

Database Systems

Lecture 3 – 4 – 5 – 6

Instructor: Bilal Khalid Dar



Agenda

- Data Model
 - Data Model Structure and Constraints
 - Data Model Operations
- Categories of Data Models
 - (Conceptual – Physical - Implementation)
- Database Schema VS Database State, Data Base Instance
- Relational DBMS
- Database Design /Database Model
- Database Development Stages

Agenda

- Relational Model
 - What is Relational Model,
 - The Use Of Relational Model,
 - Parts of Relational Model (Structural, Manipulative, Integrity)
- Database Relations
- Properties of Relation
- Relational Keys
- Representing Relational DB Schema
- Integrity Constraints
- View, Purpose of Views, Updating View

Data Models

Data Model: A set of concepts to describe the structure of a database, the operations for manipulating these structures, and certain constraints that the database should obey.

Data Model Structure and Constraints

- Constructs are used to define the database structure
- Constructs typically include elements (and their data types) as well as groups of elements (e.g. entity, record, table), and relationships among such group
- Constraints specify some restrictions on valid data; these constraints must be enforced at all times

Data Models (continued)

- Data Model Operations:
- These operations are used for specifying database retrievals and updates by referring to the constructs of the data model.
- Operations on the data model may include basic model operations (e.g. generic insert, delete, update, remove) and user-defined operations (e.g. compute_student_gpa, update_inventory)

Categories of Data Models :

Conceptual (high-level, semantic) data models:

- Provide concepts that are close to the way many users perceive data.
- (Also called entity-based or object-based data models.)

Physical (low level internal) data models:

- Provide concepts that describe details of how data is stored in the computer.
- These are usually specified in an ad-hoc manner through DBMS design and administration manuals

Implementation (representational) data models:

- Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems)

Schemas versus Instances

•Database Schema:

- The description of a database
- Includes descriptions of the database structure, data types, and the constraints on the database

•Schema Construct:

A component of the schema or an object within the schema, e.g., STUDENT, COURSE.

Schemas versus Instances

•Database State

- The actual data stored in a database at a particular moment in time. This includes the collection of all the data in the database.
- It is also called database instance (or occurrence or snapshot).
- The term instance is also applied to individual database components, e.g. record instance, table instance, entity instance.

Schemas versus Instances

- Database State:

Refers to the content of a database at a moment in time.

- Initial Database State:

Refers to the database state when it is initially loaded into the system.

- Valid State:

A state that satisfies the structure and constraints of the database.

Schemas versus Instances

Distinction (Note)

- The database schema changes very infrequently.
- The database state changes every time table is updated
- Schema is also called intension.
- State is also called extension.
- Schema – Structural description of relations in database
- Schema is a repository or structure to express the format and other different information about data and database,
- Actual place where these definitions and descriptions are performed is database schema.
- Instance – Actual contents at given point in time

Example of a Database Schema

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

Figure 2.1

Schema diagram for the database in Figure 1.2.

Example of a database state

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2

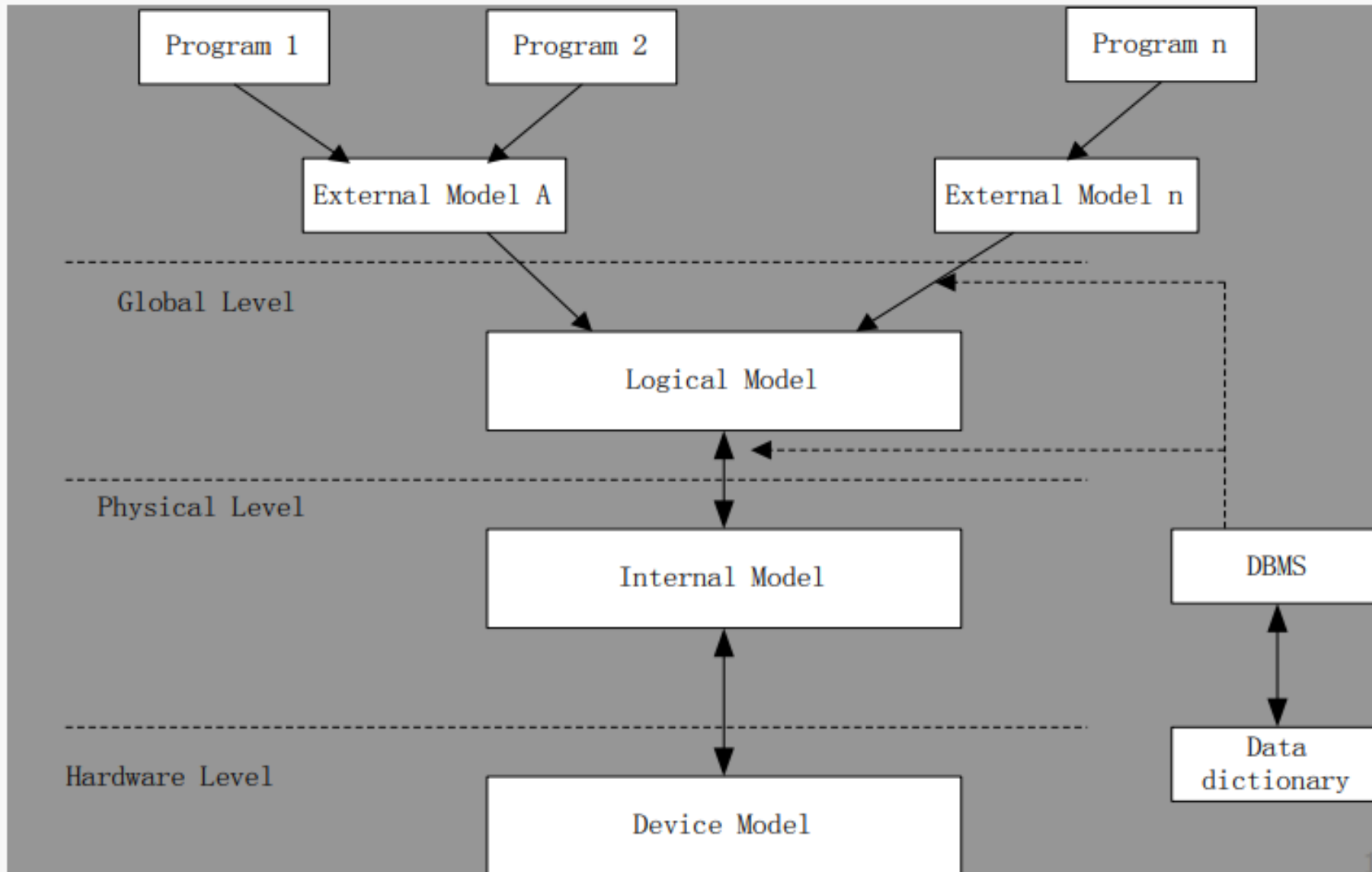
A database that stores student and course information.

Relational DBMS - (RDBMS)

- **RDBMS** is one of the most widely used and easiest to operate database systems.
- **Relational data model (RDM)** – Theoretical basis of RDBMS.
- **Logical data structure – Table (relation)**

A C. Each row (tuple) contains one value for the associated column.

Relational DBMS - (RDBMS)



Database Design /Database Model

- These terms can be used interchangeably for the logical structure of the database.
- The Database design/model stores the structure of the data and the links/relationships between Data that should be stored to meet the users' requirements.
- Database design is stored in the database schema, which is in turn stored in the data dictionary

Database Modeling

- **The process of creating the logical structure of the database is called database modeling.**
- It is a very important process because the designing of the application provides us the basis for running our database system. If the database is not designed properly the implementation of the system can not be done properly

Design Stages



Fig: Database Development Stages.

Relational Model

- The relational model is the conceptual basis of relational databases. Proposed by E.F. Codd in 1969, it is a method of structuring data using relations, which are grid-like mathematical structures consisting of columns and rows. Codd (1970) introduced normalized relations in the relational model .

- Very simple model
- Efficient implementations
- Used by all major commercial database systems
- Query with high-level languages: simple yet expressive

Relational Model - The Use – Why a relational model?

To represent data in an understandable way

It is basis for designing relational RDBMSs.

- 1) The designer to reason about the database systems without actually having build them;
- 2) 2) An RDBMS implements the structures, operations and rules of the relational model.

Relational Model

- Three parts of a relational model:

1) **Structural part**, Description of the **database elements** of and the **structure** of the database system.

2) **Manipulative part**, **Operations** working on databases to implement the functions of systems;

3) **Integrity part**, The **rules** that databases must obey – to make sure the system work properly.

Relational Model

1- The Structural Part

Example COMPANY Database

- We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:
 - The company is organized into DEPARTMENTS. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager. A department may have several locations.
 - Each department *controls* a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.

Example COMPANY Database (Contd.)

- We store each EMPLOYEE's social security number, address, salary, gender, and birthdate.
 - Each employee *works for* one department but may *work on* several projects.
 - We keep track of the number of hours per week that an employee currently works on each project.
 - We also keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTS.
 - For each dependent, we keep track of their name, gender, birthdate, and relationship to the employee.

ER Model Concepts

- Entities and Attributes
 - Entities are specific objects or things in the mini-world that are represented in the database.
 - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
 - Attributes are properties used to describe an entity.
 - For example an EMPLOYEE entity may have the attributes Name, SSN, Address, Gender, BirthDate
 - A specific entity will have a value for each of its attributes.
 - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Gender='M', BirthDate='09-JAN-55'
 - Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, subrange, enumerated type, ...

Types of Attributes (1)

Simple

- Each entity has a single atomic value for the attribute. For example, SSN or Gender.

Composite

- The attribute may be composed of several components. For example:
 - Address(Apt#, House#, Street, City, State, ZipCode, Country), or
 - Name(FirstName, MiddleName, LastName).
- Composition may form a hierarchy where some components are themselves composite.

Multi-valued

- An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT.
 - Denoted as {Color} or {PreviousDegrees}.

1- The structural part

Components and the structure of database systems:

- Relations;
- Attributes;
- Keys;
- Domains

Relational database: is made up of relations, i.e., relations are the elements of database systems

Relation: is a structure describing the relationships between things in the real world.

Book_Title	Authors	Published date	Price
Relational Databases	BEA	2000.5	12.35
Java	EFH	2001.2	16.50

1- The structural part

The table – relations;

The columns – attributes;

The column headings – attribute names;

The values in the column – attribute values.

The rows -- tuples.

Table::Relation

Column::attribute

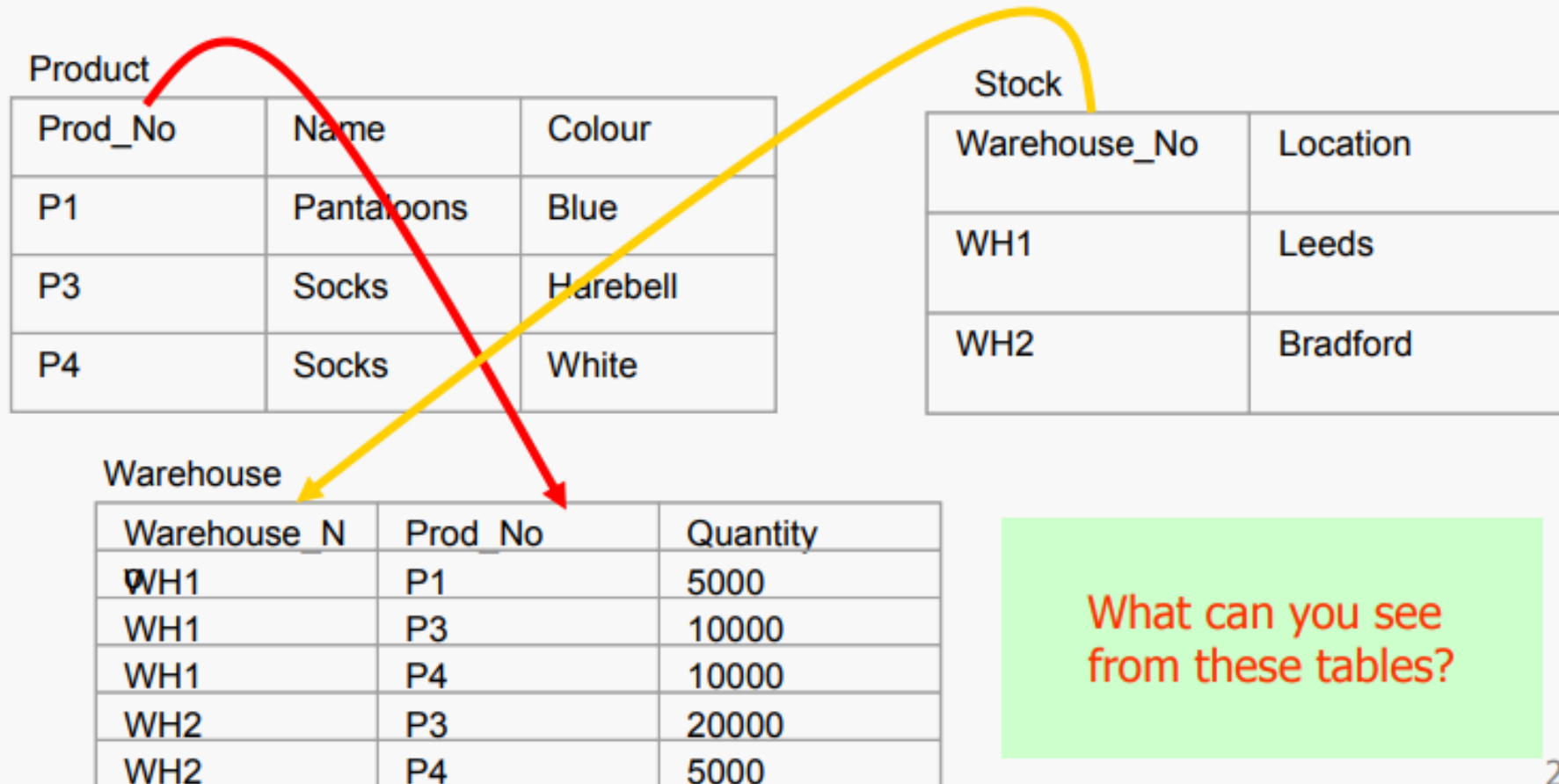
Book_Title	Authors	Published date	Price
Relational	BEA	2000.5	12.35
Databases	EFH	2001.2	16.50
Java			

??

Row::Tuple

1- The structural part

- Example: to understand the concepts



1- The structural part

- **Concepts: Relations, attributes and tuples**

- A relational database -- a number of relations (tables)
- **Relational Database** is a collection of normalized relations with distinct relation names.
 - **Each relation has a set of named attributes (or columns)**
 - A relation is made up of tuples (rows);
 - **Each tuple (or row) has a value for each attribute**
 - An attribute is given a name,
 - **Each attribute has a type (or domain)**
 - Each attribute must take a value from the domain.
 - **what is domain?**

1- The structural part

■ Domain:

Prod_No	Name	Colour
P1	Pantaloons	Blue
P3	Socks	Harebell
P4	Socks	White

Domain of Prod_No:
{P1, P2, P3}

Domain of Name:
{Pantaloons, Socks}

- A domain is a set of values, from which one or more attributes can take values.
- Every value in a database must be a member of a certain domain;
- Domain values must be single-valued.

Q: What is the domain of {name and Colour} ?

Domain: A domain is the set of allowable values for one or more attributes

Relational Data Structure Terminologies

- A relational database : A number of **relations (tables)**.
- Relation (File, Table): A relation is a **table** with columns and rows.
- **Intension**: The structure of a relation, together with a specification of the domains and any other restrictions on possible values, is sometimes called its intension
- Attribute(Column, Fields):An attribute is a named **column** of a relation
- **Degree**: The degree of a relation is the **number of attributes** it contains
- Tuple (Row, Record): A tuple is a **row** of a relation.
- **Cardinality**: The cardinality of a relation is the number of tuples it contains.
- **Extension**: The tuples are called the extension (or state) of a relation, which changes over time.
- Domain: A domain is the set of allowable values for one or more attributes.

Relational Data Structure Terminologies...

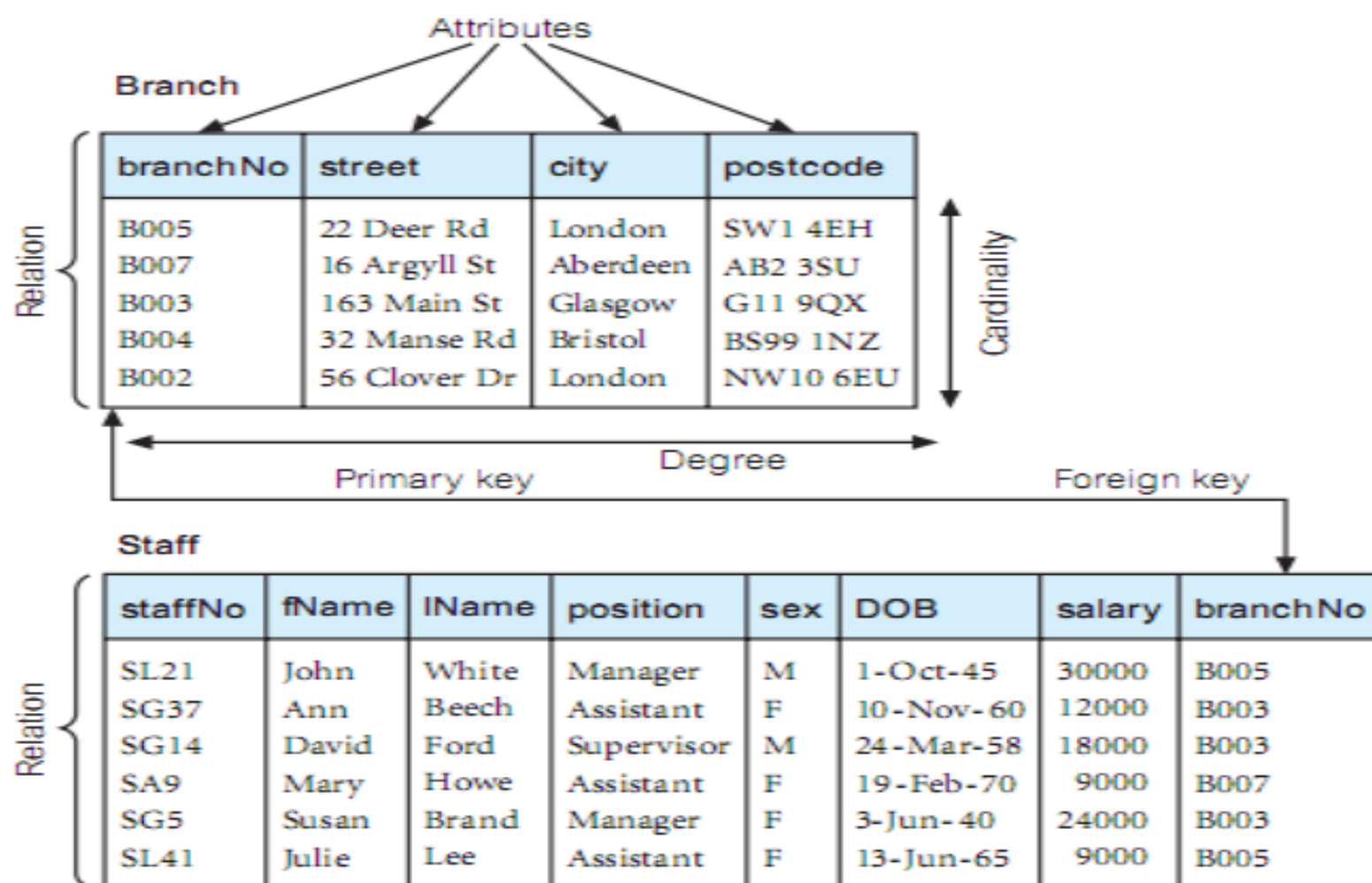


Figure : Instances of Branch and Staff Relations

Relational Model

1- The Structural Part

KEYS

1- The structural part (Keys)

Concept: Key

Prod_No	Name	Colour
P1	Pantaloons	Blue
P3	Socks	Harebell
P4	Socks	White

HOW to distinguish each PRODUCT?

Need to define a Key?

1- The structural part (Keys)

■ Example: the keys

Product

Prod_No	Name	Colour
P1	Pantaloons	Blue
P3	Socks	Harebell
P4	Socks	White

Stock

Warehouse_No	Location
WH1	Leeds
WH2	Bradford

Warehouse

Warehouse_N	Prod_No	Quantity
WH1	P1	5000
WH1	P3	10000
WH1	P4	10000
WH2	P3	20000
WH2	P4	5000

Keys:

- Product: Prod_No.
- Warehouse: Warehouse_No, or Location.
- Stock: Warehouse NO and Prod_No.

1- The structural part (Keys)

Concept: Key

- **The value of a key must be unique for each tuple.**
- A key can be one or more than one attribute.
- **Key.** A key is a set of attributes whose values uniquely name each tuple of relation.
- **Super-key** is a key plus some other attributes

1- The structural part (Keys)

Note 1:

- The key for STOCK is made up of **two** attributes, i.e. any one of them is not key.
- **Super-key:** key + other attributes, e.g., Warehouse_NO+Prod_No, + Quantity.

Prod_No	Name	Colour
P1	Pantaloons	Blue
P3	Socks	Harebell
P4	Socks	White

Warehouse_No	Location
WH1	Leeds
WH2	Bradford

Warehouse_No	Prod_No	Quantity
WH1	P1	5000
WH1	P3	10000
WH1	P4	10000
WH2	P3	20000
WH2	P4	5000

1- The structural part (Keys)

Note 2:

- A relation may have many keys – they are called *candidate keys*.
- Every relation should have at least one **candidate key**.
- One candidate key is selected as the *primary key*.
- The candidate keys not selected as the primary key are called *alternate* keys.

Warehouse_No	Location
WH1	Leeds
WH2	Bradford

Keys:

- Warehouse: Warehouse_No, or Location.

1- The structural part (Keys)

- A **primary key** is selected so that we can make sure that each tuple has a unique identifier.
- The attributes of primary key are not allowed to have NO value at all.
- **A foreign key:** attributes whose values are used elsewhere as primary key values.
 - Prod_No: a primary key in Product,
 - A foreign key in Stock.
- The associated primary key and foreign key are defined on the same domain but not necessary have the same attribute names.

Relational Model

2- The Integrity Part

2- The Integrity Part

■ What is the integrity part?

- A set of rules that the Relational Database must obey
- Which ensure the database works properly as a system.

■ What are the rules ?

Only two general rules:

1. Entity integrity rules
2. Referential rules

2- The Integrity Part

■ 1- Entity Integrity Rules

Each tuple must include a unique name or identifier.
i.e. The primary key must have a non-null value.

■ 2- Referential Integrity Rules

All cross-referenced objects must be within the database.

The **foreign key** value must be either NULL or a value that occurs elsewhere as A PRIMARY KEY.

2- The Integrity Part

■ Example

Customer

Cust_No	Name	Address
C45	Boys	Newsome
C46	Girls	Holey

Sales_Order

Order_No	Date	Cust_No
011	1/7/89	C45
012	1/8/89	Null
013	2/5/88	C47
null	1/1/89	C45

Anything wrong??

Relational Keys

- **Superkey**
 - A Super key is any combination of fields within a table that uniquely identifies each record within that table.
- **Candidate Key**
 - A candidate is a subset of a super key.
 - A candidate key is a single field or the least combination of fields that uniquely identifies each record in the table. The least combination of fields distinguishes a candidate key from a super key.
 - Every table must have at least one candidate key but at the same time can have several.

Relational Keys

- **Candidate Key**
 - In order to be eligible for a candidate key it must pass certain criteria.
 - It must contain unique values
 - It must not contain null values
 - It contains the minimum number of fields to ensure uniqueness
 - It must uniquely identify each record in the table

Relational Keys

- **Primary Key**

- Candidate key selected to identify tuples uniquely within relation.

- **Alternate Keys OR Secondary Key**

- Candidate keys that are not selected to be primary key.

- **Foreign Key**

- Attribute, or set of attributes, within one relation that matches candidate key of some (possibly same) relation.

Relational Keys

- **Composite Primary Key (Composite Key)**
- When we have a Primary Key of a table defined using more than one columns then it is known as a Composite Key, each columns data can be duplicated, but combined values cannot be. The columns which are participating in a composite primary key are not simple keys.
- **For Example**, we can have a situation where there is a need to define the key using first Name + last Name.

Integrity Constraints

- Null
 - Represents value for an attribute that is currently unknown or not applicable for tuple.
 - Deals with incomplete or exceptional data.
 - Represents the absence of a value and is not the same as zero or spaces, which are values.

Database Design

- Before we look at how to create and use a database we'll look at how to design one
- Need to consider
 - What tables, keys, and constraints are needed?
 - What is the database going to be used for?
- Conceptual design
 - Build a model independent of the choice of DBMS
- Logical design
 - Create the database in a given DBMS
- Physical design
 - How the database is stored in hardware

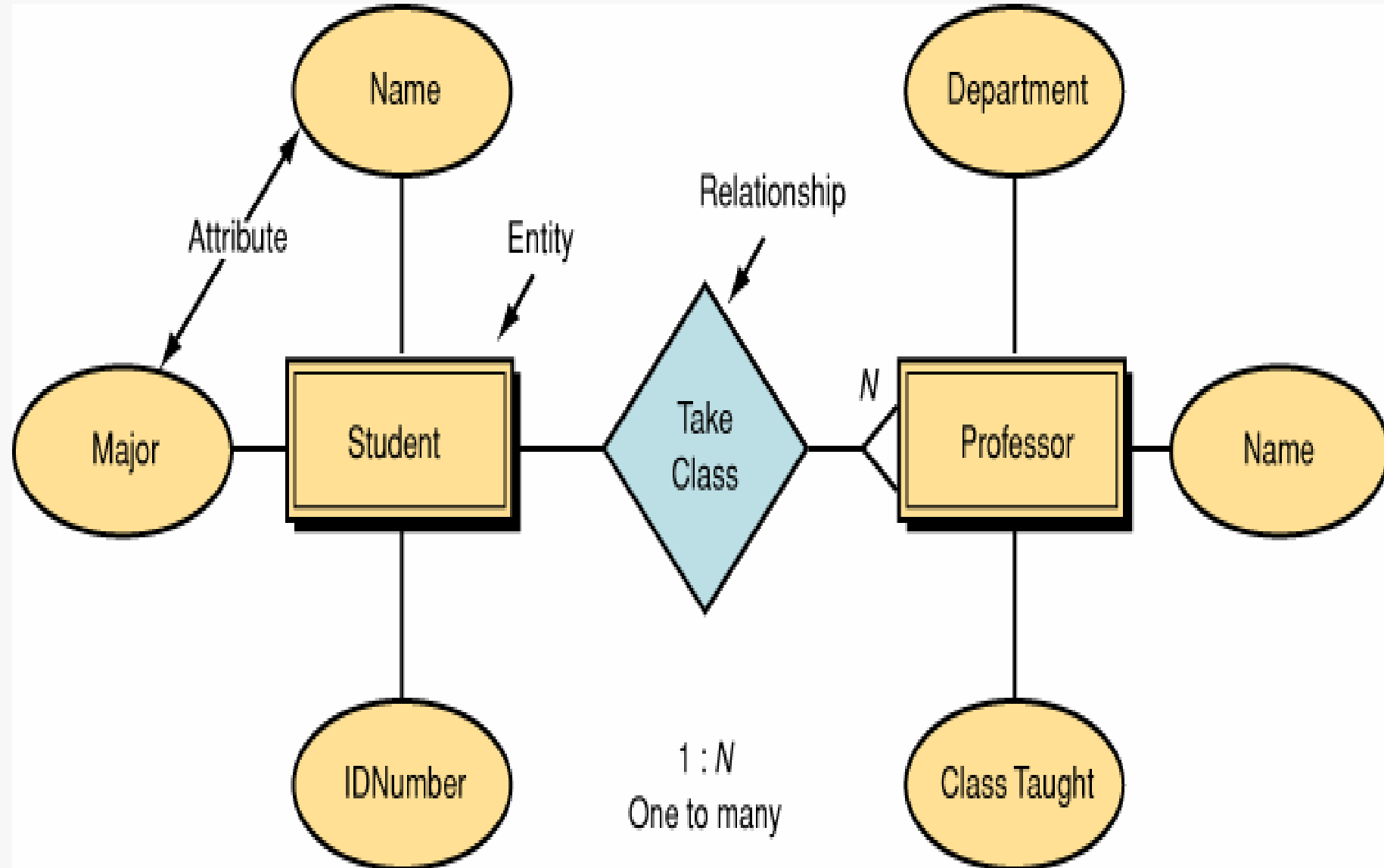
What is an Entity Relationship Diagram (ERD)?

- ERD is a data modeling technique used in software engineering to produce a conceptual data model of an information system.
- So, ERDs illustrate the logical structure of databases.

Entity/Relationship Modelling

- E/R Modelling is used for conceptual design
 - Entities - objects or items of interest
 - Attributes - facts about, or properties of, an entity
 - Relationships - links between entities
- Example
 - In a University database we might have entities for Students, Modules and Lecturers. Students might have attributes such as their ID, Name, and Course, and could have relationships with Modules (enrolment) and Lecturers (tutor/tutee)

Data Model by Peter Chen' Notation (first - original)

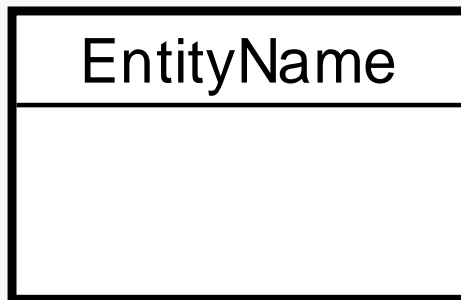


Making E/R Models

- To make an E/R model you need to identify
 - Entities
 - Attributes
 - Relationships
 - Cardinality ratios
- from a description
- General guidelines
 - Since entities are things or objects they are often nouns in the description
 - Attributes are facts or properties, and so are often nouns also
 - Verbs often describe relationships between entities

Entities

- Entity - distinguishable “thing” in the real world
 - Strong (or regular) entity - entities have an independent existence (e.g. staff)
 - Weak entity - existence dependent on some other entity (e.g. next of kin)



Entity type name
(singular, no spaces,
capital letter at start of each word)

space for attributes

Relationship (Type)

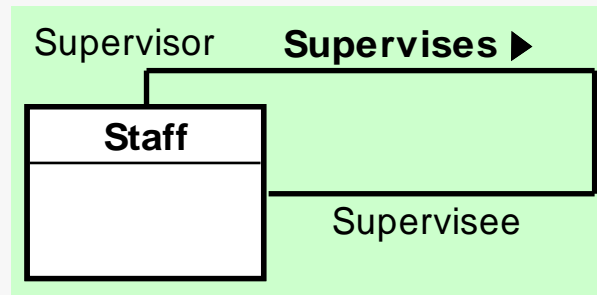
- Definition
 - A Relationship Type is an association among Entity Types. It indicates that there is a business relationship between these Entity Types.
 - Relationship Membership is the participation of an Entity Type in a Relationship.
 - In IE, a Relationship Type can involve only two Entity Types (binary relationship). Some other modeling techniques allow n-ary relationships.
- Examples
 - CUSTOMER places ORDER
 - ORDER is placed by CUSTOMER
 - EMPLOYEE works on PROJECT
 - PROJECT has project member EMPLOYEE

Identifying Relationships

- Association between entity types
- Entity types that are used on the same forms or documents.
- A description in a business document that has a verb that relates two entity types
 - has
 - consists of
 - uses

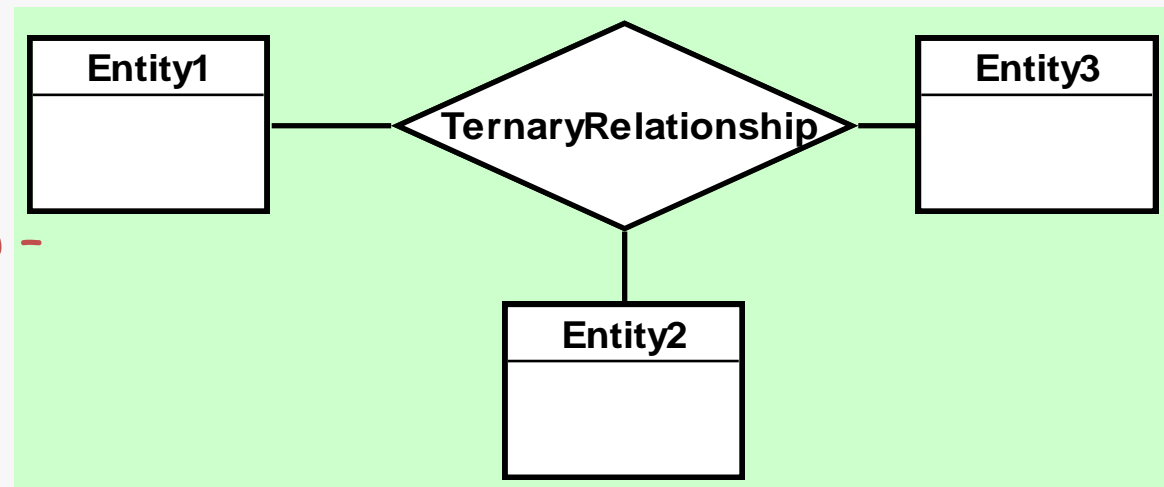
Relationships: Degree

Binary relationship

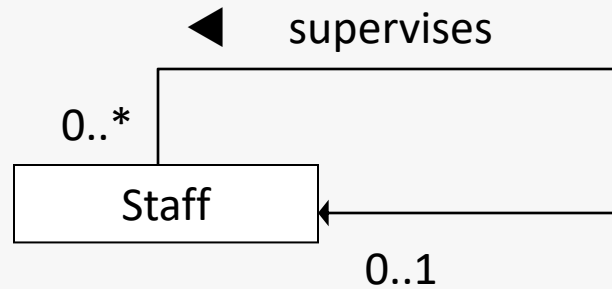


Recursive (Unary) relationship - example

Complex relationship - here ternary



Unary Example with Data



A member of staff may supervise another staff member, but a staff member may be supervised by one or more staff members

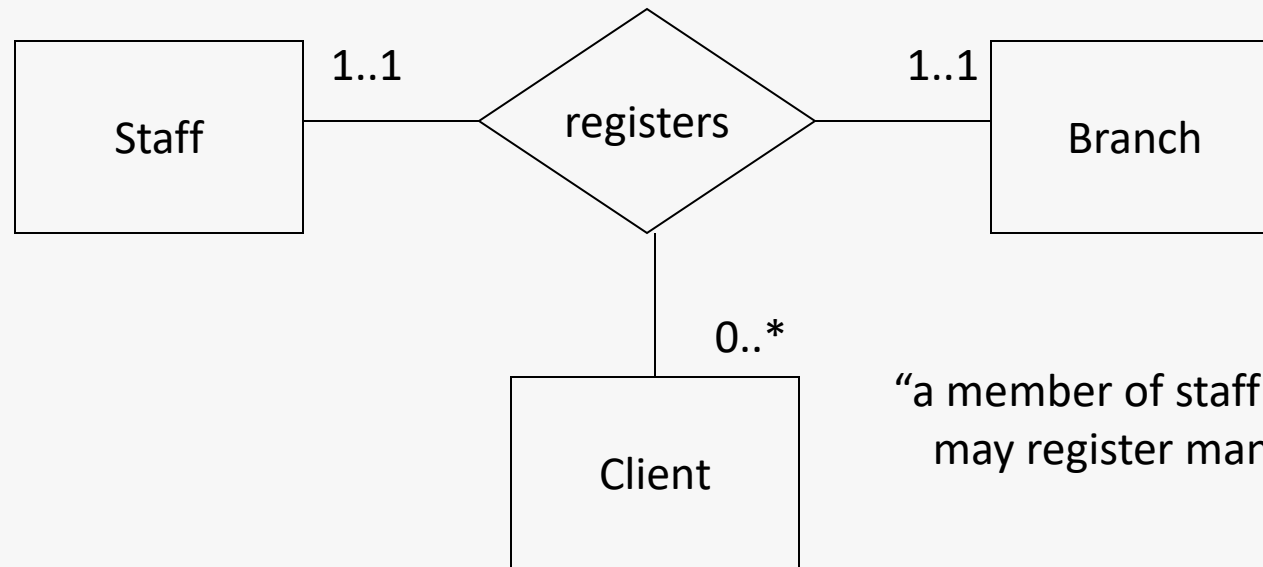
STAFF

<u>Member</u>	Age	Supervisor
Grey	43	Black
Black	27	
Brown	35	Black
White	33	Brown

Ternary Diagrams are Tricky!

“a client at a branch will be registered by one member of staff”

“a member of staff will register a client at one branch”



“a member of staff at a branch may register many clients”

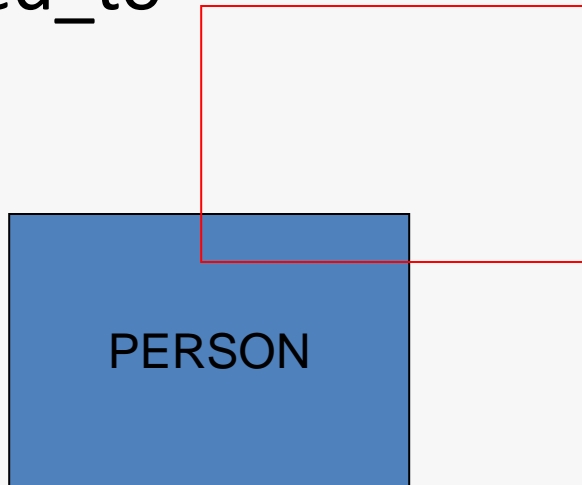
Try to determine participation/cardinality by operating in pairs

Scenario modified from Connolly & Begg page 350

Unary Relationship - Another Example

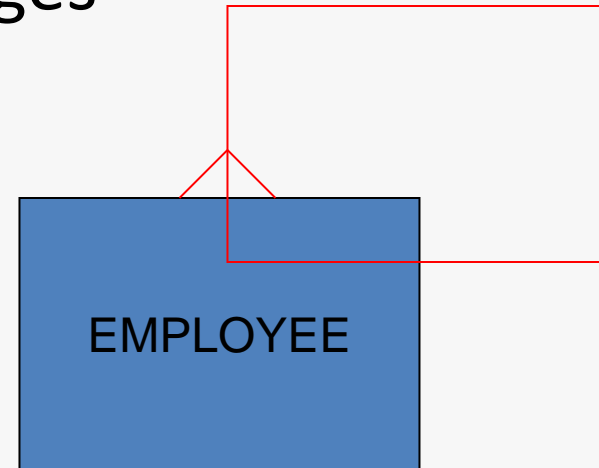
- Relationship between the instances of one entity type.

Is_married_to



One-to-one

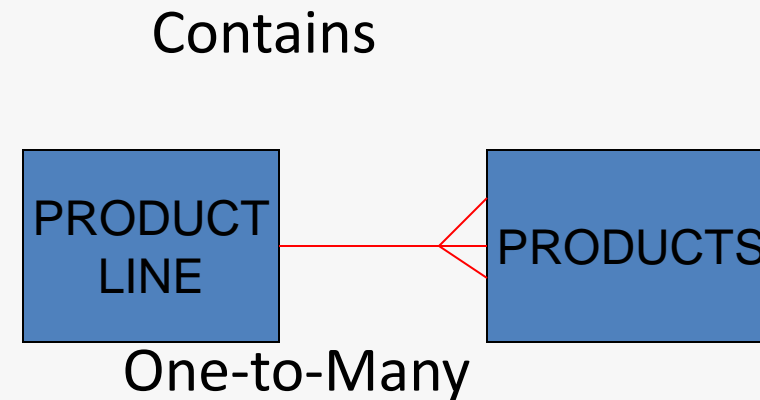
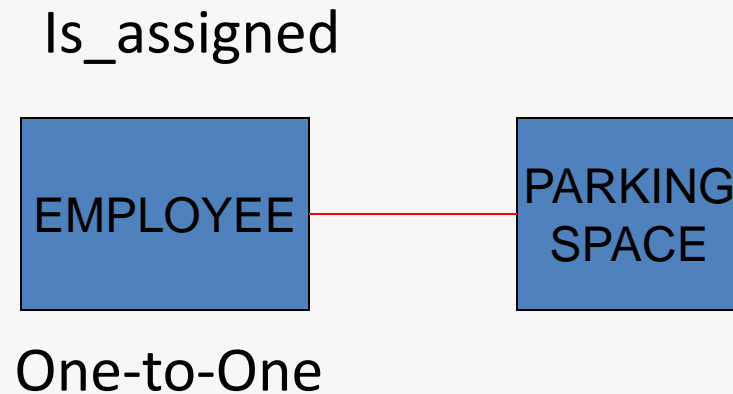
Manages



One-to-many

Binary Relationship - Another Example

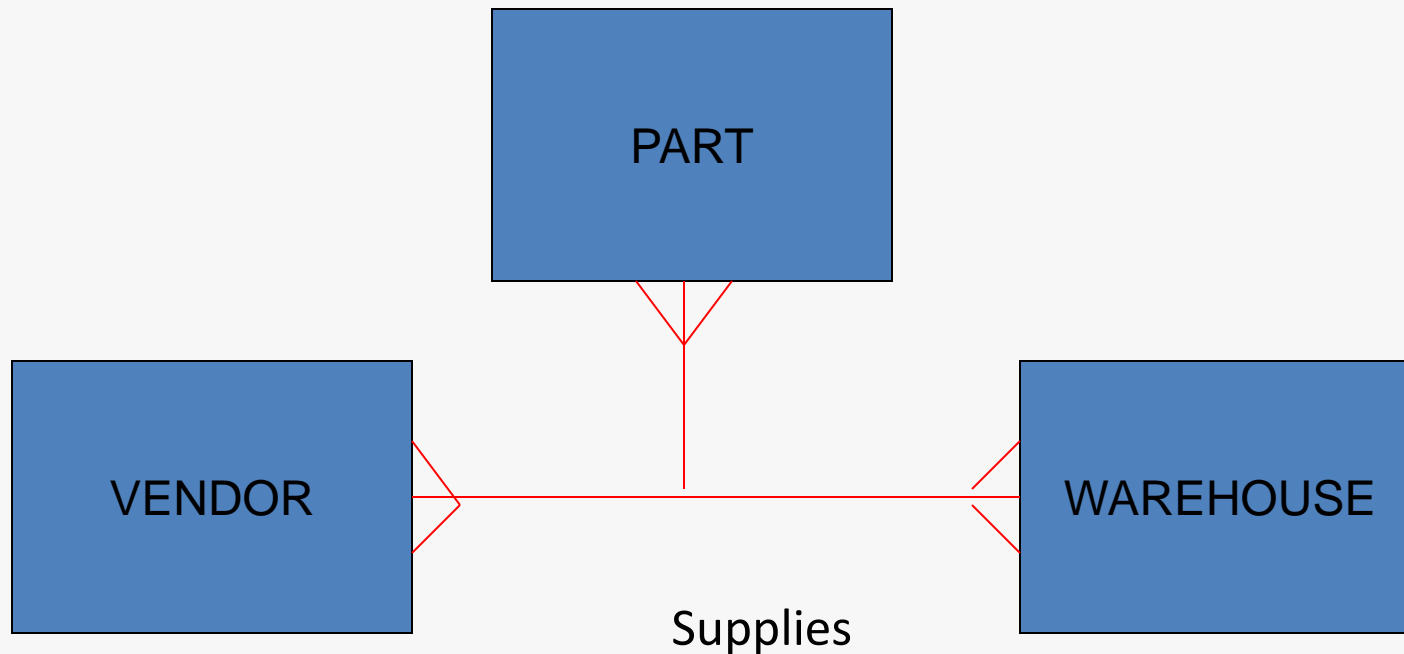
- Relationship between the instances of two entity type.



Can also have many to many!

Ternary Relationship - Another Example

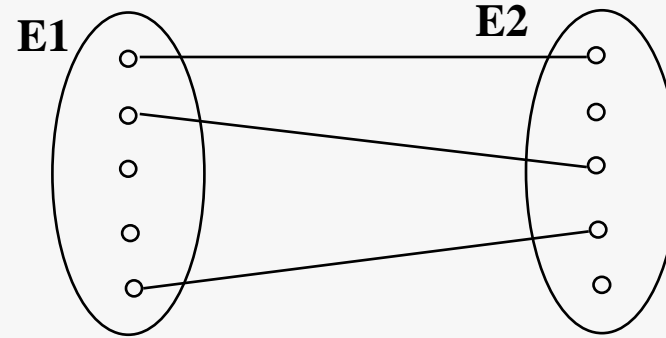
- A simultaneous relationship among instances of three entity types.



Relationship Cardinality

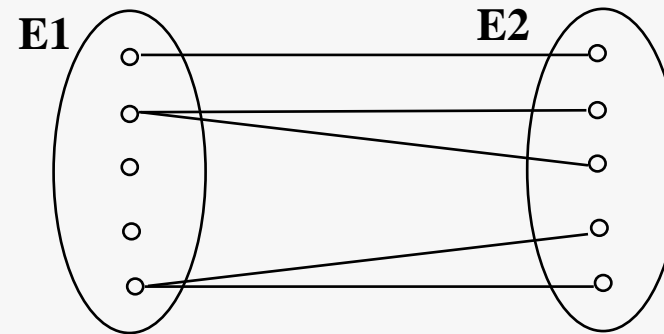
One-to-One

1:1



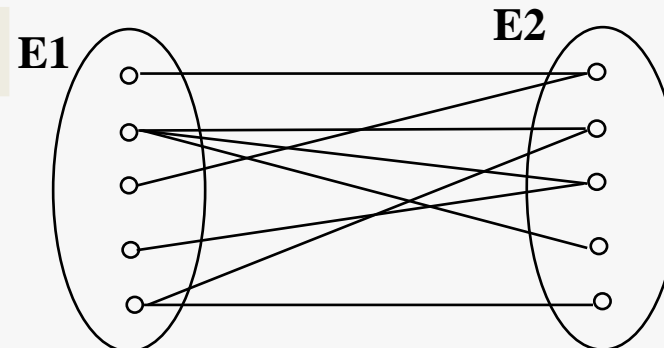
One-to-Many

1:M



Many-to-Many

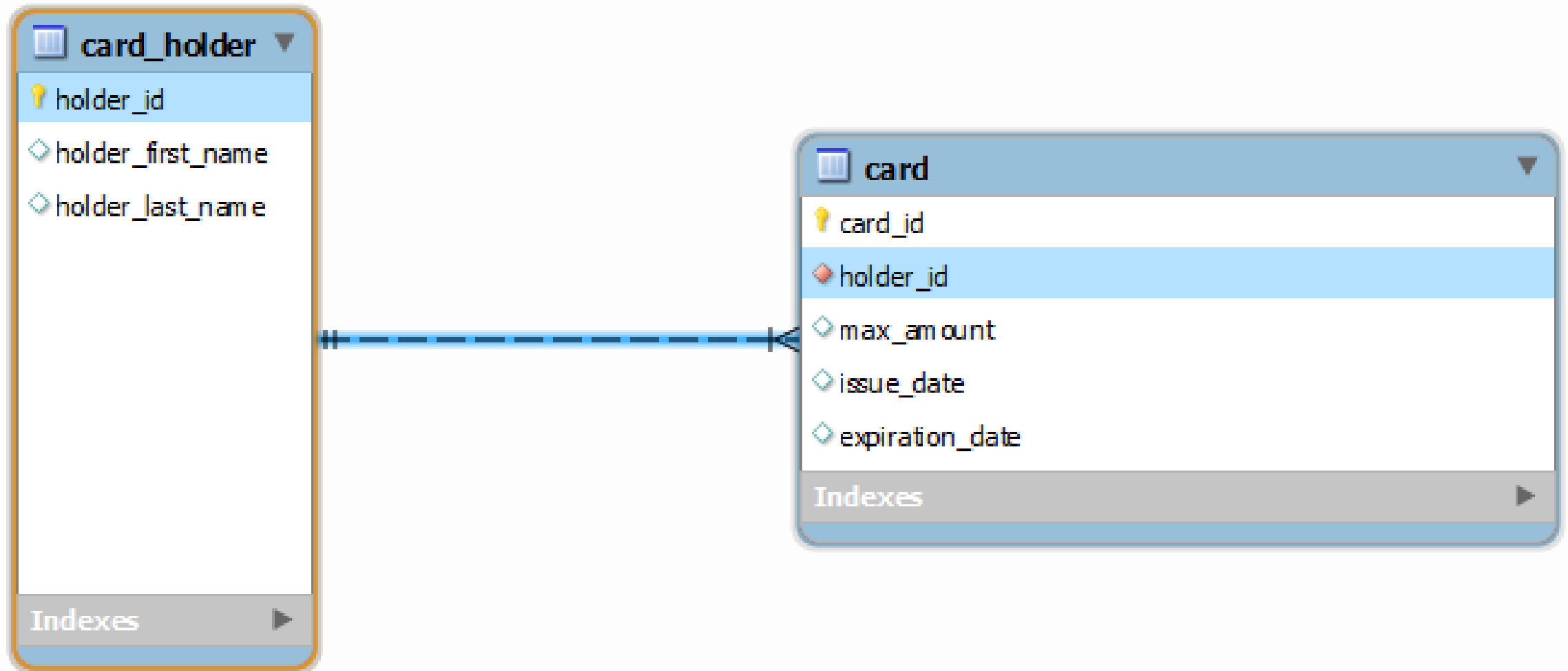
M:N



Modality

- As cardinality is the maximum number of connections between table rows (either one or many), **modality** is the least number of row connections! Modality also only has two options, 0 being the least or 1 being the least.

Example – Credit Card



Modality



Modality

Cardinality

Easiest Way to Remember

nullable



not nullable



Modality - Possibilities

0 or 1



Atleast 0



Only 1



Atleast 1



Example

One card holder can have:

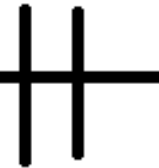
0 or 1



Credit card

One card holder can have:

Only 1



Credit card

One card holder can have:

Atleast 0



Credit card

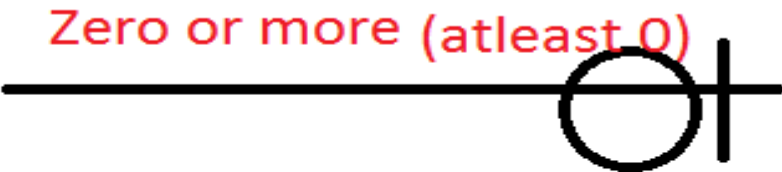
One card holder can have:

Atleast 1



Credit card

How to Read Relationships

Read left to right: One person can take:  Zero or more (atleast 0) Classes

Read right to left: student  atleast 1 One class can have



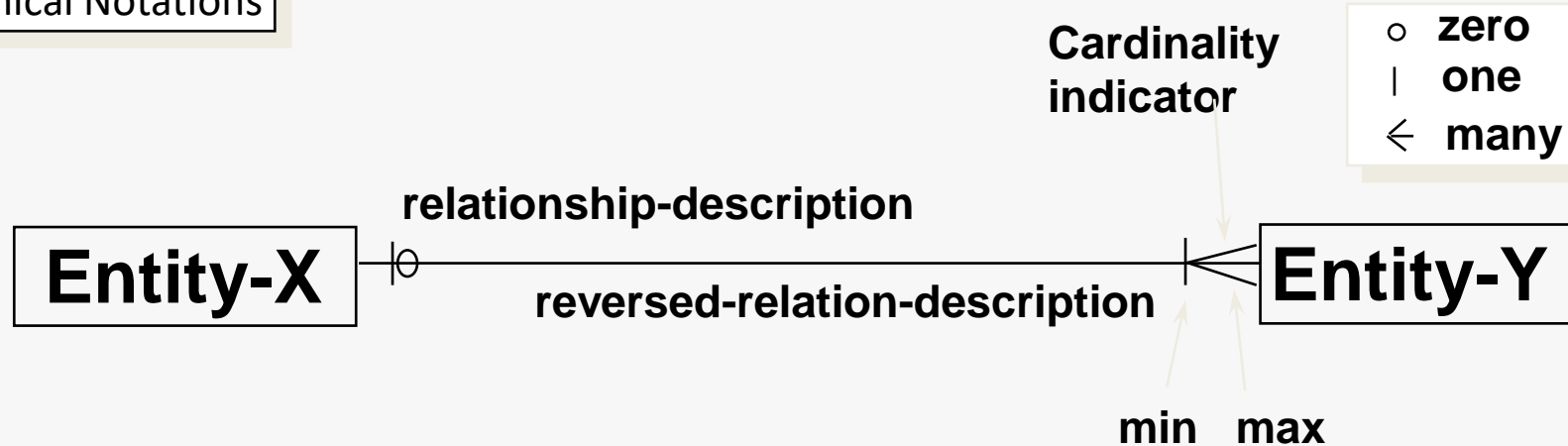
Student  Class

Relationship Cardinality

- The number of Entity Instances involved in the Relationship Instances Grouping in a Relationship Type.
- Three Forms of Cardinality
 1. One-to-one (1:1)
DEPARTMENT has MANAGER
Each DEPARTMENT has one and only one MANAGER
Each MANAGER manages one and only one DEPARTMENT
 2. One-to-many (1:m)
CUSTOMER places ORDER
Each CUSTOMER sometimes (95%) place one or more ORDERs
Each ORDER always is placed by exactly one CUSTOMER
 3. Many-to-many (m:n)
INSTRUCTOR teaches COURSE
Each INSTRUCTOR teaches zero, one, or more COURSEs
Each COURSE is taught by maybe one or more INSTRUCTORs

Entity Relationship Diagram (ERD): Notations

Graphical Notations



Translate into two structured statements

Each Entity-X relationship-description cardinality-indicator (one-or-many) Entity-Y
Each Entity-Y reversed-relationship-description (zero-or-one) Entity-Y

Example

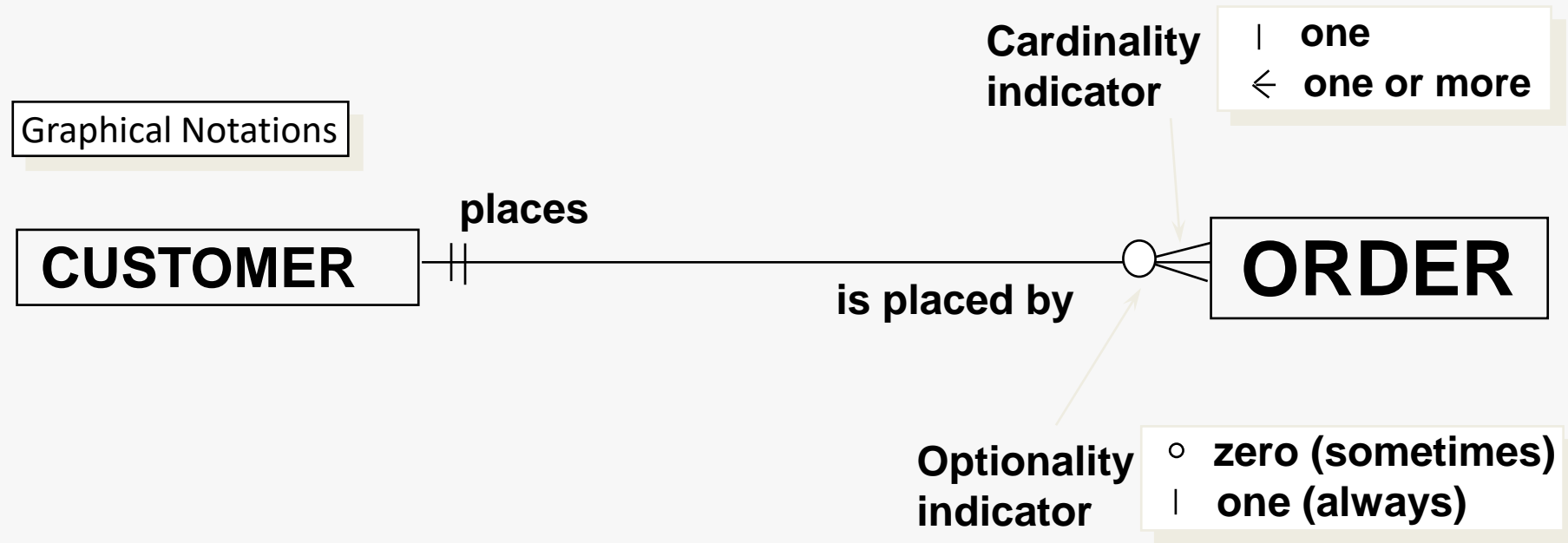


Optionality of Relationship Memberships

- Whether all entity instances of both entity types need to participate in relationship pairing.
- Optionality:
 - Mandatory
 - Optional
- Example:
 - CUSTOMER membership is optional
 - ORDER membership is mandatory



Relationship Statements

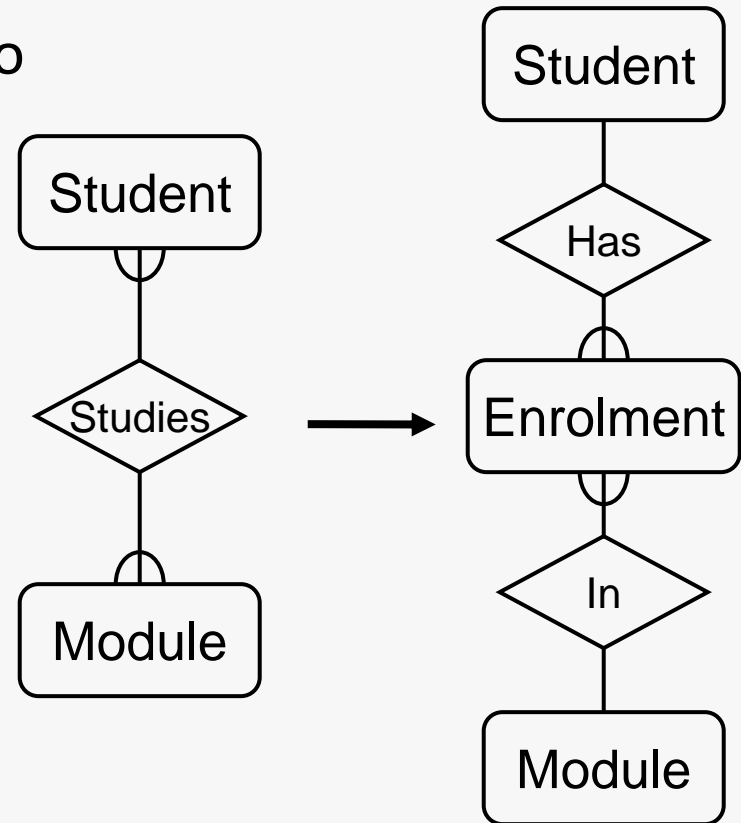


Each Entity X optionality relationship cardinality Entity Y

**Each CUSTOMER sometimes places one or more ORDER.
Each ORDER always is placed by one CUSTOMER.**

Removing M:M Relationships

- Many to many relationships are difficult to represent
- We can split a many to many relationship into two one to many relationships
- An entity represents the M:M relationship



Paring (Relationship Instance)

- Relationship paring is a pair of Entity Instances of two Entity Types associated by a Relationship Type between these two Entity Types.

Entity Types	Entity Instance
Student	Student#1 Student#2
Course	Course#A Course#B Course#C Course#D

Relationship	Relationship Paring
Student takes Course	Student#1 takes Course#A Student#1 takes Course#B Student#1 takes Course#D Student#2 takes Course#A Student#2 takes Course#C Student#2 takes Course#D

Relationship Instances Grouping

- Definition: A collection of pairings of a Relationship Membership in which an Entity Instance is involved.
- Examples:
 - Student#1 takes Course#A, #B, and #D
 - Student#2 takes Course#A, #C, and #D
 - Course#A is taken by Student#1 and Student#2

Attributes

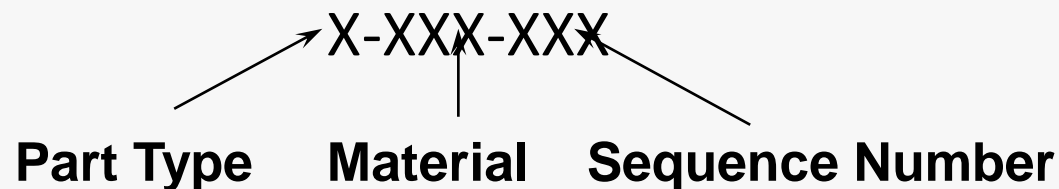
- Definition
 - Characteristics that could be used to describe Entity Types and Relationship Types. However, in IE, relationship types are not allowed to have attributes.
- Naming Conventions:
 - Names that have business meaning
 - Don't use abbreviation or possessive case, e.g., PN and Customer's name
 - Don't include entity type name because IEF will prefix the attribute name with entity type name automatically
 - Use standard format:

<u>Entity Type Name (Qualifiers)</u>	<u>Domain Name</u>
Customer	Name
Employee	Starting Date
- Examples
 - Customer has customer name, address, and telephone number
 - Product has quantity-on-hand, weight, volume, color, and name.
 - Employee has SSN, salary, and birthday.
 - Employee-works-for-project has percentage-of-time, starting-date.

Attribute Value

- Definition
 - Attribute Values are instances of Attributes used to describe specific Entity Instances
- Examples
 - Customer Number: 011334
 - Customer Name: Minder Chen
 - State: VA
 - Order Total: \$23,000
 - Sale tax: \$250
- An attribute of an entity type should have only one value at any given time. (No repeating group)
- Avoid using complex coding scheme for an attribute.

For example: PART Number:



Properties of Attributes

- Name
- Description
- Attribute Source Category: Basic, Derived, Designed
- Domain or data type: Text, Number, Date, Time, Timestamp
- Optionality: Mandatory or optional
- Length and/or precision
- Permitted Values (Legal Values)
 - Ranges
 - A set of values (Code Table)
- Default value or algorithm

Tools such as PowerBuilder has additional properties for table's columns called extended attributes

- | | |
|--------------------|-------------------------|
| – Validation Rule | – Column Heading |
| – Editing Format | – Form Label |
| – Reporting Format | – Code Table |

Attributes

*“Describe detail information about **an entity**”*

- **Entity:** Employee
- **Attributes:**
 - Employee-Name
 - Address (composite)
 - Phone Extension
 - Date-Of-Hire
 - Job-Skill-Code
 - Salary

Classes of attributes

- Simple attribute
- Composite attribute
- Derived attributes
- Multi-valued attribute

Simple/Composite attribute

- A **simple attribute** cannot be subdivided.
 - Examples: Age, Gender, and Marital status
- A **composite attribute** can be further subdivided to yield additional attributes.
 - Examples:
 - ADDRESS --→ Street, City, State, Zip
 - PHONE NUMBER --→ Area code, Exchange number

Derived attribute

- is not physically stored within the database
- instead, it is derived by using an algorithm.
 - Example 1: Late Charge of 2%
 - MS Access: $\text{InvoiceAmt} * 0.02$
 - Example 2: AGE can be derived from the date of birth and the current date.
 - MS Access: $\text{int}(\text{Date}() - \text{Emp_Dob})/365$

Multi-valued attributes

- can have many values.
 - Examples:
 - A person may have several college degrees.
 - A household may have several phones with different numbers
 - A car color

Example - "Movie Database"

- Entity:
 - Movie Star
- Attributes:
 - CNIC#: "123-45-6789"
(simple)
 - Cell Phone: "(661)123-4567, (661)234-5678"
(multi-valued)
 - Name: "Harrison Ford"
(composite)
Address: "123 Main Str., LA, CA"
(composite)
 - Gender: "Female"
(simple)
 - Age: 24
(derived)

Data Modeling Process

- List entity types
- Create relationships
 - Pick a central entity type
 - Work around the neighborhood
 - Add entity types to the diagram
 - Build relationships among them
 - Determine cardinalities of relationships
- Find/Create identifiers for each entity type
- Add attributes to the entity type in the data model
- Analyze and revise the data model

Example

A university consists of a number of departments. Each department offers several courses. A number of modules make up each course. Students enrol in a particular course and take modules towards the completion of that course. Each module is taught by a lecturer from the appropriate department, and each lecturer tutors a group of students

Example - Entities

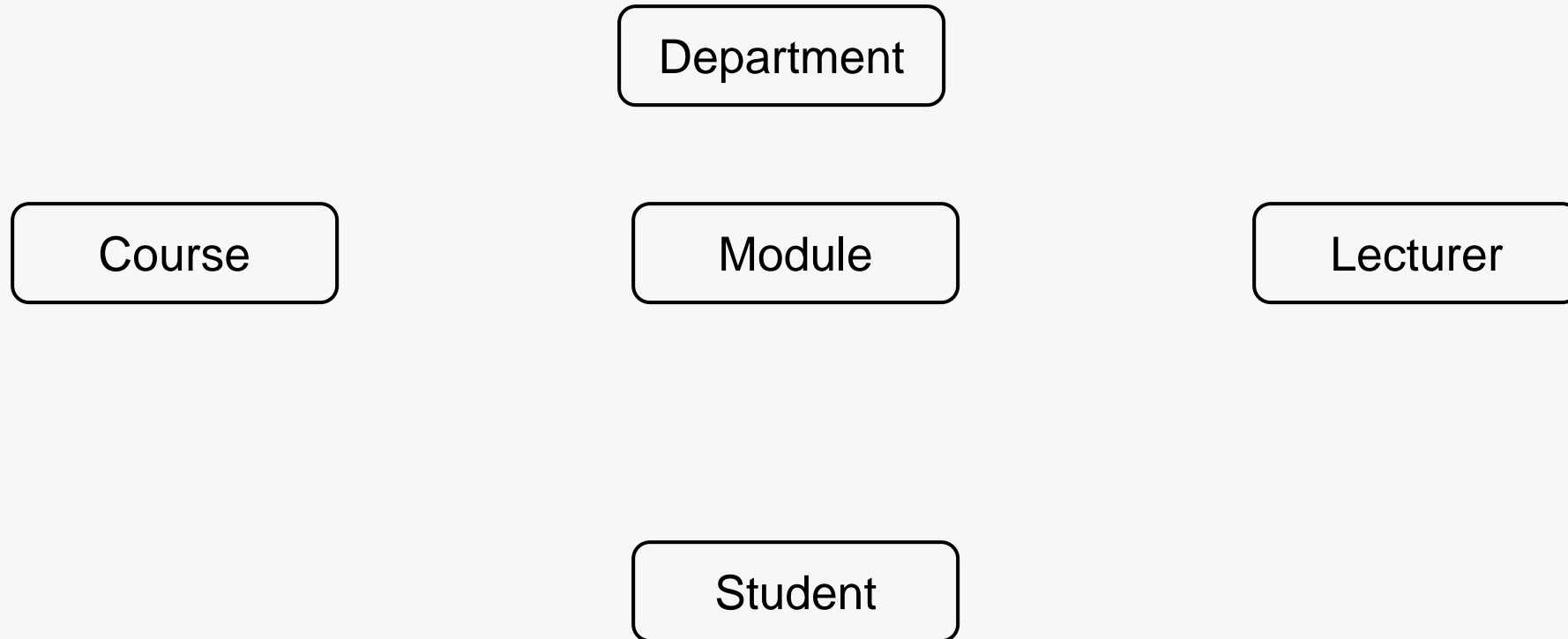
A university consists of a number of **departments**. Each department offers several **courses**. A number of **modules** make up each course. **Students** enrol in a particular course and take modules towards the completion of that course. Each module is taught by a **lecturer** from the appropriate department, and each lecturer tutors a group of students

Example - Relationships

- A university consists of a number of departments. Each department **offers** several courses. A number of modules **make up** each course. Students **enrol in** a particular course and **take** modules towards the completion of that course. Each module is **taught by** a lecturer **from the** appropriate department, and each lecturer **tutors** a group of students

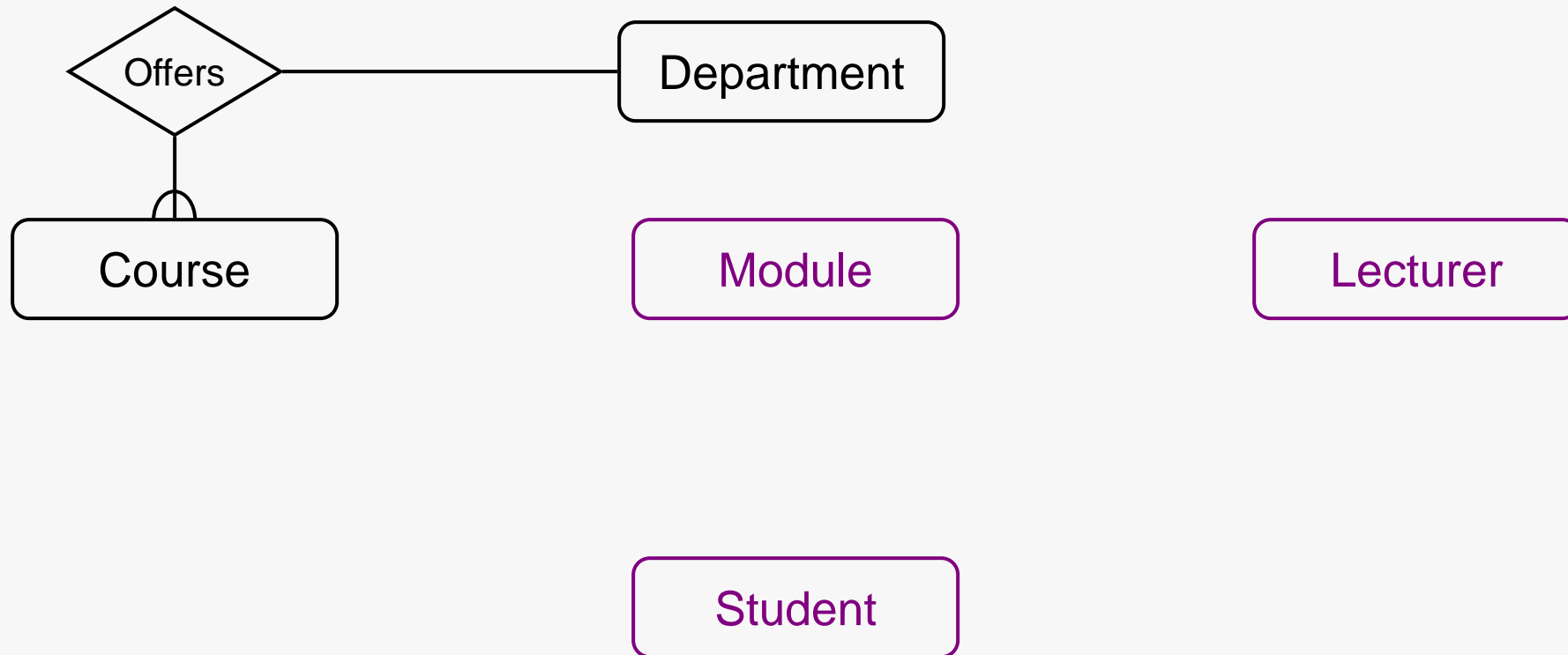
Example - E/R Diagram

Entities: Department, Course, Module, Lecturer, Student



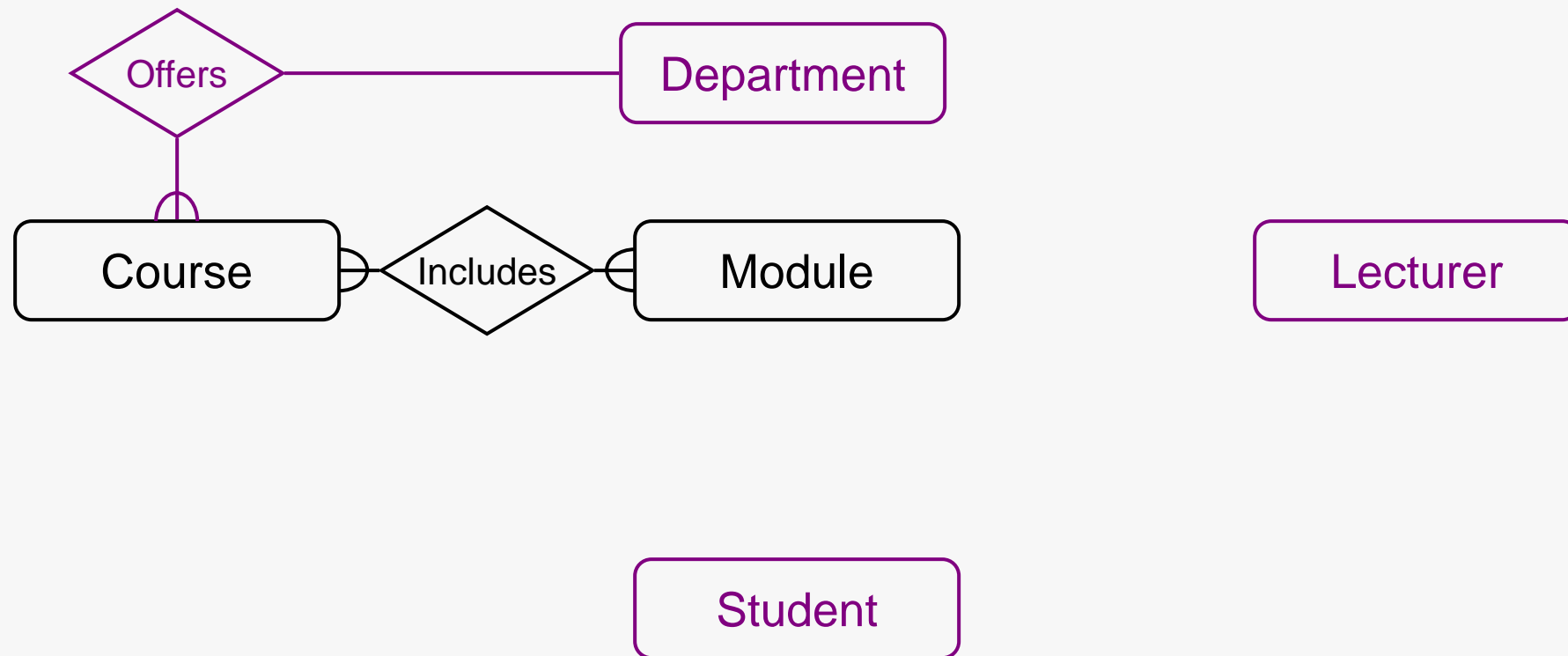
Example - E/R Diagram

Each department *offers* several courses



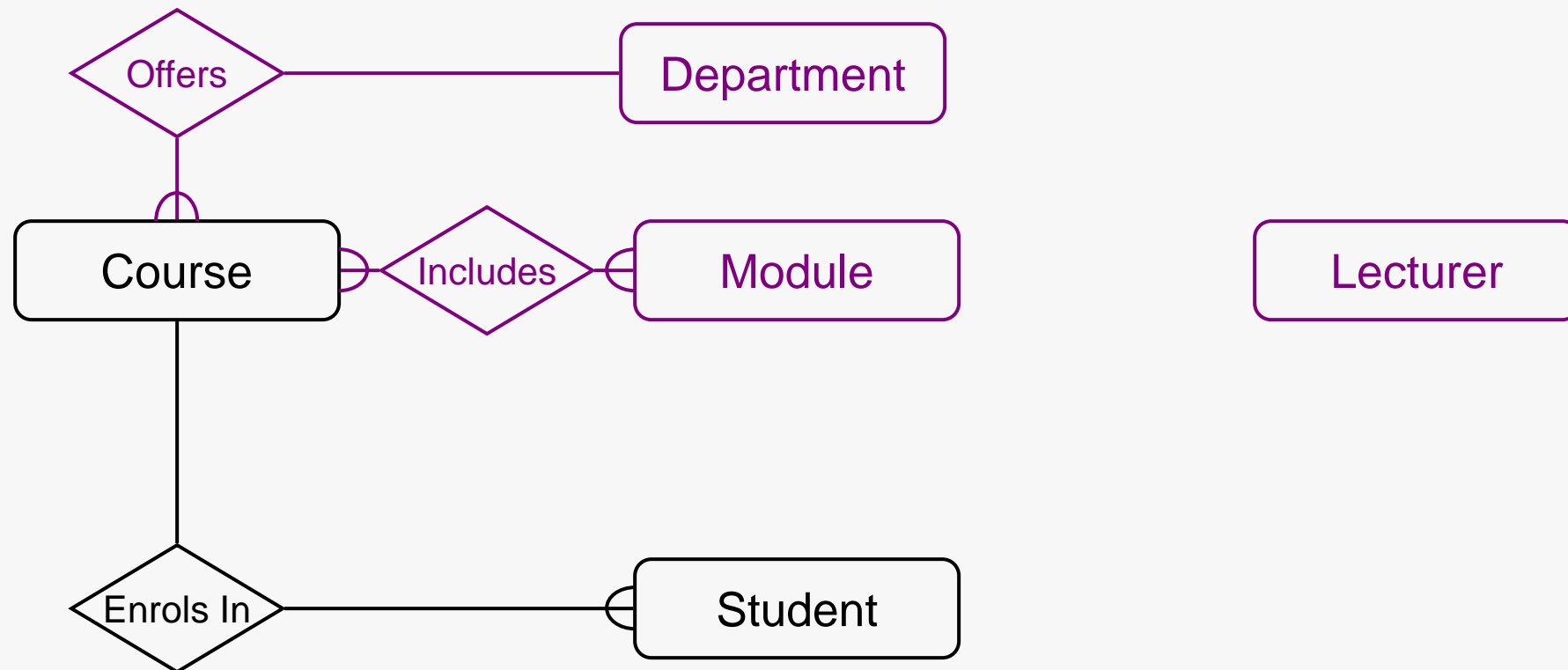
Example - E/R Diagram

A number of modules **make up** each courses

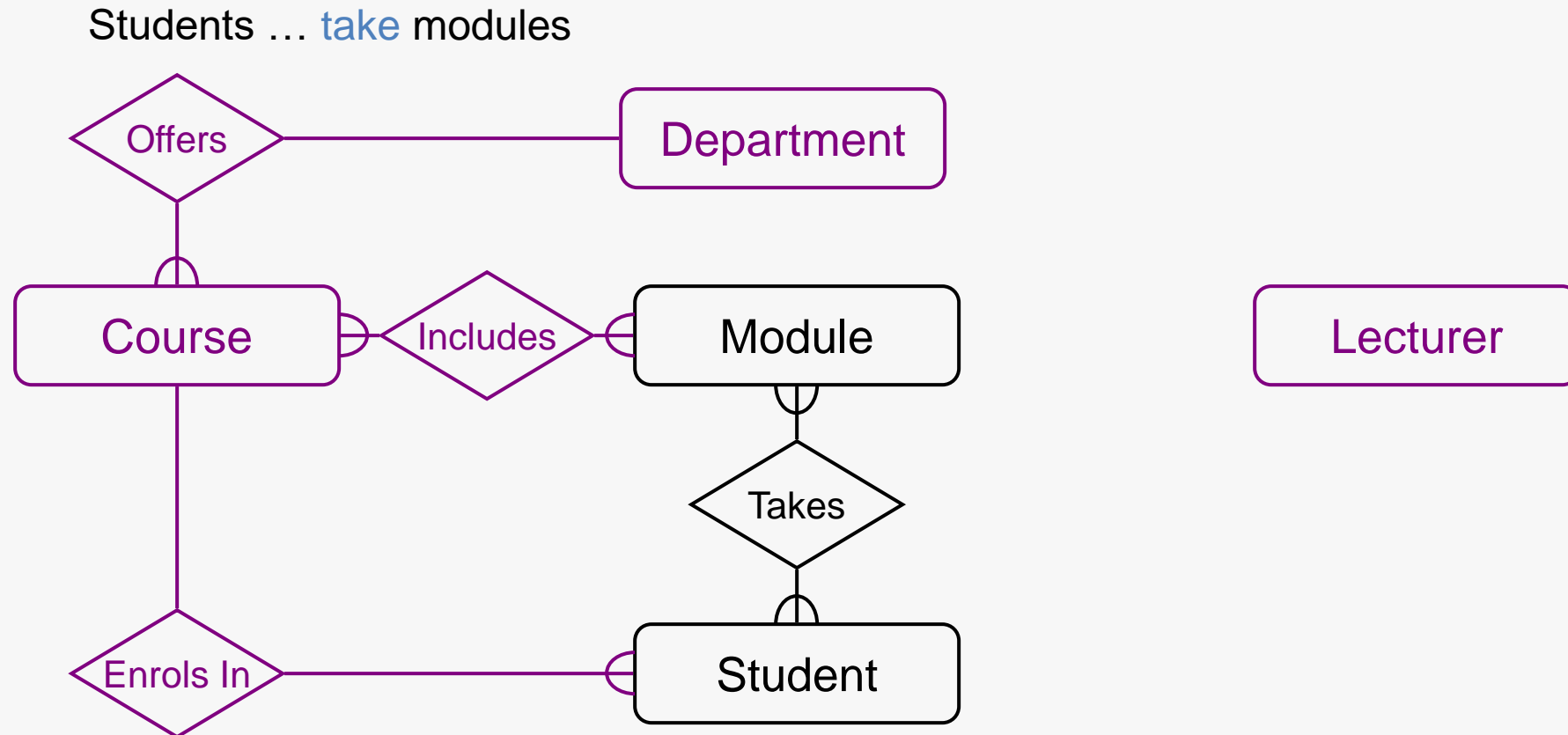


Example - E/R Diagram

Students **enrol in** a particular course

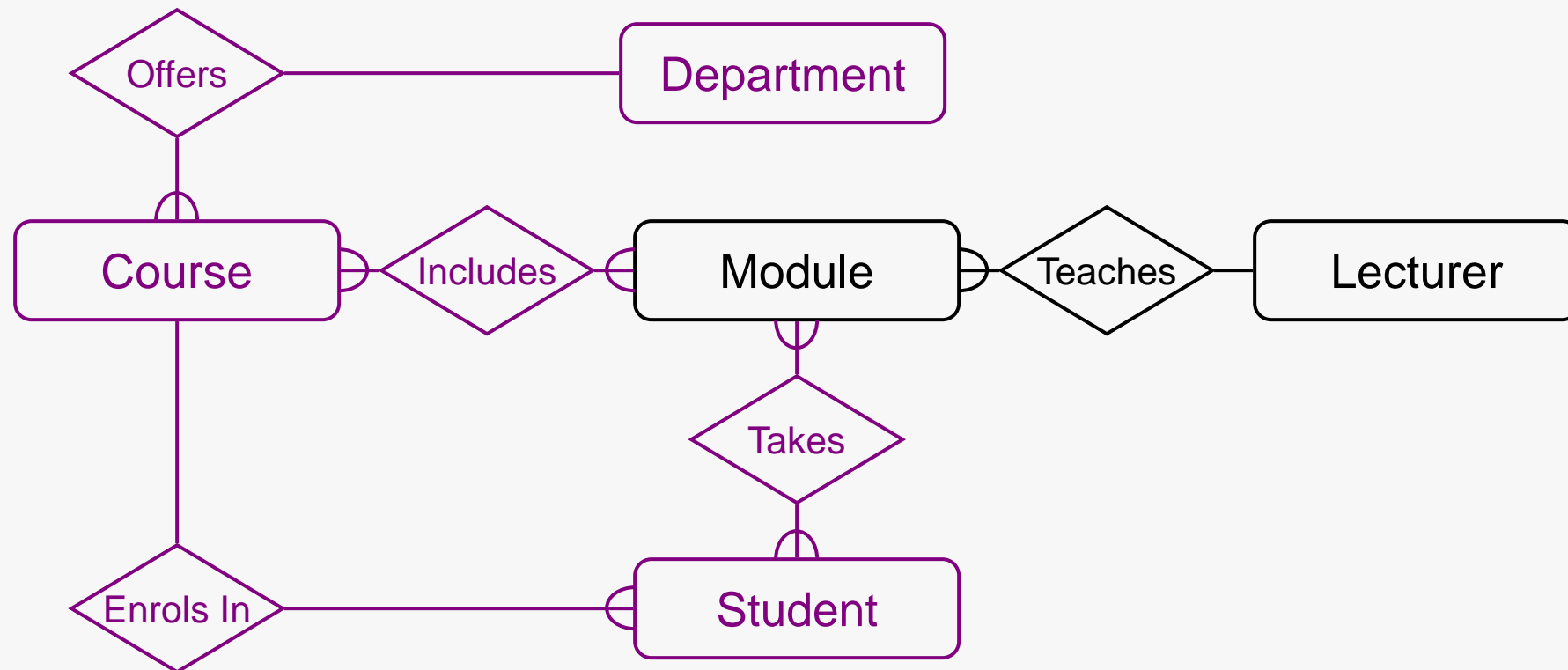


Example - E/R Diagram



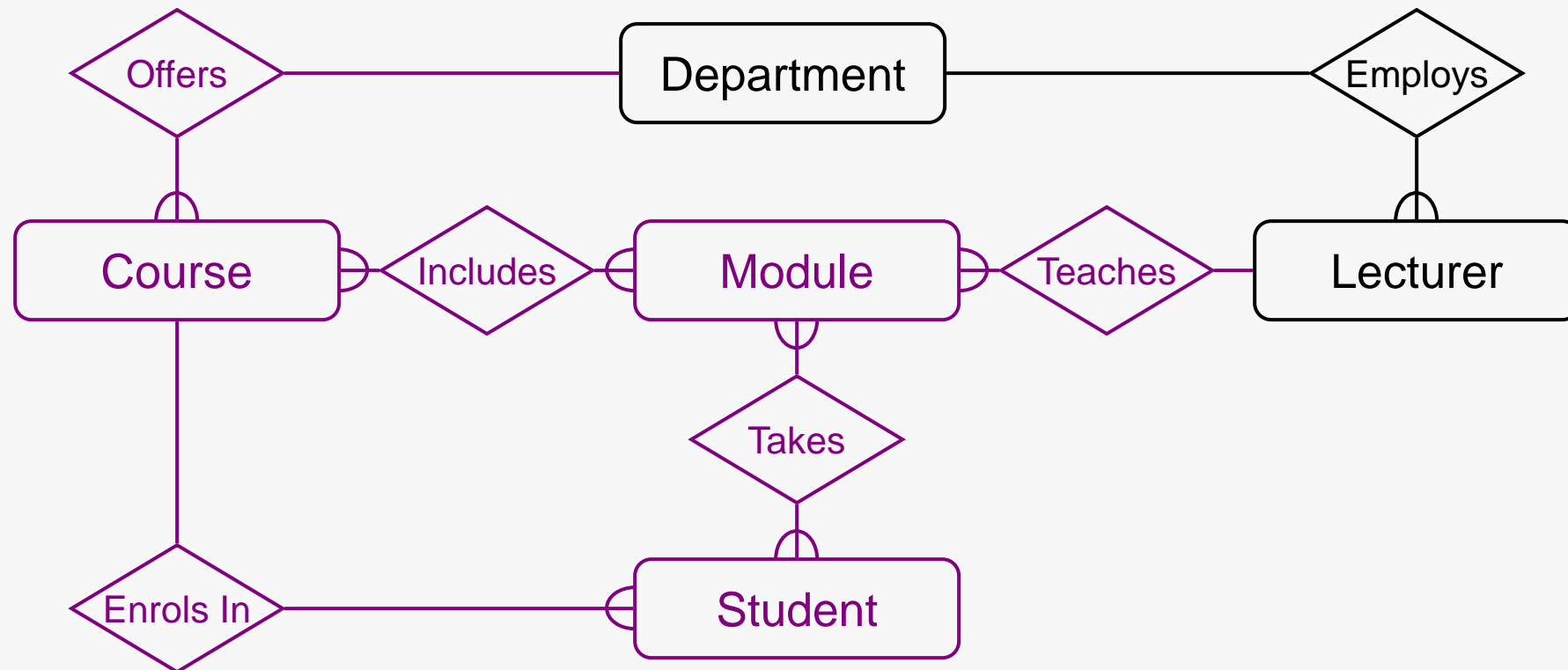
Example - E/R Diagram

Each module is taught by a lecturer



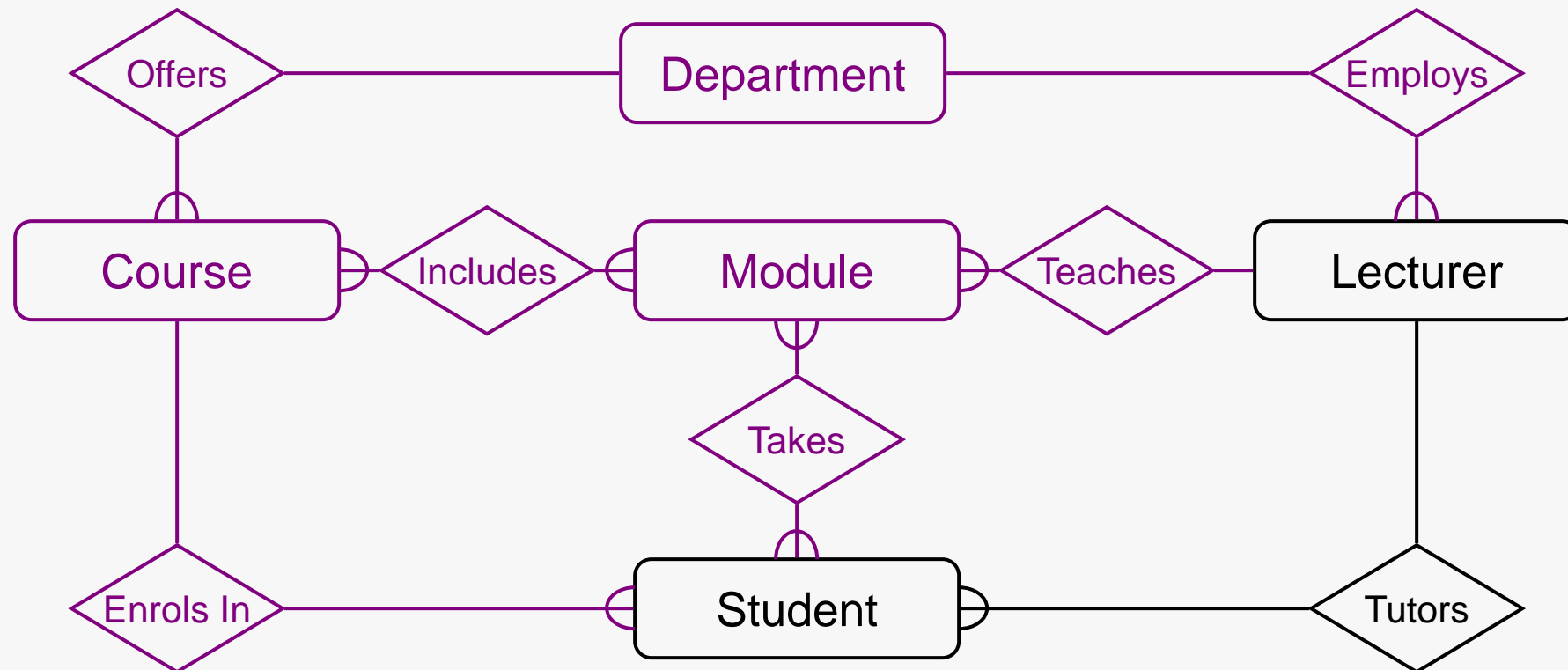
Example - E/R Diagram

a lecturer from the appropriate department

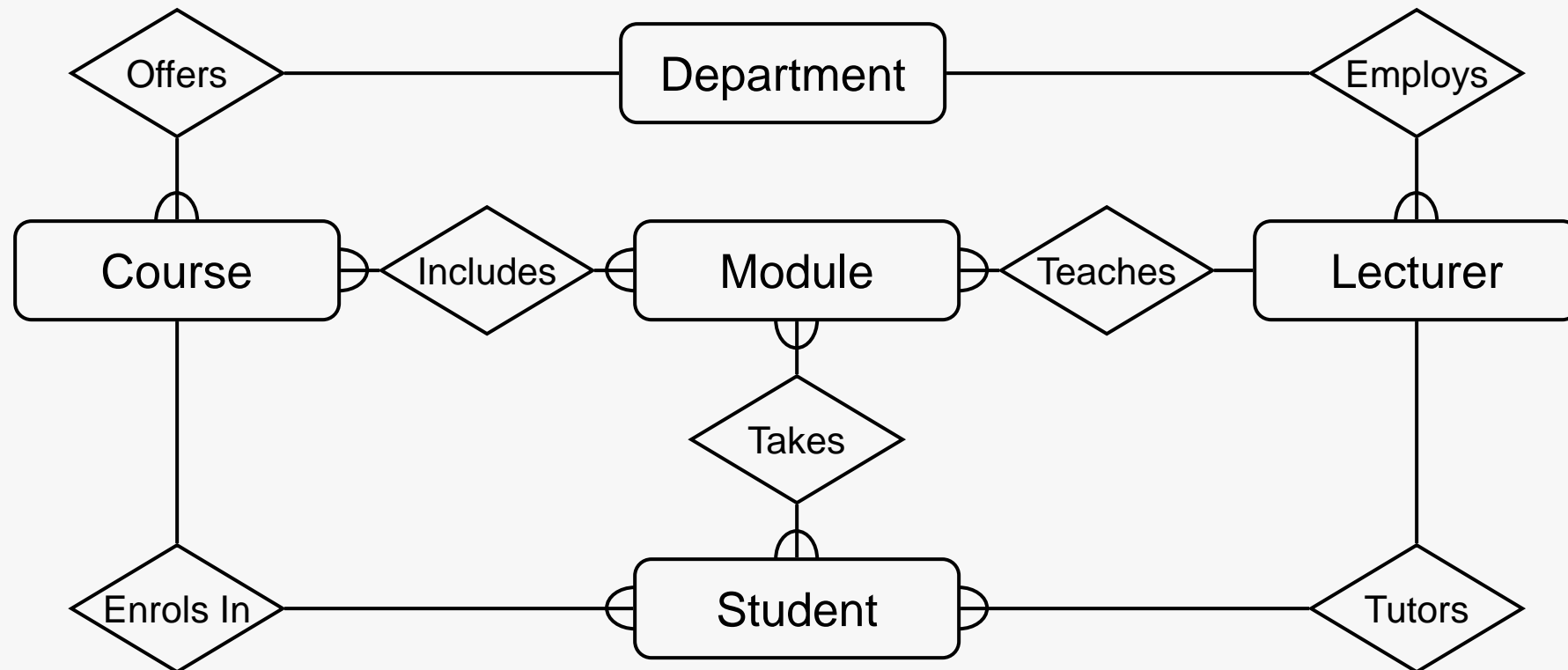


Example - E/R Diagram

each lecturer **tutors** a group of students



Example - E/R Diagram



Entities and Attributes

- Sometimes it is hard to tell if something should be an entity or an attribute
 - They both represent objects or facts about the world
 - They are both often represented by nouns in descriptions
- General guidelines
 - Entities can have attributes but attributes have no smaller parts
 - Entities can have relationships between them, but an attribute belongs to a single entity

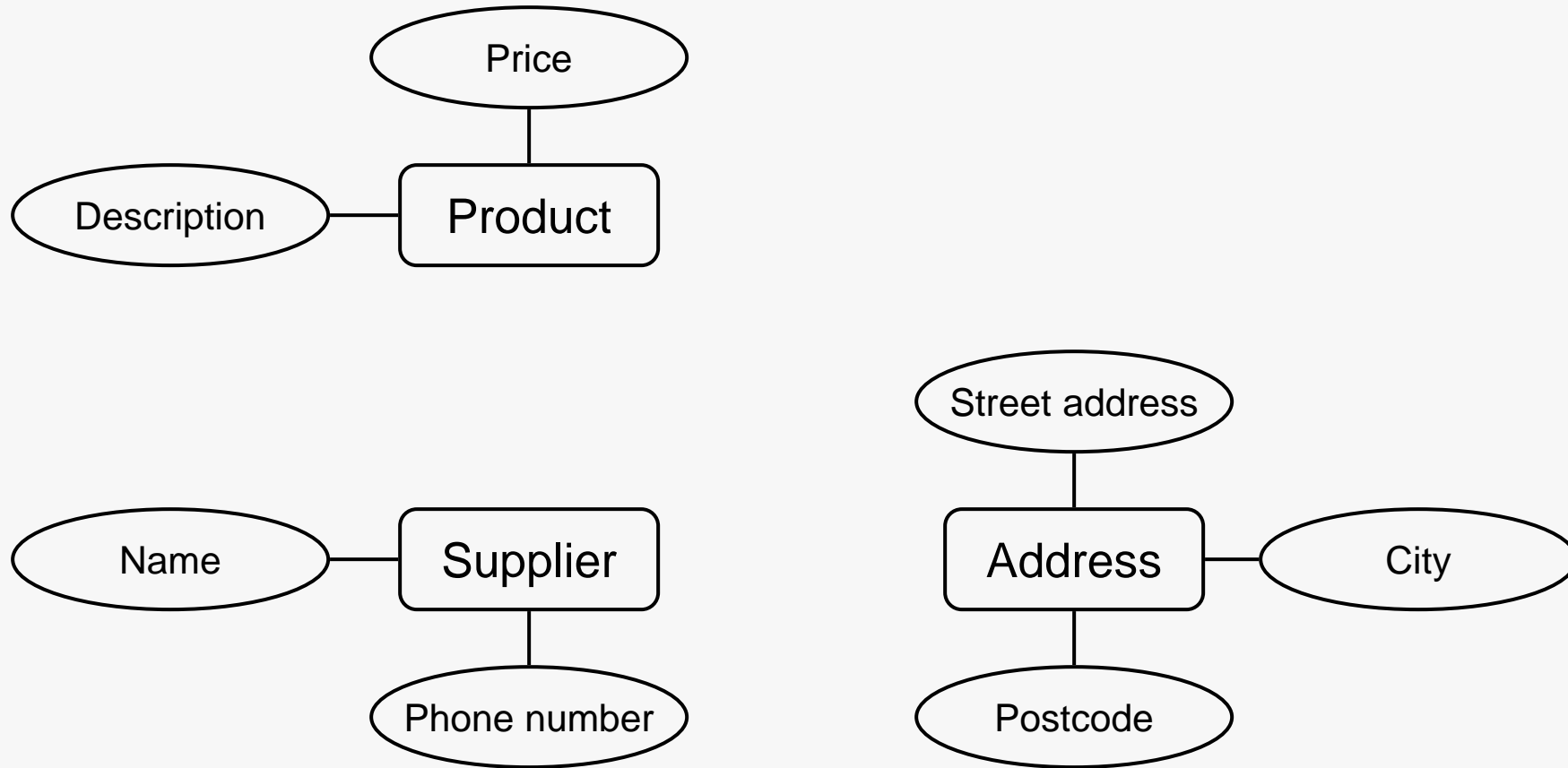
Example

We want to represent information about products in a database. Each product has a description, a price and a supplier. Suppliers have addresses, phone numbers, and names. Each address is made up of a street address, a city, and a postcode.

Example - Entities/Attributes

- Entities or attributes:
 - product
 - description
 - price
 - supplier
 - address
 - phone number
 - name
 - street address
 - city
 - postcode
- Products, suppliers, and addresses all have smaller parts so we can make them entities
- The others have no smaller parts and belong to a single entity

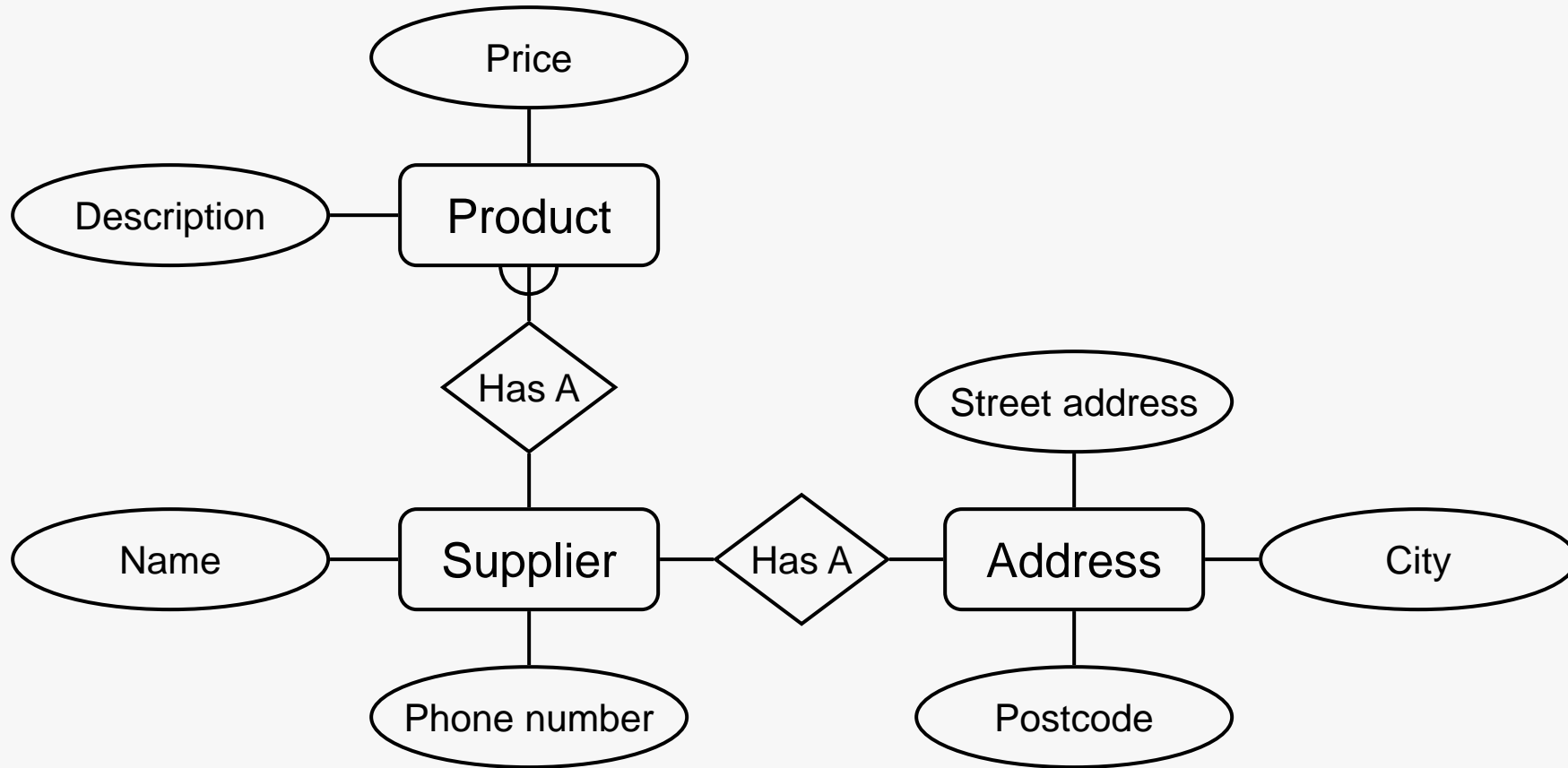
Example - E/R Diagram



Example - Relationships

- Each product has a supplier
 - Each product has a single supplier but there is nothing to stop a supplier supplying many products
 - A many to one relationship
- Each supplier has an address
 - A supplier has a single address
 - It does not seem sensible for two different suppliers to have the same address
 - A one to one relationship

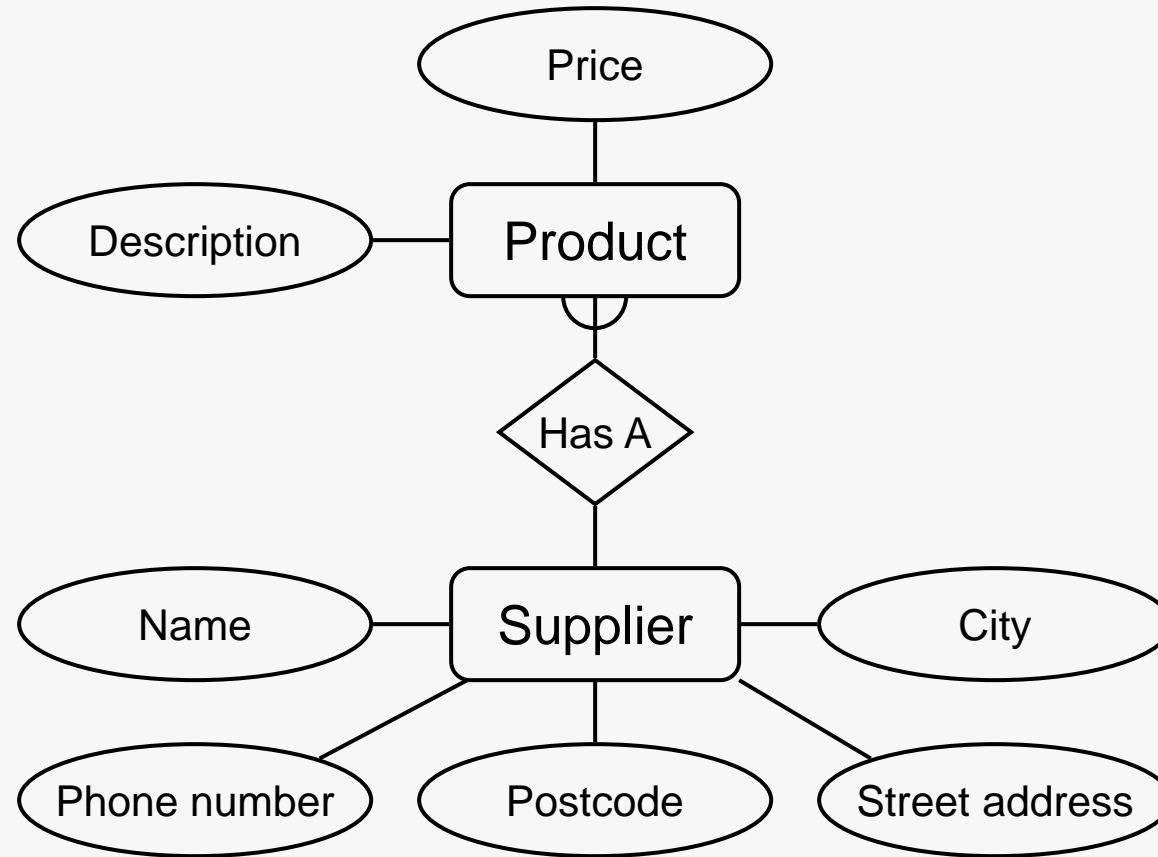
Example - E/R Diagram



One to One Relationships

- **Some** relationships between entities, A and B, **might** be redundant if
 - It is a 1:1 relationship between A and B
 - Every A is related to a B and every B is related to an A
- Example - the supplier-address relationship
 - Is one to one
 - Every supplier has an address
 - We don't need addresses that are not related to a supplier

Example - E/R Diagram

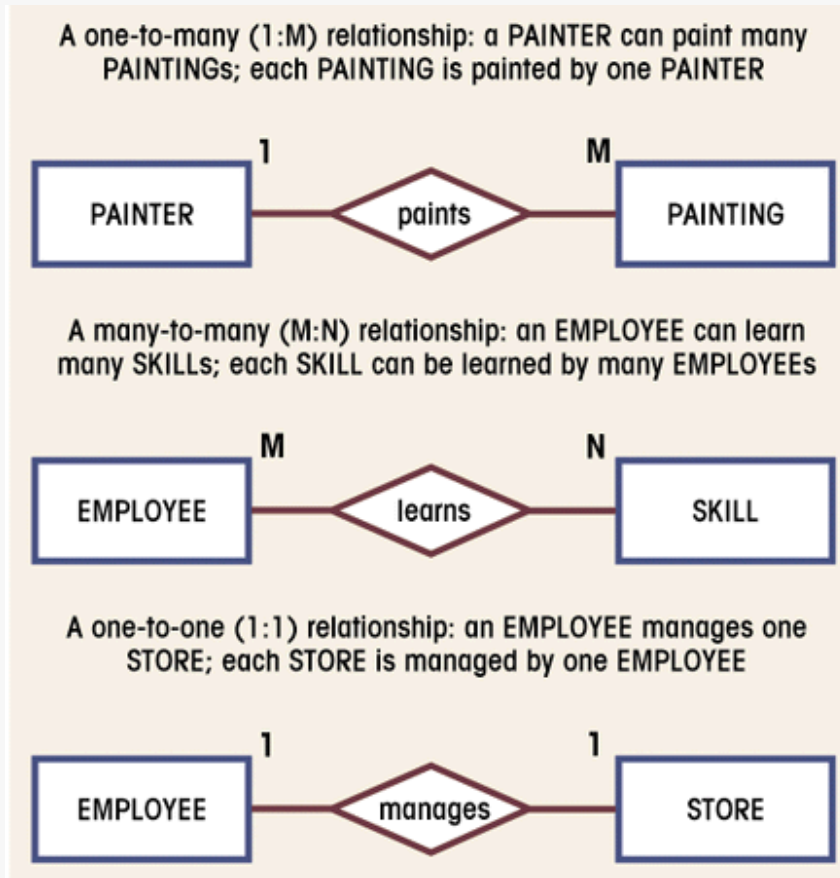


Making E/R Diagrams

- From a description of the requirements identify the
 - Entities
 - Attributes
 - Relationships
 - Cardinality ratios of the relationships
- Draw the E/R diagram and then
 - Look at one to one relationships as they might be redundant
 - Look at many to many relationships as they might need to be split into two one to many links

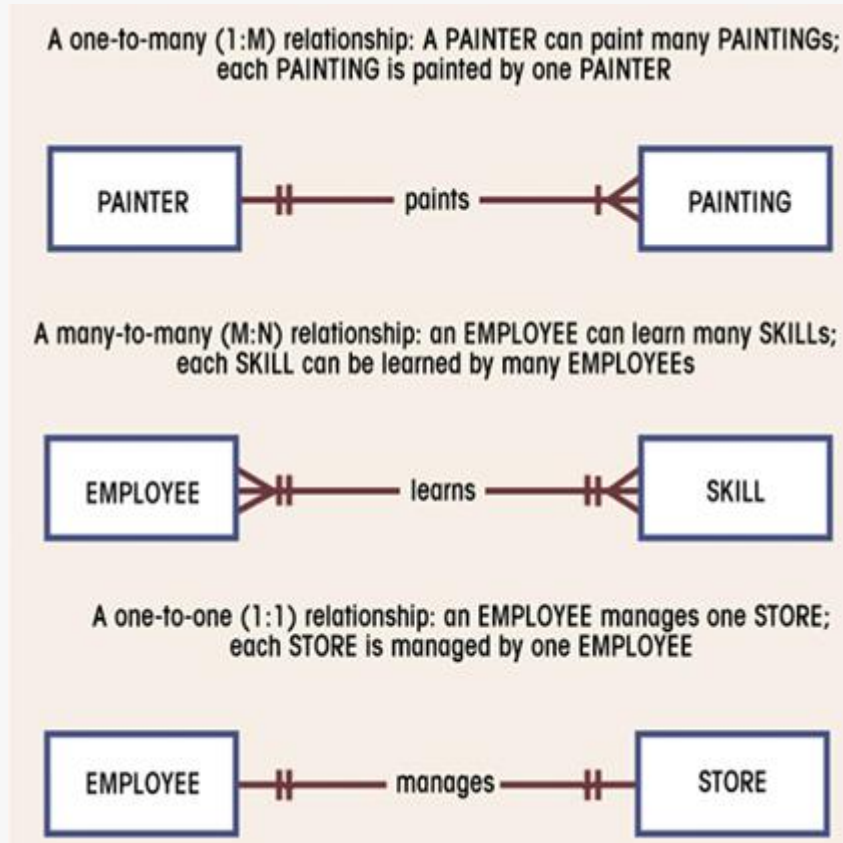
ER Diagram Notation

E-R Diagram: Chen Model

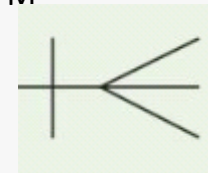


- **Entity**
 - represented by a rectangle with its **name** in capital letters.
- **Relationships**
 - represented by an active or passive **verb** inside the **diamond** that connects the related entities.
- **Connectivities**
 - i.e., types of relationship
 - written next to each entity box.

E-R Diagram: Crow's Foot Model



- **Entity**
 - represented by a rectangle with its name in capital letters.
- **Relationships**
 - represented by an active or passive verb that connects the related entities.
- **Connectivities**
 - indicated by symbols next to entities.
 - 2 vertical lines for 1
 - “crow’s foot” for M
- It is good to use for many part



Crow's Foot Connections

One-to-One
Relationship

One-to-Many
Relationship

Many-to-Many
Relationship

One-to-One
Relationship

One-to-Many
Relationship

Many-to-Many
Relationship

One-to-One
Relationship

One-to-Many
Relationship

Many-to-Many
Relationship

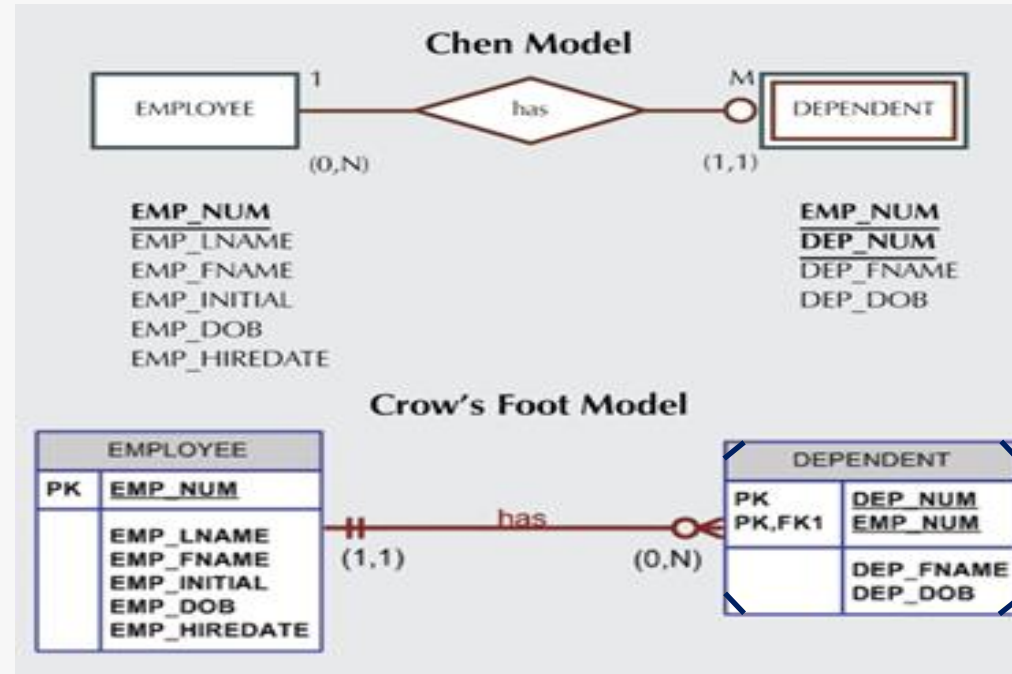
One-to-One
Relationship

One-to-Many
Relationship

Many-to-Many
Relationship

Symbol	Meaning
	One — Mandatory
	Many — Mandatory
	One — Optional
	Many — Optional

Relationship: Weak Entities



Database Systems: Design, Implementation, & Management: Rob & Coronel

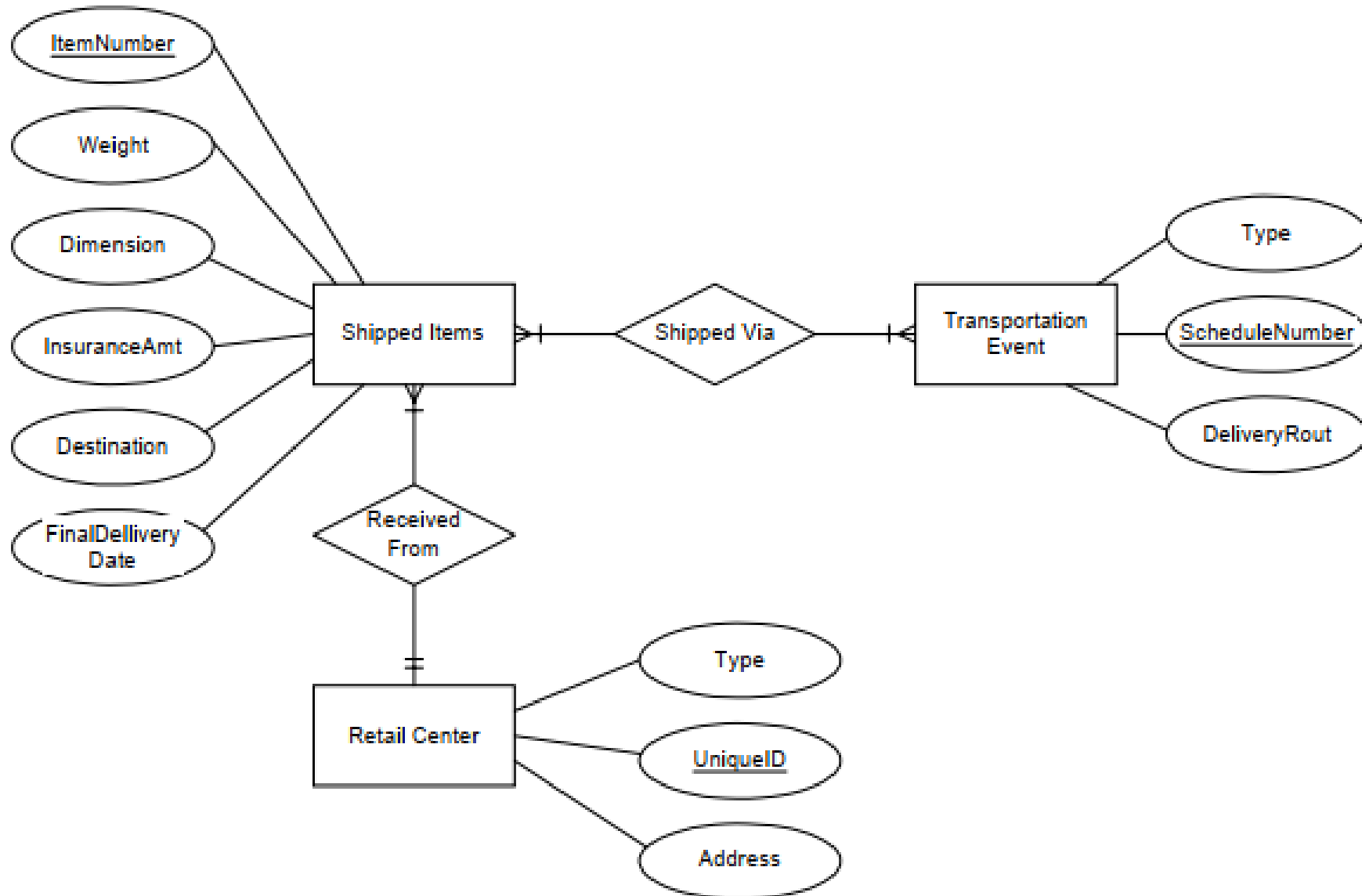
Strong vs. Weak entities

- Strong Entity = existence-independent entity
- Weak Entity
 - ✓ existence-dependent entity in a strong relationship
 - ✓ inherits all or part of its primary key from parent entity
 - ✓ entity w/ clipped corners in CF model, double-walled in Chen model

Exercise

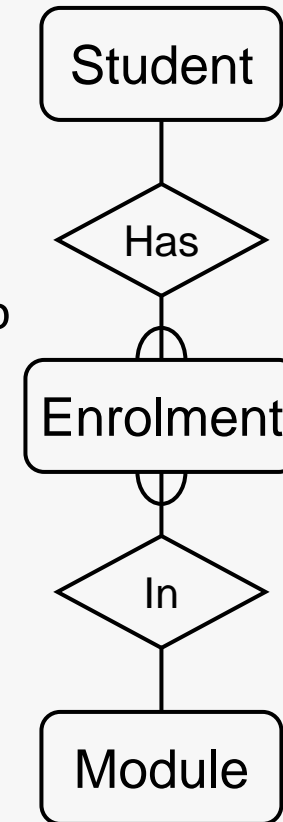
UPS prides itself on having up-to-date information on the processing and current location of each shipped item. To do this, UPS relies on a company-wide information system. Shipped items are the heart of the UPS product tracking information system. Shipped items can be characterized by item number (unique), weight, dimensions, insurance amount, destination, and final delivery date. Shipped items are received into the UPS system at a single retail center. Retail centers are characterized by their type, uniqueID, and address. Shipped items make their way to their destination via one or more standard UPS transportation events (i.e., flights, truck deliveries). These transportation events are characterized by a unique scheduleNumber, a type (e.g, flight, truck), and a deliveryRoute.

Exercise – Solution



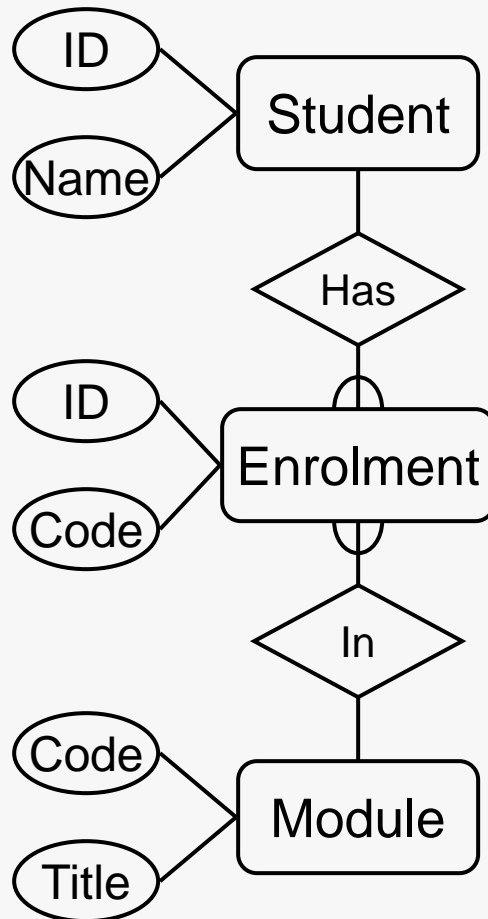
Debugging Designs

- With a bit of practice E/R diagrams can be used to plan queries
 - You can look at the diagram and figure out how to find useful information
 - If you can't find the information you need, you may need to change the design



How can you find a list of students who are enrolled in Database systems?

Debugging Designs



(3) For each instance of Enrolment in the result of (2) find the corresponding Student








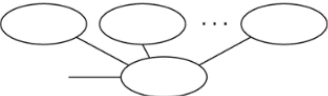



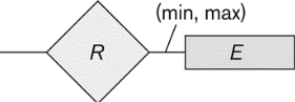
(2) Find instances of the Enrolment entity with the same Code as the result of (1)

(1) Find the instance of the Module entity with title 'Database Systems'

Summary of notation for ER diagrams

Figure 3.14

Summary of the notation for ER diagrams.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1: N for $E_1:E_2$ in R
	Structural Constraint (min, max) on Participation of E in R

Other alternative diagrammatic notations

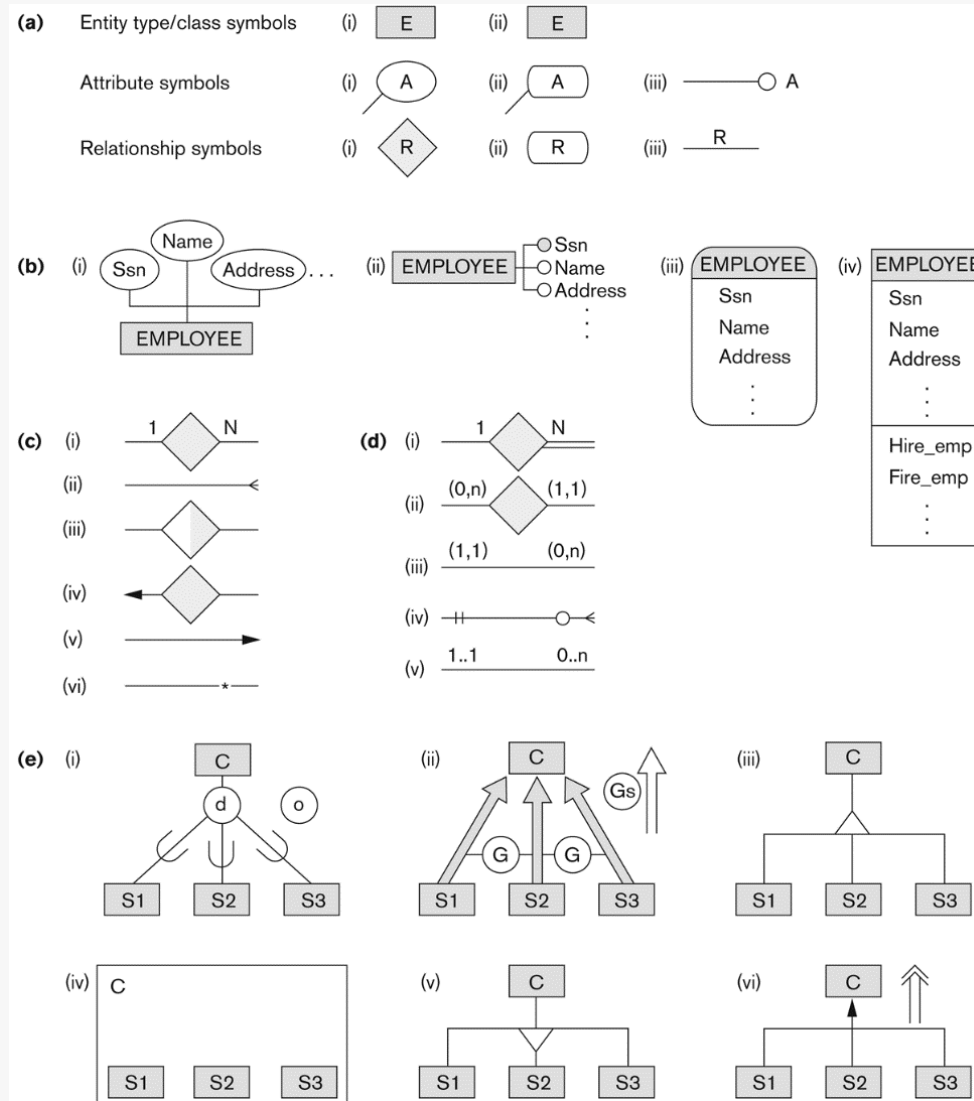


Figure A.1

Alternative notations. (a) Symbols for entity type/class, attribute, and relationship. (b) Displaying attributes. (c) Displaying cardinality ratios. (d) Various (min, max) notations. (e) Notations for displaying specialization/generalization.

