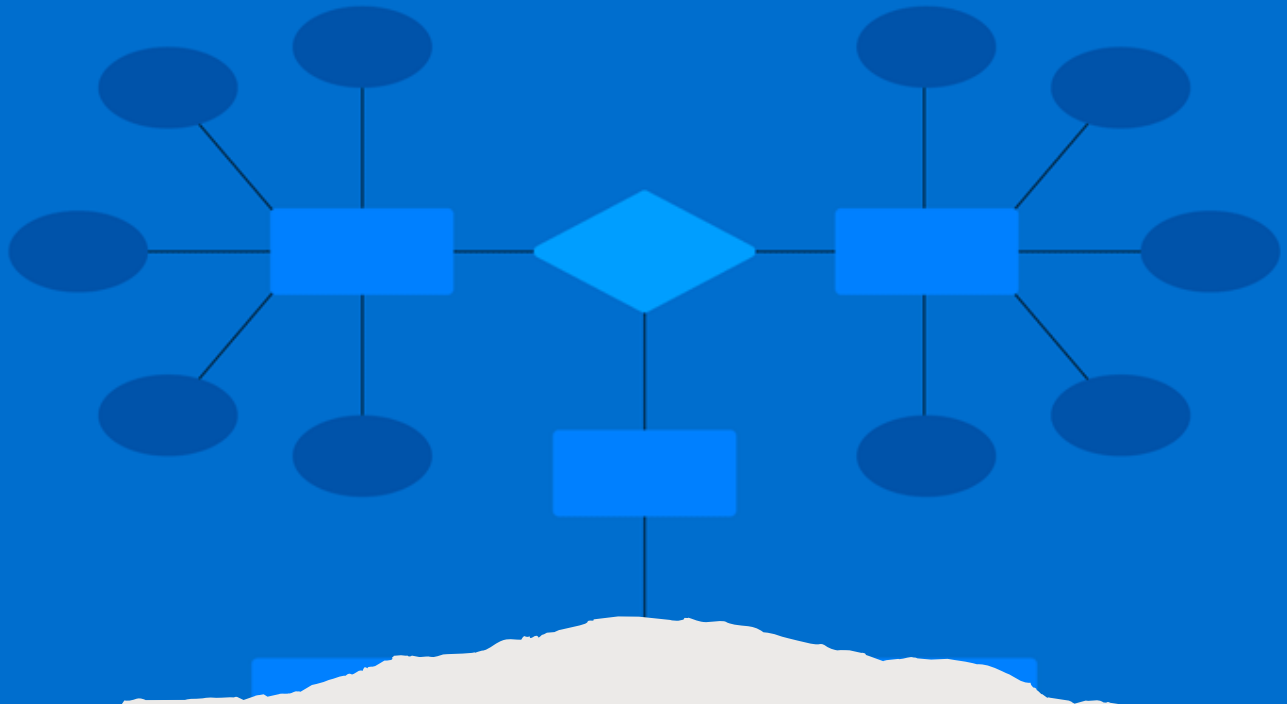
ENTITY RELATIONSHIP MODEL



COM106: Introduction to Databases

Introduction to Entity-Relationship Modelling

Introduction to Entity-Relationship Modelling

See Chapter 12 of Connolly & Beggs (6th ed)
Or similar chapters in other textbooks.

Taking Stock:

So far in this module we have:

- Studied different database models (focusing on the Relational Model),
- Implemented a database using SQL server 2017 and
- Learned how to use SQL to manipulate and query an existing database.

In this section we will cover:

- An introduction to **Entity-Relationship (ER) Modelling**; what is and how it is used in database design.
- The basic concepts of ER modelling - **entities**, **relationships** and **attributes**
- ER Diagrams, Symbols and Notations.
- Using ER modelling in database design; examples, issues, identifying and resolving problems.
- A taster of some slightly more advanced concepts in ER modelling.
- Translating an ER model into a **Relational Schema** (a set of linked tables)

Introduction to ER Modelling

Database Development Stages

Process	Tasks Involved	End Result
Data Investigation	Identification of nature and use of data	Outline Data Dictionary
Data Modelling	Shaping real world facts into real world concepts	Generalised Conceptual Model
Database Conceptual Design	Mapping conceptual model to specific data model	DBMS-specific and Conceptual External Schema
Database Implementation	Structuring the actual data in the physical database	Internal Schema and Database
Database Security, Monitoring and Tuning	Determining and implementing security measures. Monitoring database usage and Restructuring/Reorganising Database	Optimised and Secure Database

The Data Investigation and Data Modelling stages are **independent of the eventual implementation** and of the DBMS used to realise it.

Investigation and modelling are **critically important**. Errors at this stage that find their way into the developed system can be notoriously **difficult to correct**.

Introduction to ER Modelling

Data Investigation

Process	Tasks Involved	End Result
Data Investigation	***Identification of nature and use of data***	***Outline Data Dictionary***
- - - - -	- - - - -	- - - - -

Within the constraints of the **business scenario**, the database designer must decide:

- What are the **entities**, and the **relationships** among these entities in the application?
- What information about these entities and relationships should we store in the database - **attributes**?
- What are the **integrity constraints** or **business rules** that hold?

The database designer must investigate the current system to **understand the requirements**

- Interview customers, use existing system forms/reports/common queries, etc, etc.
- Determine the **scope** of the required system – what's important and what's not.

Output :

An outline **Data Dictionary**: Set of **attributes** with sample values , source, usage etc.

Attributes can be grouped into **entity types** (i.e. what is the basic object that an attribute is describing)

Introduction to ER Modelling

Data Modelling

Process	Tasks Involved	End Result
- - - - -	- - - - -	- - - - -
Data Modelling	***Shaping real world facts into real world concepts***	***Generalised Conceptual Model***
- - - - -	- - - - -	- - - - -

Produce a single **conceptual model** of the system using:

Entity Relationship Modelling
top down approach (generalisation of entities/relationships)

Normalisation *(We will look at this later in the module)*
bottom up approach (linked to relational data model)

The next stage of the database design process involves transforming the **conceptual model** into a **data model** ready for implementation in a particular DBMS (e.g., relational, object-oriented, etc).

In our case, we will end up with a **Relational Schema**, ready for implementation in a Relational DBMS (**RDBMS**).

Introduction to ER Modelling

ER Modelling and Normalisation

The Conceptual Model developed must be **relevant**, and **strictly limited** to the application.

Consider a Private Hospital:

a database developed for **billing patients** would not need to include information about **allergies**.

Different applications can have different needs, and different perspectives – even to model the ***same*** object

Billing Department:

PATIENT(id, name, insurance, address)

VISIT(patientId, procedure, date, charge)

Inpatient:

PATIENT(id,name,age,address)

ALLERGIES(id,allergies)

PERSCRIPTION(patientId,date,medicine)

The process of designing a database involves:

1. Creating an **ER Diagram** based on the results of a data investigation (**ER Modelling**)
2. Using the mapping (translation rules) to create a **relational schema** (a set of linked tables)
3. Checking that each table is correctly designed (i.e. in **3rd Normal Form**) which may involve decomposition (splitting into more tables)

Steps 1 & 2 are part of **ER modelling**; 3 is **Normalisation**

Introduction to ER Modelling

ER Modelling - A Brief Overview

Entities

The basic concept of the ER Model is the **Entity Type**.

An **entity type** is a group of objects having the same properties recognised by an enterprise as:

- important
- uniquely identifiable
- has more than one attribute to describe it.

Can have a physical or 'real' existence – PART, SUPPLIER
Or a conceptual or 'abstract' existence – SALE, INSPECTION

Relevant to the given problem. Information on VEHICLE may or may not be relevant in a database concerned with an enterprise's staff.

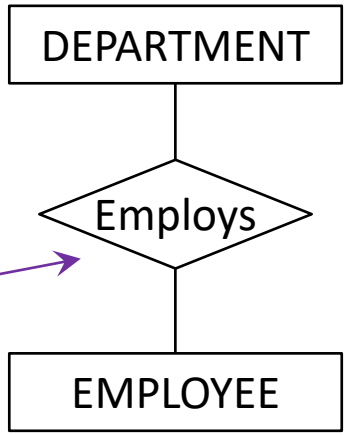
An attribute is a property of an entity type (e.g. surname)

An **Entity Occurrence** is a uniquely identifiable occurrence of an entity type.

Relationships

A **relationship** is a meaningful association between two (or more) entities.

e.g. The **EMPLOYS** relationship links DEPARTMENT & EMPLOYEE entity types



Entity Relational (ER) Diagrams

ENTITY TYPE: Represented by labelled rectangular box

RELATIONSHIP: Represented by labelled diamond box between pair of entities

Introduction to ER Modelling

Entities

How do we determine what the entities should be?

Start **by understanding the application** required.

- By talking with the customer/users/stakeholders to understand the application required,
- By studying the existing system/process, forms, sorts of information required, etc

Some Examples:

People: staff, clients, patients, members, owners, contacts, other individuals, . . .

Objects: stock items, real estate, vehicles, offices, . . .

Organisations: suppliers, customers, departments, charities, clubs, committees, . . .

Object classes: recordings, films, books, types of stock, biological species, roles, . . .

Events: concerts, examinations, lecture courses, consultations, sales, . . .

Potential entities are usually **nouns** in natural language - but not every noun should be an entity:

- Is there more than one instance of this entity? If not, a constant will do.
- If there are a number of instances, are these fixed? (e.g. the days of the week)? If so, there is no need to create an entity.
- Is anyone really interested in the instances? If not, there is no need for the entity.

Introduction to ER Modelling

Entities and Attributes

Attribute:- A **property** of an entity type which associate each entity with a value from a **domain** of values for that attribute

(Relationship types can also have attributes - more on relationships and attributes later).

Simple Attribute:- An attribute composed of a **single component** with an **independent existence** *(e.g. employee name, part number, cost, etc).*

Composite Attribute:- An attribute composed of **multiple components** each with an **independent existence** *(e.g. Date is day/month/year).*

Multi-valued:- e.g. phone number

Derived Attribute:- An attribute with a value that is **derivable** from one or more attributes *(e.g. Total Cost = Quantity * Unit Cost).*

Every instance of an entity type **must** be **uniquely identifiable** (from every other instance).

*For example, every employee must be **identifiable as different** from any other employee.*

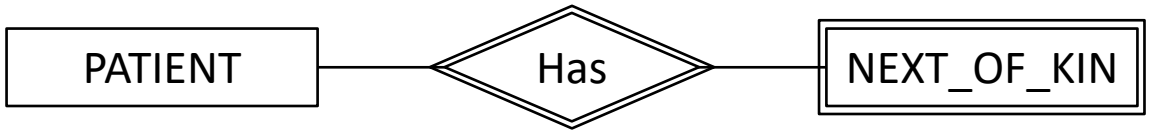
An attribute - or combination of attributes - is chosen to identify each instance of an Entity Type - this is equivalent to the **Primary Key**.

Introduction to ER Modelling

Weak and Strong Entities

Weak Entity Type:- an entity type that is **existence-dependent** on some other entity type
i.e., only exists if another entity exists.

Consider a Hospital Records application:



NEXT_OF_KIN is weak in the relationship

NEXT_OF_KIN has **no independent existence** – only recorded if a PATIENT is recorded – no other relationship.

Strong Entity Type:- An entity type that is **not existence-dependent** on some other entity type.

EMPLOYEE and **DEPARTMENT** are examples

Most entity types are strong

Introduction to ER Modelling

Relationships

Some definitions:

Degree of a Relationship (**unary, binary, ternary, etc**)

The number of participating entity types in a relationship.

Cardinality Ratio (**1:1, 1:N, N:1, M:N**)

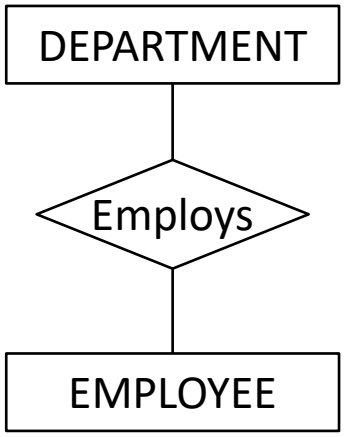
Determines the type of relationships between the participating entity types.

Participation Constraints (**Full or Partial**)

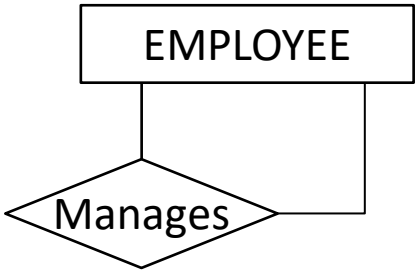
Determines whether the existence of an entity type depends on its being related to another entity type through the relationship

Degree of a Relationship

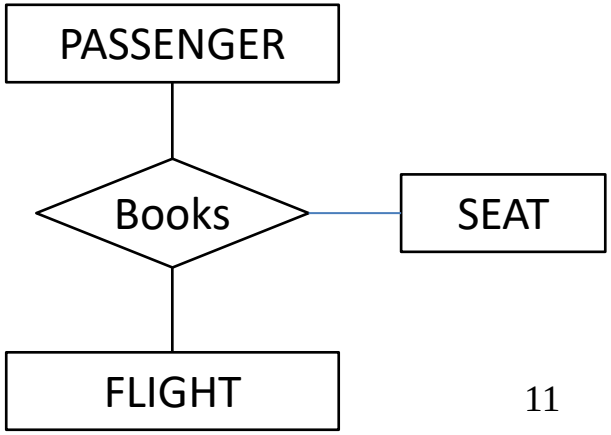
Binary
(connects 2 entity types)



Unary or Recursive
(connects one entity type with itself)



Complex
(connects three or more entity types (Ternary - connects 3))



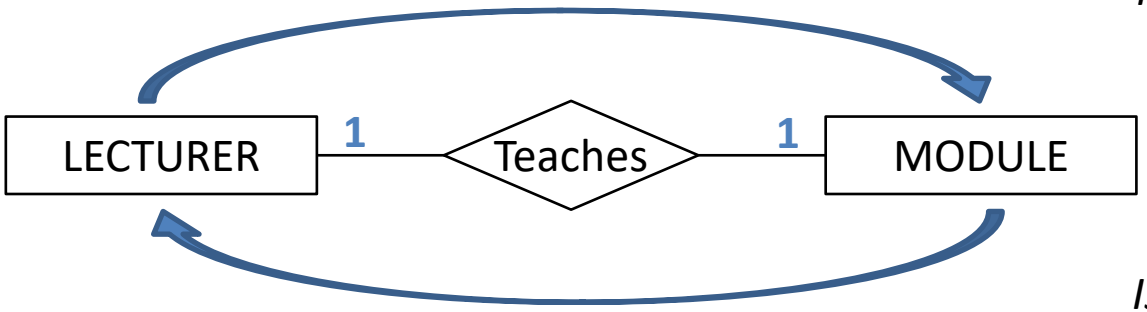
Introduction to ER Modelling

Cardinality Ratio of a Relationship

To determine the cardinality ratio, you must consider the relationship in **BOTH** directions

Consider the following example:

Does every Lecturer teach at most one or more than one (many) module?



Is every Module taught by at most one or more than one Lecturer?

1:1 (one-to-one)

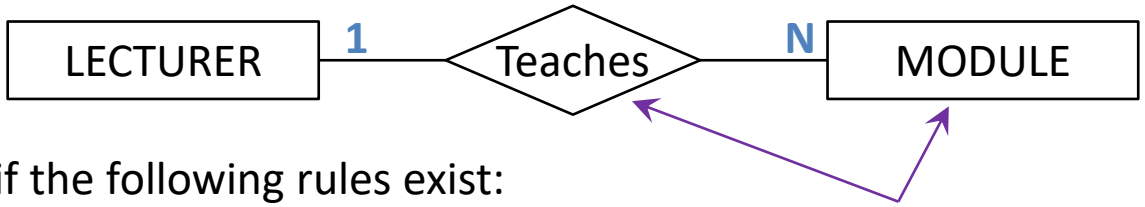
True for the **Teaches** relationship if the following rules exist:

- Every Lecturer teaches **at most** one Module
  (1:1 in the direction LECTURER to MODULE)
- Every Module is taught by **at most** one Lecturer
  (1:1 in the direction MODULE to LECTURER)

Overall a 1:1 relationship is a combination of two 1:1 relationships in both directions

Introduction to ER Modelling

1:N (one-to-many)



In the business scenario:
***nouns** can suggest **entities**,*
verbs** can suggest **relationships

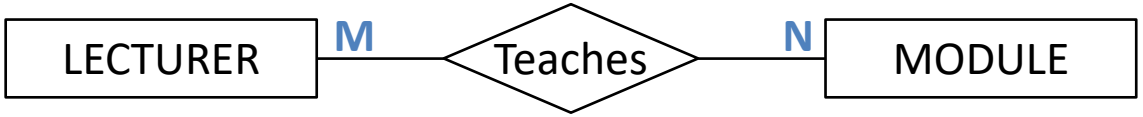
True for the **Teaches** relationship if the following rules exist:

- Some Lecturers may teach **more than one** (**many**) Modules
  (although other might teach only one)
- All Modules are taught by **at most** one Lecturer

So a 1:N relationship is made up of a **1:N** relationship in one direction
AND a **1:1** relationship in the other

*By convention, entity names are singular – **MODULE** rather than MODULES*

M:N (many-to-many)



True for the **Teaches** relationship if the following rules exist:

- Some *(not necessarily all)* Lecturers may teach **many** (**more than one**) Modules
- Some *(not necessarily all)* Modules are taught by **many** (**more than one**) Lecturers

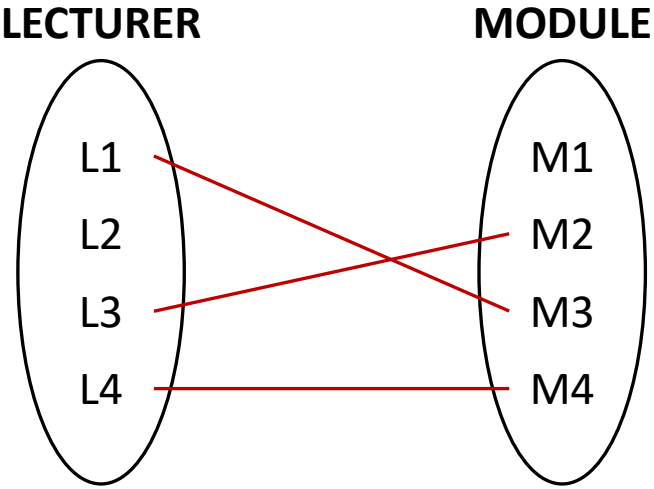
Overall a M:N relationship is a combination of **two 1:N relationships** in both directions

Introduction to ER Modelling

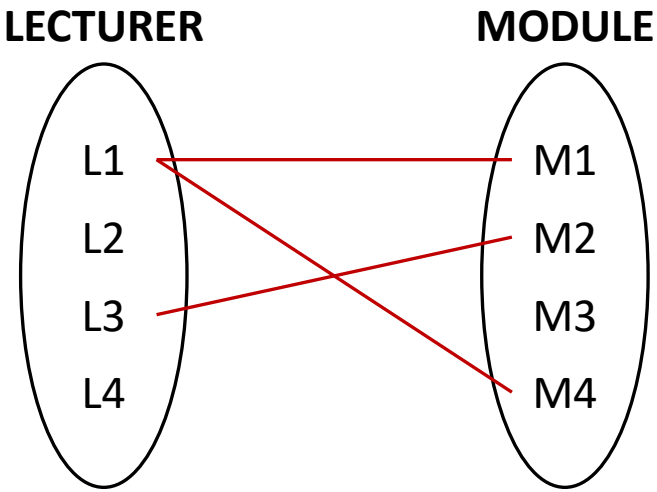
Semantic Networks

A **semantic network**, is a common tool used to represent **meaningful relationships** between **concepts**.

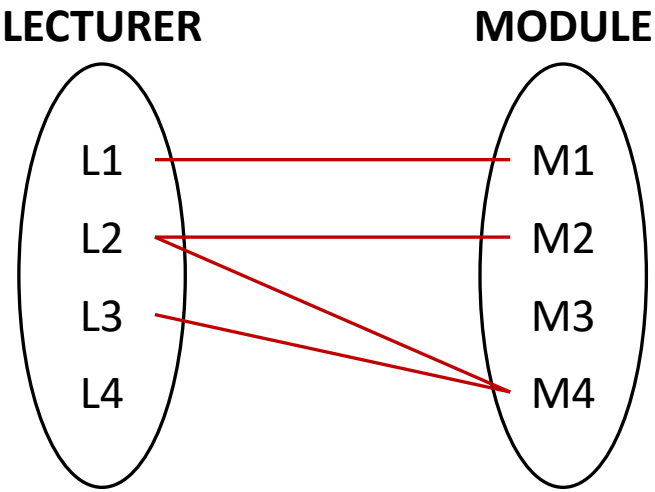
Here they are used to represent **relationships** between **entity occurrences**.



1:1 Relationship



1:N Relationship



M:N Relationship

Introduction to ER Modelling

Time and Relationship Cardinality

Often the time requirements of a relationships affect its cardinality

For the relationship **DEPARTMENT** **Employs** **EMPLOYEE**

- if the database wants to record only the **CURRENT** situation
  (i.e. which Department employs which Employee **at the moment**),
  then a **1:N relationship is appropriate**.

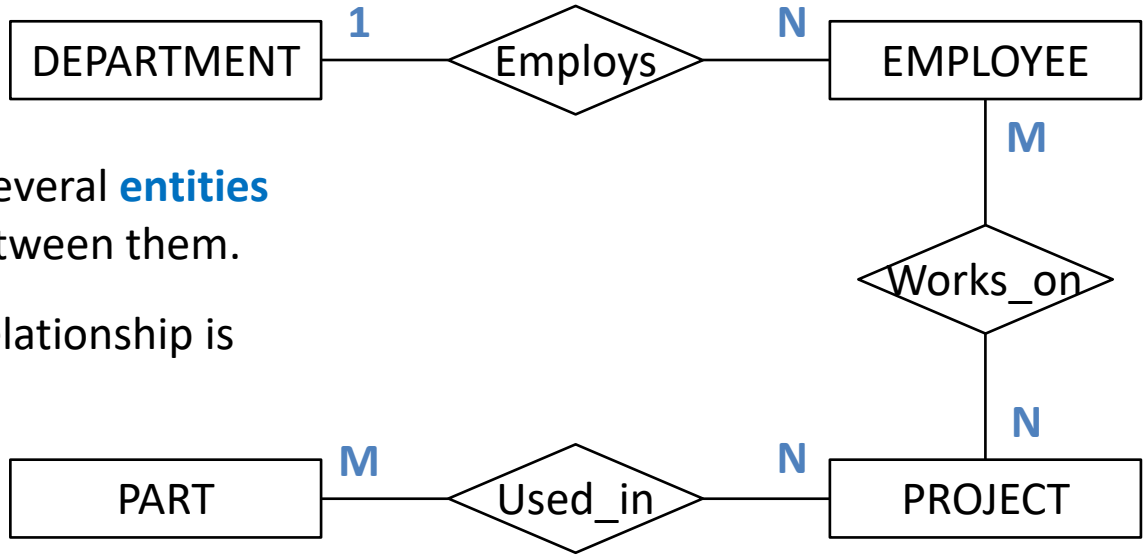
  Thus, if an employee changes department the record of his old department is **deleted**

- if however the database wants to record the **HISTORIC** situation
  (i.e. which departments the employee has been in, **including past departments**),
  then the **appropriate relationship is M:N**.

  Thus, records of an employee's old departments are **stored and not deleted**.

Introduction to ER Modelling

Basic E-R Diagram



An **E-R Diagram** will contains several **entities** and show the **relationships** between them.

The **cardinality ratio** of each relationship is shown on the diagram.

Consider the **Employs** Relationship (***in both directions***)

- Does every **DEPARTMENT** employ at most one or more than one **EMPLOYEE**?
  1 to many – *a department normally employs many employees*
- Is every **EMPLOYEE** employed in just one **DEPARTMENT**
  1 to 1 - *an employee is normally employed in just one department*
  assuming we are considering their current situation only.

So, overall, the **Employs** relationship between **DEPARTMENT** and **EMPLOYEE** has a cardinality ratio of **1:N**.

Now consider the **Used_in** relationship

And finally, the **Works_on** relationship

Introduction to ER Modelling

Participation Constraints

The **participation constraint** is a characteristic of how an entity type participates in a particular relationship

The questions to be asked are:

1. **Is every instance of an entity type linked to an instance of the other entity type in the relationship?**
   *(**full participation** aka **mandatory participation** – no instances of the entity type are allowed to exist unless they are involved in the relationship)*

2. **Or, are some instances not linked?** They exist but are not participating in the relationship
   *(**partial participation** aka **optional participation** – some instances of the entity type can exist without involvement in the relationship)*

Consider the **DEPARTMENT** **Employs** **EMPLOYEE** relationship:

- if **all instances** of the **EMPLOYEE** entity type (i.e. all employees) participate in the **Employs** relationship (i.e. are employed in some department) then **participation constraint** of **EMPLOYEE** in **Employs** is **FULL**

- if however, **any instances** of **EMPLOYEE** (i.e. any employees) do not participate in the **Employs** relationship (i.e. exist but not assigned to any department) then the **participation constraint** of **EMPLOYEE** in **Employs** is **PARTIAL**

Introduction to ER Modelling

There are always **two** possible participation constraints in a (binary) relationship (i.e. **EMPLOYEE** in **Employs** and **DEPARTMENT** in **Employs**) and these **don't have to be the same**

If an entity type participates in more than one relationship (e.g., **EMPLOYEE** or **PROJECT** in the earlier example), it **may participate differently** in those relationships

For mapping to relational model, not all mapping rules are **relevant** (needed to decide which rule to use)

To decide on a participation constraint for a particular entity type in a particular relationship you need to decide on **FULL** or **PARTIAL** participation

(either because information is given in a specification or your common sense shows how it works in the real world)

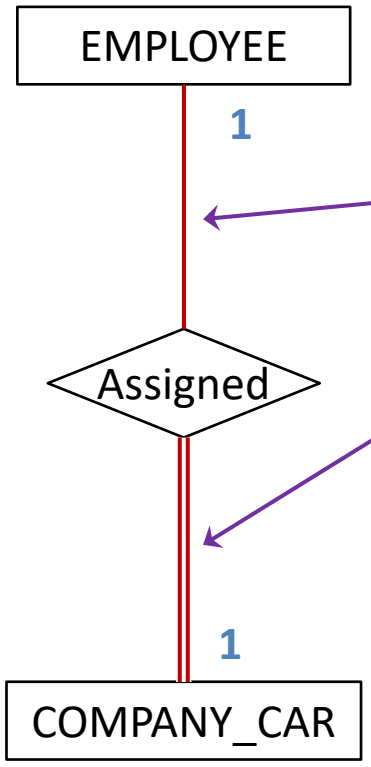
The **most common participation constraint** is **FULL** - this should be the default assumption

(unless there is evidence to the contrary in the specification or from real world knowledge)

Introduction to ER Modelling

Participation Constraints – An Example

Consider an E-R Diagram showing a 1:1 relationship (**Assigned**) between **EMPLOYEE** and **COMPANY_CAR**



If **all** employees are assigned a company car
EMPLOYEE participates **fully** in **Assigned**

If **some** employees are **not assigned** a company car
EMPLOYEE participates **partially** in **Assigned**

If **all** cars are assigned to an employee
COMPANY_CAR participates **fully** in **Assigned**

If **some** cars are **not assigned** to an employee - e.g., a car pool
COMPANY_CAR participates **partially** in **Assigned**

Participation Constraints are represented on an E-R Diagram as follows:

Full Participation:- Double line

Partial Participation:- Single Line

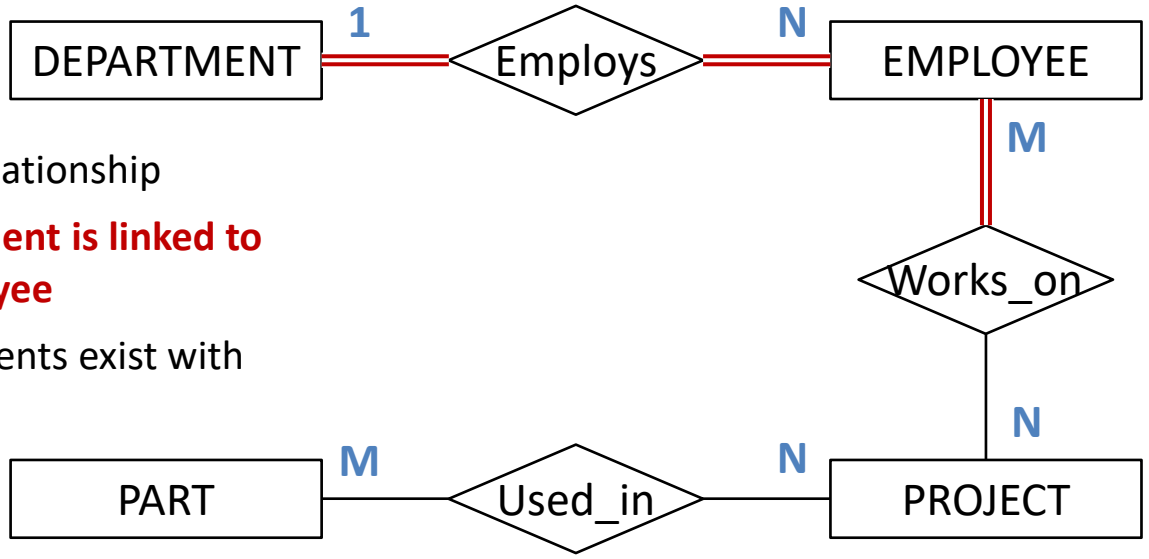
So – if **not all** employees are allocated car and **all** cars are allocated to some employees …..?

Introduction to ER Modelling

Example - E-R Diagram with Participation Constraints

A participation constraint is how an entity type participates in a relationship (**FULL/PARTIAL**)

Consider the basic E-R diagram developed earlier:



DEPARTMENT in **Employs** relationship

FULL - if every department is linked to at least 1 employee

PARTIAL - if some departments exist with NO employees

EMPLOYEE in **Employs** relationship

FULL - if every employee is always in one department

PARTIAL - if some employees exist with NO department

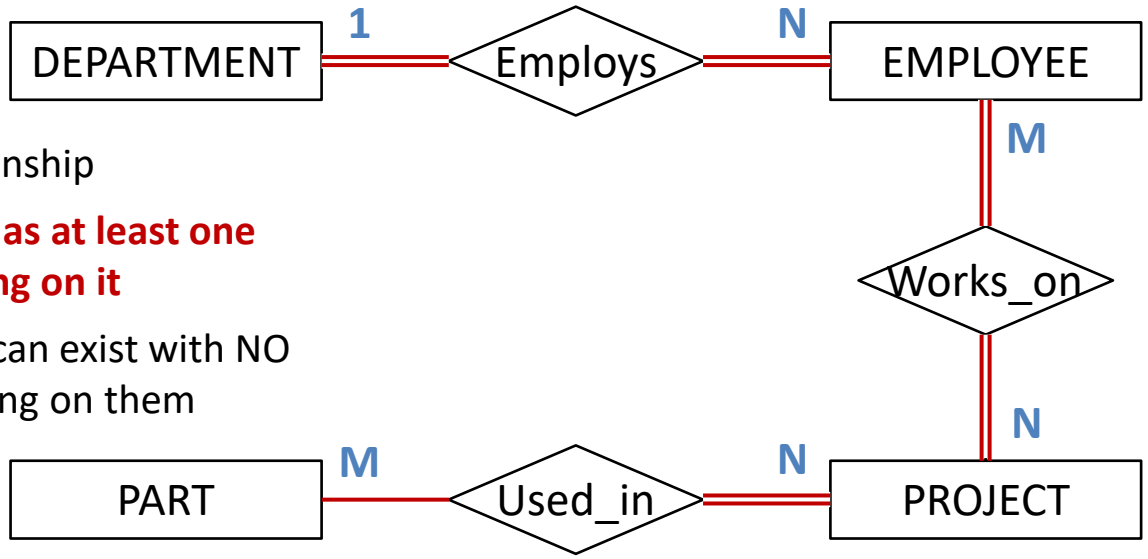
EMPLOYEE in **Works_on** relationship

FULL - if every employee has to work on at least one project

PARTIAL - if some employees work on NO projects

Introduction to ER Modelling

Example continued:



PROJECT in **Works_on** relationship

FULL - if every project has at least one employee working on it

PARTIAL - if some projects can exist with NO employees working on them

PROJECT in **Used_in** relationship

FULL - if every project uses at least one part

PARTIAL - if some projects can exist using NO parts

PART in **Used_in** relationship

FULL - if every part is used in at least one project

PARTIAL - if some parts exist but are not used in any project

Introduction to ER Modelling

Another Example - *Studies Advice System*

Draw an E-R Diagram which models the **University's Studies Advice System** (ie. *Individual Meetings between Student & Advisor*)

STUDENTS on a **COURSE** are allocated a **STUDIES ADVISER** who sets up **MEETINGS** (Studies Advice Meetings with Students)

Timescale: One Academic Year

Solution

First, identify the Entities.

Then consider which relationships should be modelled?

Most Important:

COURSE Contains STUDENT

STUDENT Allocated ADVISOR

STUDENT Attends MEETING

MEETING Hosted By ADVISOR