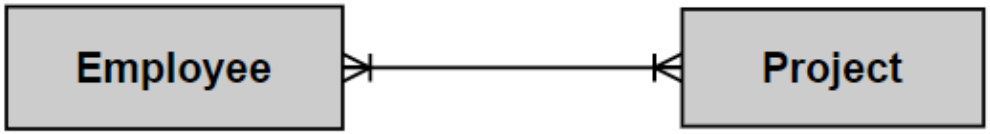
- Note: This is an example of a many-to-many relationship.
- A project can have more than one employee, and an employee can work on more than one project.



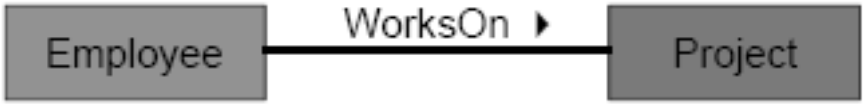
Representing Relationship Types

- In classical ER notation, a simple relationship type between two entities is represented as **a named diamond** that connects the two entity types:

In Crow's Foot Notation:



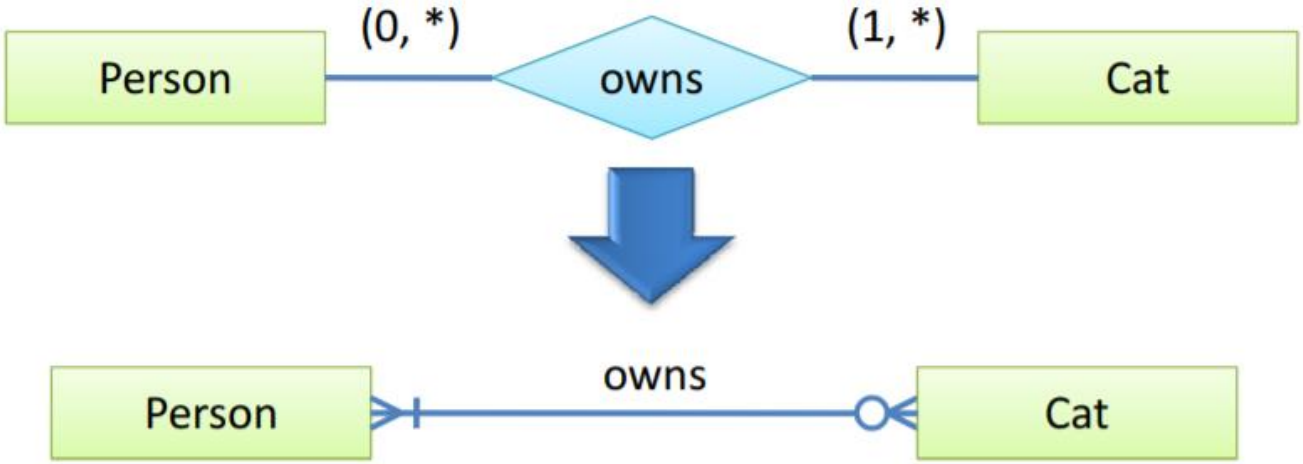
In UML:



Representing Relationship Types

Crow's Foot Notation

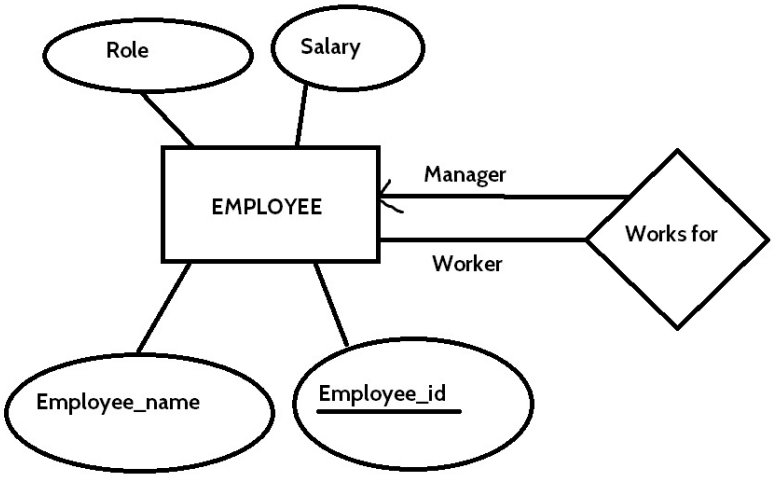
- Relationship types are modeled by lines connecting the entities.
- Line is annotated with the name of the relationship which is a verb
- Cardinalities are represented graphically:
  - ✓ (0, 1) : Zero or one
  - ✓ (1, 1) : Exactly one 
  - ✓ (0, *) : Zero or more 
  - ✓ (1, *) : one or more 



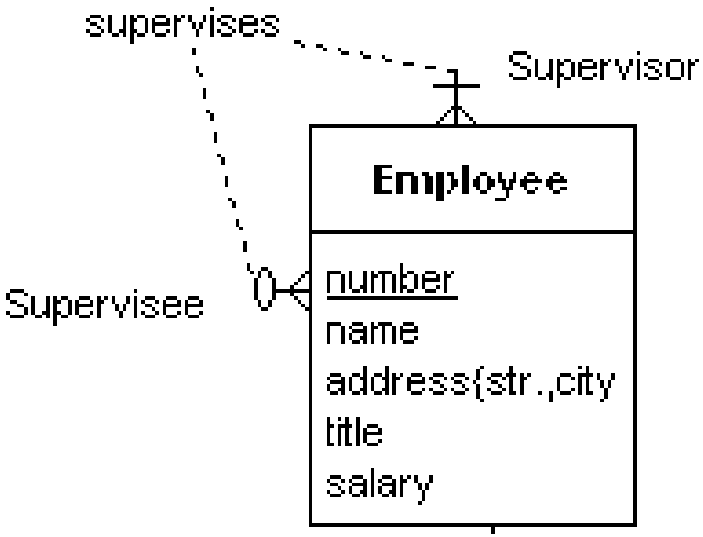
Recursive Relationships

- A ***recursive relationship*** is a relationship type where the same entity type participates more than once in different roles.
  - ✓ For example, an employee has a supervisor. The supervisor is also an employee.

Chen's notation:



Crow's foot notation:



Practice time

The Human Resources division tracks information about the employees and the facilities:

- In the Human Resource (HR) records, each employee has an identification number, e-mail address, job identification code, salary, and manager. Some employees earn commissions in addition to their salary.
- The company also tracks information about jobs within the organization. Each job has an identification code, job title, and a minimum and maximum salary range for the job.
- Some employees have been with the company for a long time and have held different positions within the company. When an employee resigns, the duration the employee was working, the job identification number, and the department are recorded
- The sample company is regionally diverse, so it tracks the locations of its warehouses and departments. Each employee is assigned to a department, and each department is identified either by a unique department number or a short name.
- Each department is associated with one location, and each location has a full address that includes the street name, postal code, city, state or province, and the country code. In places where the departments and warehouses are located, the company records details such as the → Country (country name, currency symbol, currency name, and the region where the country is located geographically.

Practice time – Entity Types

The Human Resources division tracks information about the employees and the facilities:

- In the Human Resource (HR) records, each **employee** has an identification number, e-mail address, job identification code, salary, and manager. Some employees earn commissions in addition to their salary.
- The company also tracks information about jobs within the organization. Each **job** has an identification code, job title, and a minimum and maximum salary range for the job.
- Some employees have been with the company for a long time and have held different positions within the company. When an employee resigns, the duration the employee was working, the job identification number, and the department are recorded. → **Job_History**
- The sample company is regionally diverse, so it tracks the locations of its warehouses and **departments**. Each employee is assigned to a department, and each department is identified either by a unique department number or a short name.
- Each department is associated with one **location**, and each location has a full address that includes the street name, postal code, city, state or province, and the country code. In places where the departments and warehouses are located, the company records details such as the → **Country** (country name, currency symbol, currency name, and the **region** where the country is located geographically.

3- Attributes

Attributes

- An ***attribute*** is a property of an entity or a relationship type.
  - ✓ For example, entity type Employee has attributes name, salary, title, etc.
- **Some rules:**
  - ✓ By convention, attribute names begin with a lower case letter (optional).
  - ✓ Each attribute has a *domain* which is the set of allowable values for the attribute.
  - ✓ Different attributes may share the same domain, but a single attribute may have only one domain.

Simple and Complex Attributes

- An attribute is a ***simple attribute*** if it contains a single component with an independent existence.
  - ✓ For example, salary is a simple attribute.
  - ✓ Simple attributes are often called *atomic* attributes.
- An attribute is a ***composite attribute*** if it consists of multiple components each with an independent existence.
  - ✓ For example, address is a complex attribute because it consists of street, city, and state components (subattributes).
- **Question**: Is the name attribute of Employee simple or complex?

Single- and Multi-Valued Attributes

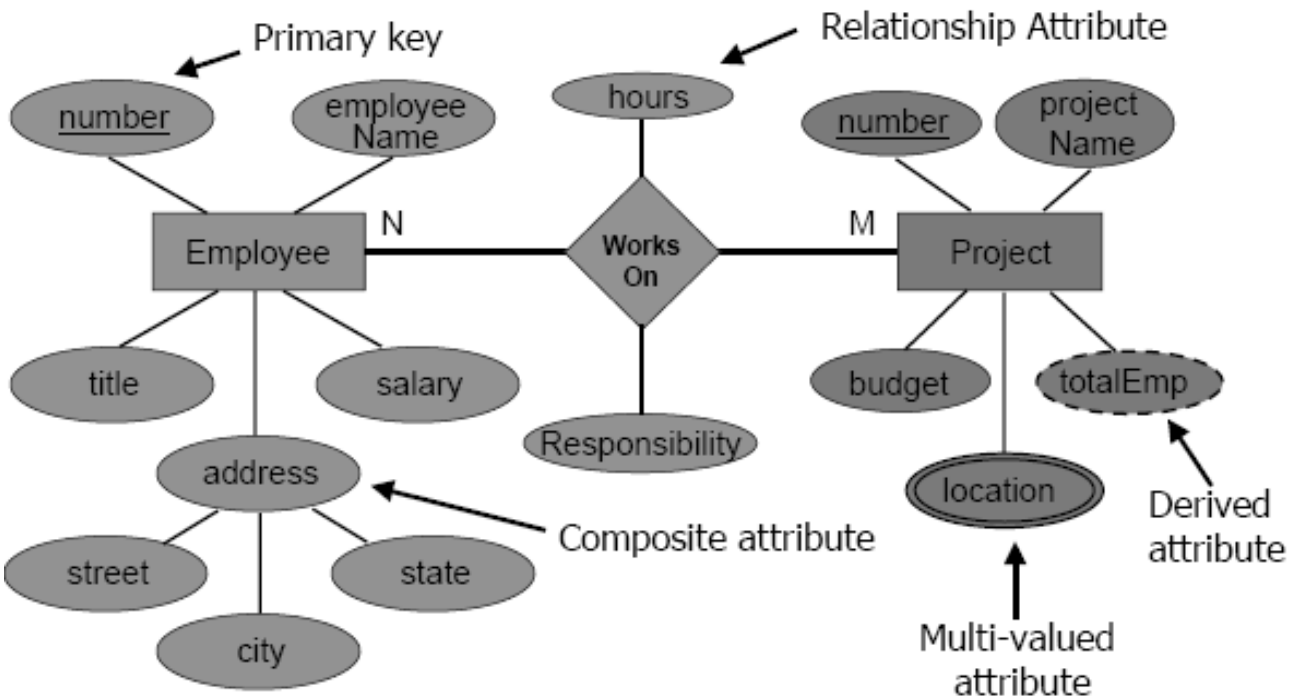
- An attribute is a ***single-valued attribute*** if it consists of a single value for each entity instance.
  - ✓ For example, salary is a single-valued attribute.
- An attribute is a ***multi-valued attribute*** if it may have multiple values for a single entity instance.
  - ✓ For example, a telephone number attribute for a person may be multivalued as people may have different phone numbers (home phone number, cell phone number, etc.)
- A ***derived attribute*** is an attribute whose value is calculated from other attributes but is not physically stored.

Keys

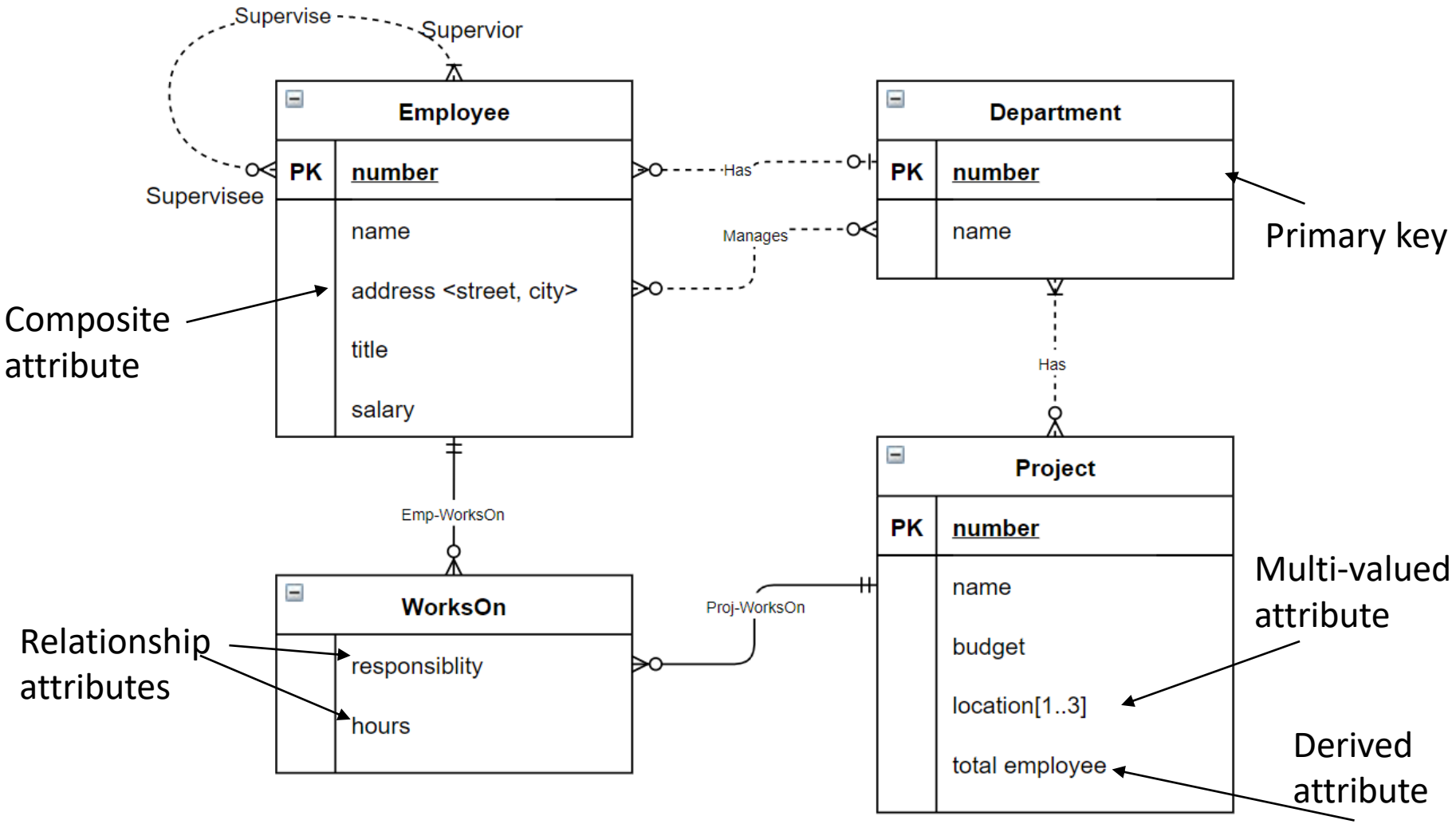
- A ***candidate key*** is a minimal set of attributes that uniquely identifies each instance of an entity type.
  - ✓ For example, the number attribute uniquely identifies an Employee and is a candidate key for the Employee entity type.
- A ***primary key*** is a candidate key that is selected to identify each instance of an entity type.
  - ✓ The primary key is chosen from a set of candidate keys. For instance, an employee may also have SSN as an attribute. The primary key may be either SSN or number as both are candidate keys.
- A ***composite key*** is a key that consists of two or more attributes.
  - ✓ For example, a course is uniquely identified only by the department code (22C) and the course number within the department (144).

Attributes on Relationships

- An attribute may be associated with a relationship type.
  - ✓ For example, the WorksOn relationship type has two attributes: responsibility and hours.
- Note that these two attributes belong to the relationship and cannot belong to either of the two entities individually (as they would not exist without the relationship).



Crow's Foot Notation



4-Relationship Cardinalities

Relationship Cardinalities

- ***Relationship cardinalities*** or ***multiplicities*** are used to restrict how entity types participate in relationships in order to model real-world constraints.
- The ***multiplicity*** is the number of possible occurrences of an entity type that may relate to a single occurrence of an associated entity type through a particular relationship.
- **For binary relationships, there are three common types:**
  - ✓ one-to-one (1:1)
  - ✓ one-to-many (1:* or 1:N)
  - ✓ many-to-many (*:* or N:M)

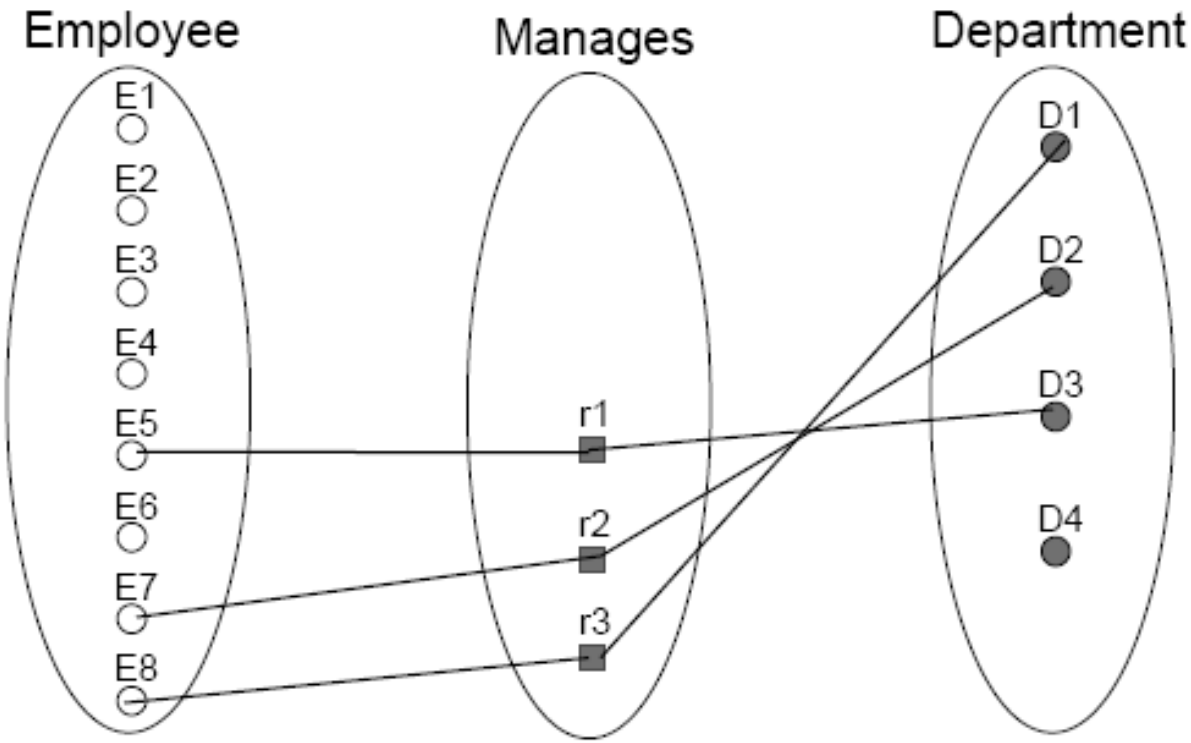
One-to-One Relationships

- In a one-to-one relationship, each instance of an entity class E1 can be associated with **at most one** instance of another entity class E2 and vice versa.
- **Example**: A department may have only one manager, and a manager may manage only one department.

Each Department may have at most one manager.

Each Employee has zero or one Departments.

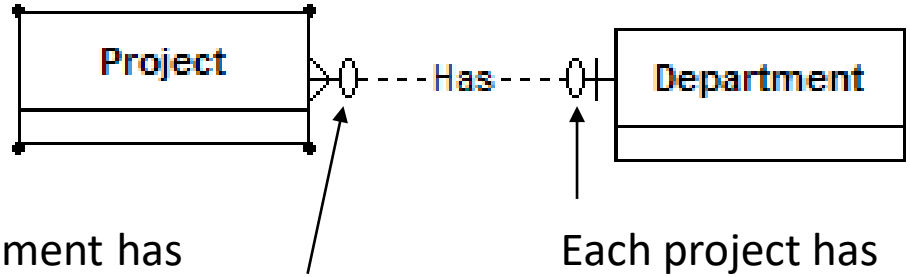
One-to-One Relationship Example



Relationship explanation: A department may have only one manager. A manager (employee) may manage only one department.

One-to-Many Relationships

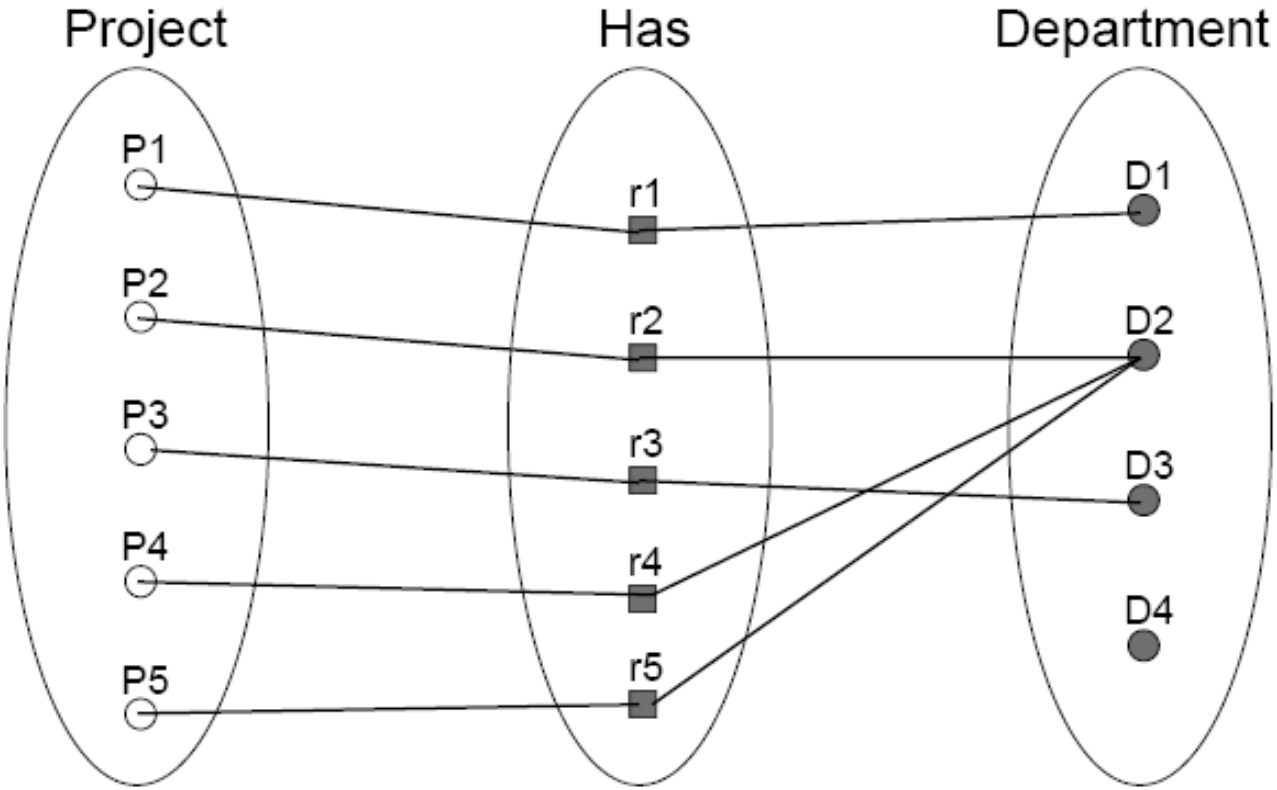
- In a one-to-many relationship, each instance of an entity class E1 can be associated with **more than one** instance of another entity class E2.
- However, E2 can only be associated with **at most one** instance of entity class E1.
- **Example**: A department may have multiple projects, but a project may have only one department.



Each department has zero or more projects.

Each project has zero or one departments.

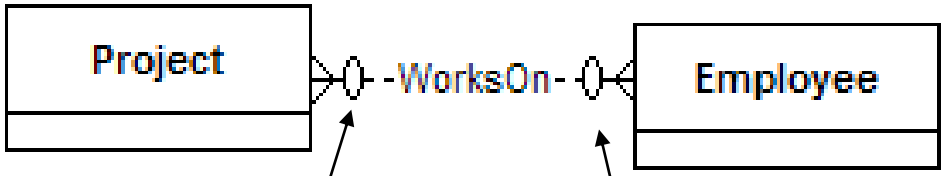
One-to-Many Relationship Example



Relationship explanation: A project may be associated with at most one department. A department may have multiple projects.

Many-to-Many Relationships

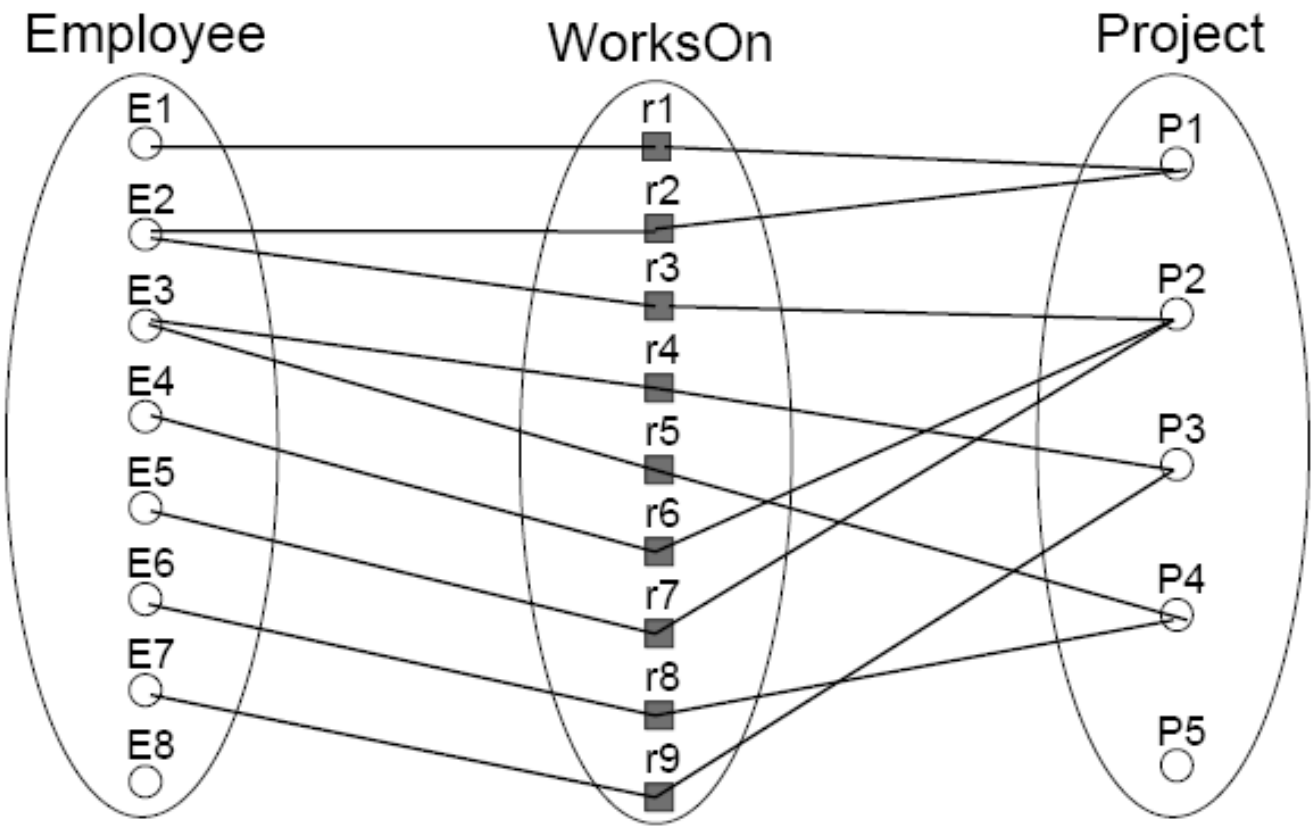
- In a many-to-many relationship, each instance of an entity class E1 can be associated with **more than one** instance of another entity class E2 and vice versa.
- **Example**: An employee may work on multiple projects, and a project may have multiple employees working on it.



Each project has zero or more employees.

Each employee works on zero or more projects.

Many-to-Many Relationship Example



ER Design Question

- **Construct a university database where:**
  - ✓ Each **student** has an id, name, gender, birth date, and GPA.
  - ✓ Each **professor** has a name and is in a department.
  - ✓ Each **department** offers courses and has professors. A department has a name and a building location.
  - ✓ Each **course** has a name and number and may have multiple sections.
  - ✓ Each **section** is taught by a professor and has a section number.
  - ✓ Students enroll in sections of courses. They may only enroll in a course once (and in a single section). Once a student completes a course, they receive a grade.

Summary

- Basic concept of Database and DBMS
- ER Model
  - ER Notation (Chen's Notation vs Crow's foot Notation)
  - Entity Types
  - Relationship Types
  - Attributes
  - Keys
  - Relationship Cardinalities

Thank you