WEEK 1

ENTITY RELATIONSHIP DIAGRAMS – REPRESENTING RELATIONSHIPS – PART 3

STUDENT OBJECTIVES

- Upon completion of this video, you should be able to:
  - Define the following terms: *Relationship, Cardinality, Unary Relationship, Binary Relationship, Ternary Relationship, One to One Relationship, One to Many Relationship,* and *Many to Many Relationship* and give an example of each
  - Using lines, diamonds and text, indicate the above relationships on an ER diagram
  - Determine if a relationship has any associated attribute(s)
  - Represent relationship attributes on an ER diagram

WHAT ARE WE MISSING FROM OUR MODEL? WHAT HAVEN'T WE REPRESENTED YET?

- Look again at our case study, what are we missing from our ER diagram?

CASE STUDY – CREATING AN ER DIAGRAM

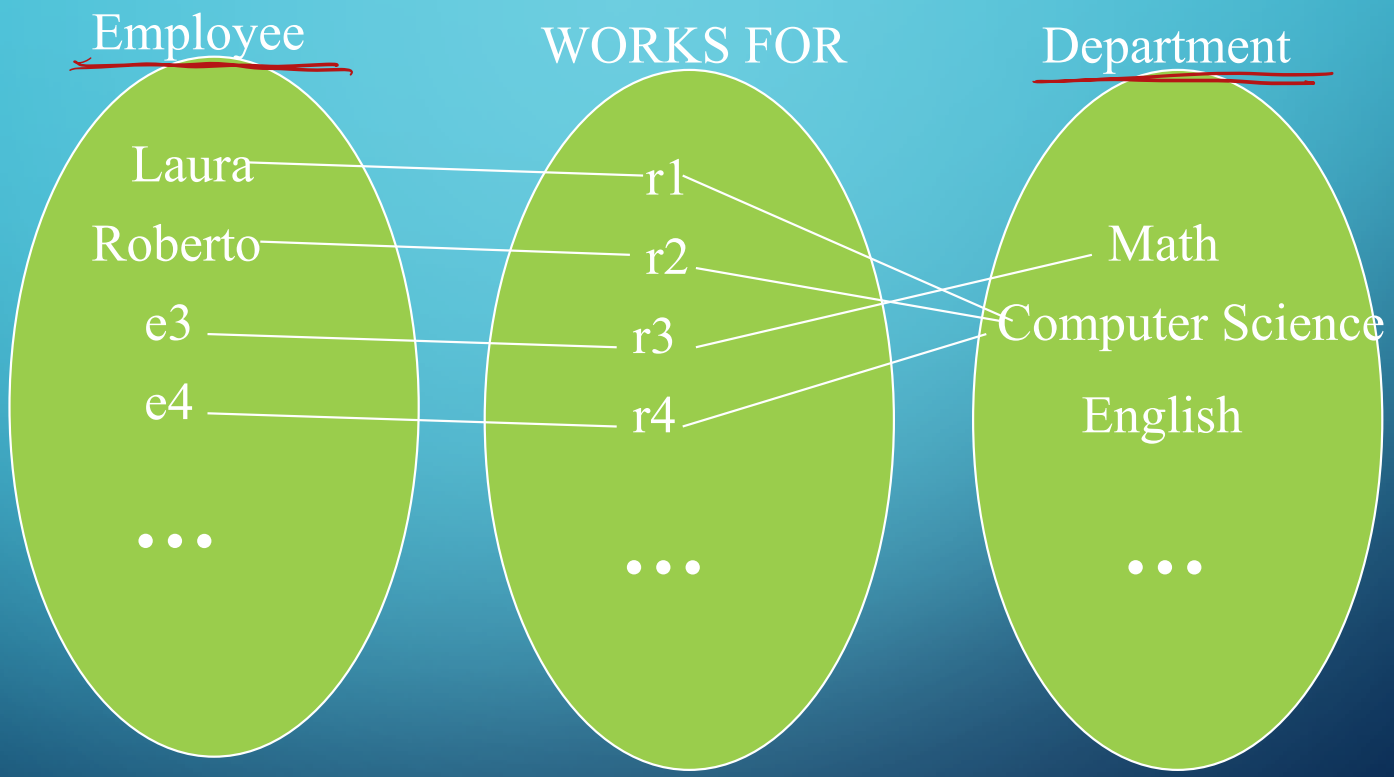
- Suppose we plan to model a company which is organized into departments.
- Each department has a unique name, number and employee who manages it (we want to keep track of when the employee started managing the department)
- A department may have several locations
- A department controls a bunch of projects, each project has a unique number, name and a single location
- Each employee has a name, ssnumber, address, salary, sex and birthdate
- An employee is assigned to only one department but may work on several projects which are not necessarily from the same department
- Keep track of the number of hours each employee works on each project.
- Keep track of the direct supervisor of each employee
- Keep track of the dependents of each employee (name, sex, birthdate and relation)

E-R MODEL CONCEPTS AND KEY TERMS

- **Relationship** – a named grouping of entities
- **Relationship Set** – an ordered list of entity sets
- **A Relationship Type R** among n entity types E1, E2, … En defines a set of associations among entities. Thus R is a set of relationship instances ri, where each ri associates n entities (e1, e2, ..., en) and each entity ej in ri is a member of entity type Ej, 1<=j<=n. Hence a relationship type is a mathematical relation on E1, E2, ... En.

Example: ("Reid", "CS3319") is a relationship set of (Prof, Course)

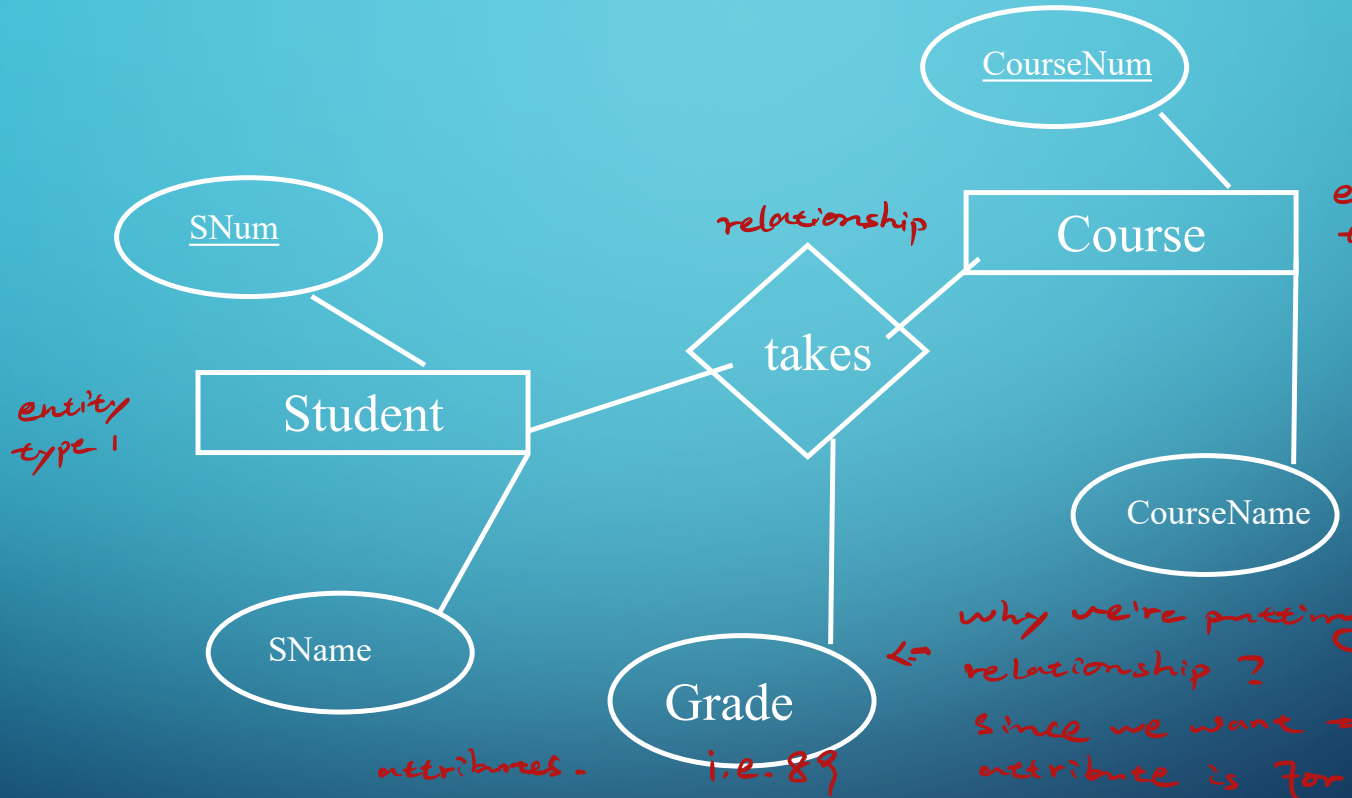
E.G. THE RELATIONSHIP: *EMPLOYEE* ***WORKS FOR*** *DEPARTMENT*



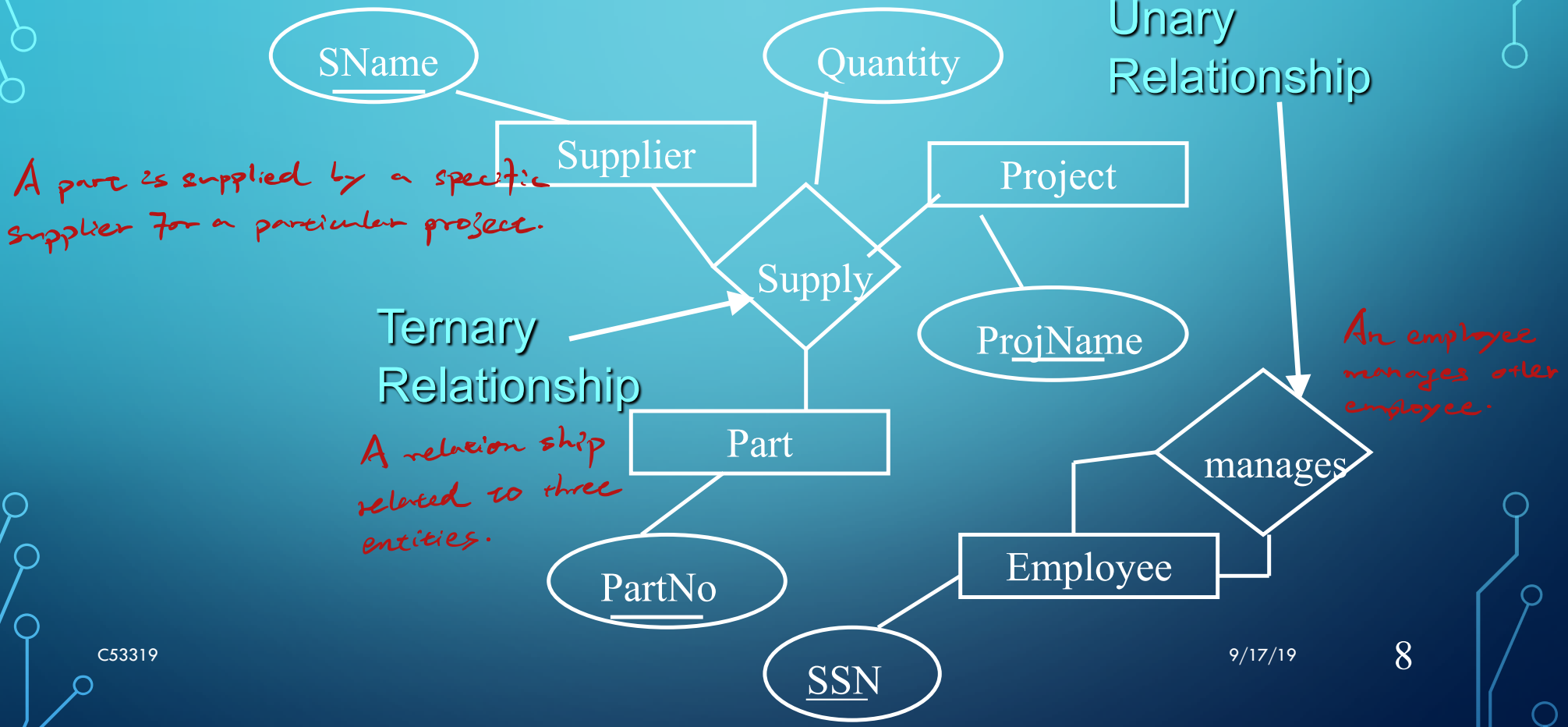
QUESTION: In the above diagram, how many entity types participate in the relationship?

2 types, so it is also called binary relationship.

BINARY RELATIONSHIP - EXAMPLE



DEGREE OF A RELATIONSHIP: BINARY, TERNARY, UNARY



MORE TERMINOLOGY

- **Recursive (Unary) Relationship:** an entity of one entity type has a relationship with other entities of that same entity (type).
- **Attributes on Relationships:** Describes some piece of information about the relationship. E.g. → Quantity see above
- **Cardinality Ratio:** number of relationships instances that an entity can participate in, there are 3 common ones:
  - One-To-One: Employee Manages Department
  - Many-To-One: Employee Works_For Department
  - Many-To-Many: Employee Works_On Project

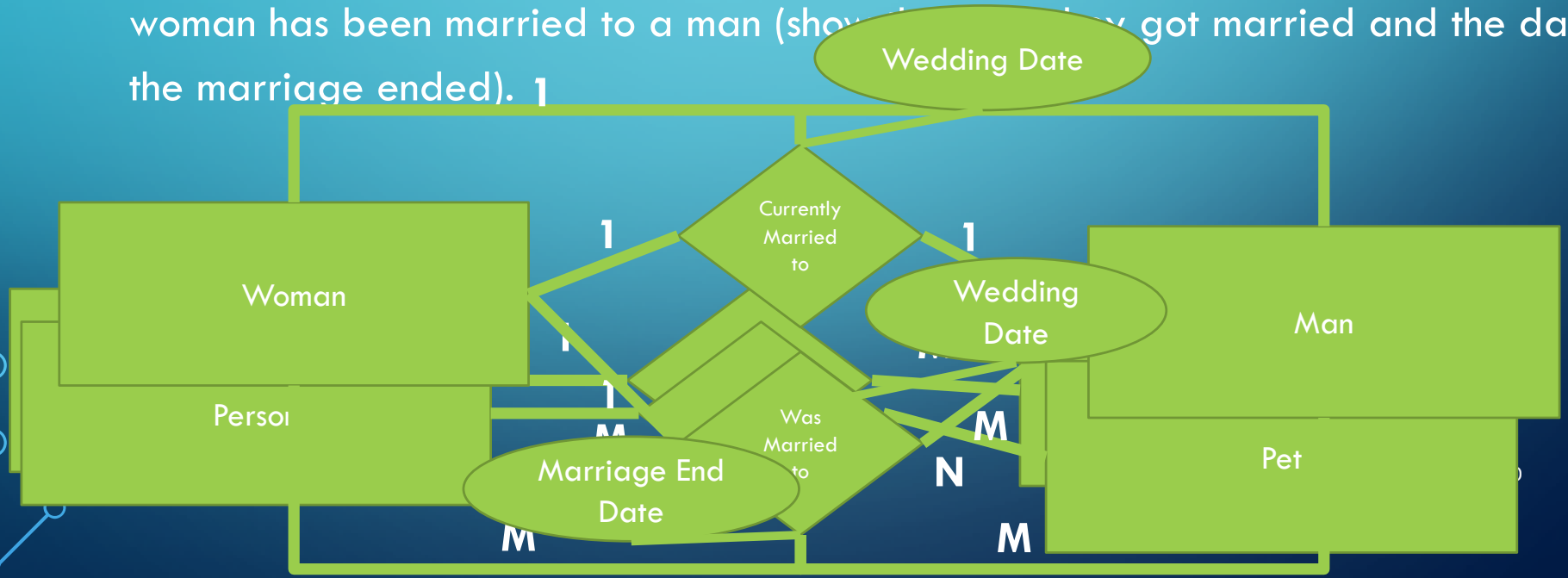
QUESTION: What is the cardinality of: Man *BIOLOGICALLY FATHERS* Child?

A man could have one or many child.
A child could have only one father

QUESTION: Give an example of a *many to many* relationship:

DRAW THE FOLLOWING RELATIONSHIPS AS THEY WOULD BE REPRESENTED IN AN ER DIAGRAM:

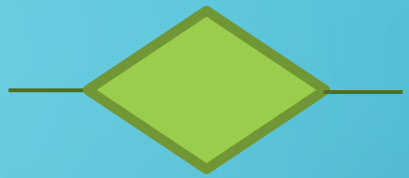
- An artist paints paintings
- A person owns pets
- A woman is currently married to a man (and show the date they got married) and a woman has been married to a man (sho got married and the date the marriage ended).



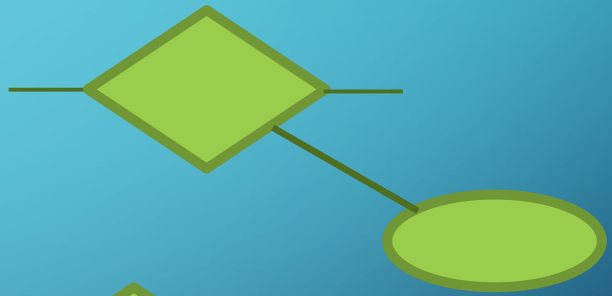

artist
1
paint
N
painting.
1
now married to
1
1
woman
N
was married to
N
man
N
N

E-R DIAGRAM NOTATION SO FAR:

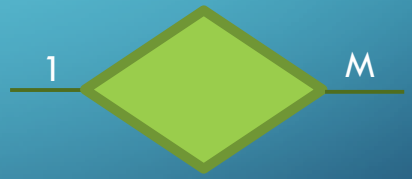
- Relationship Type



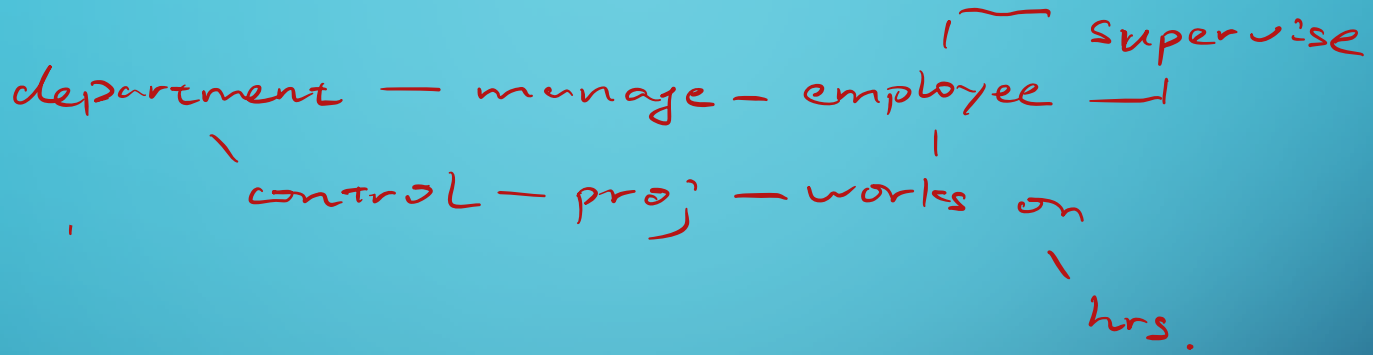
- Relationship Attribute



- Cardinality



QUESTION: WHAT ARE THE RELATIONSHIPS FROM OUR EXAMPLE?

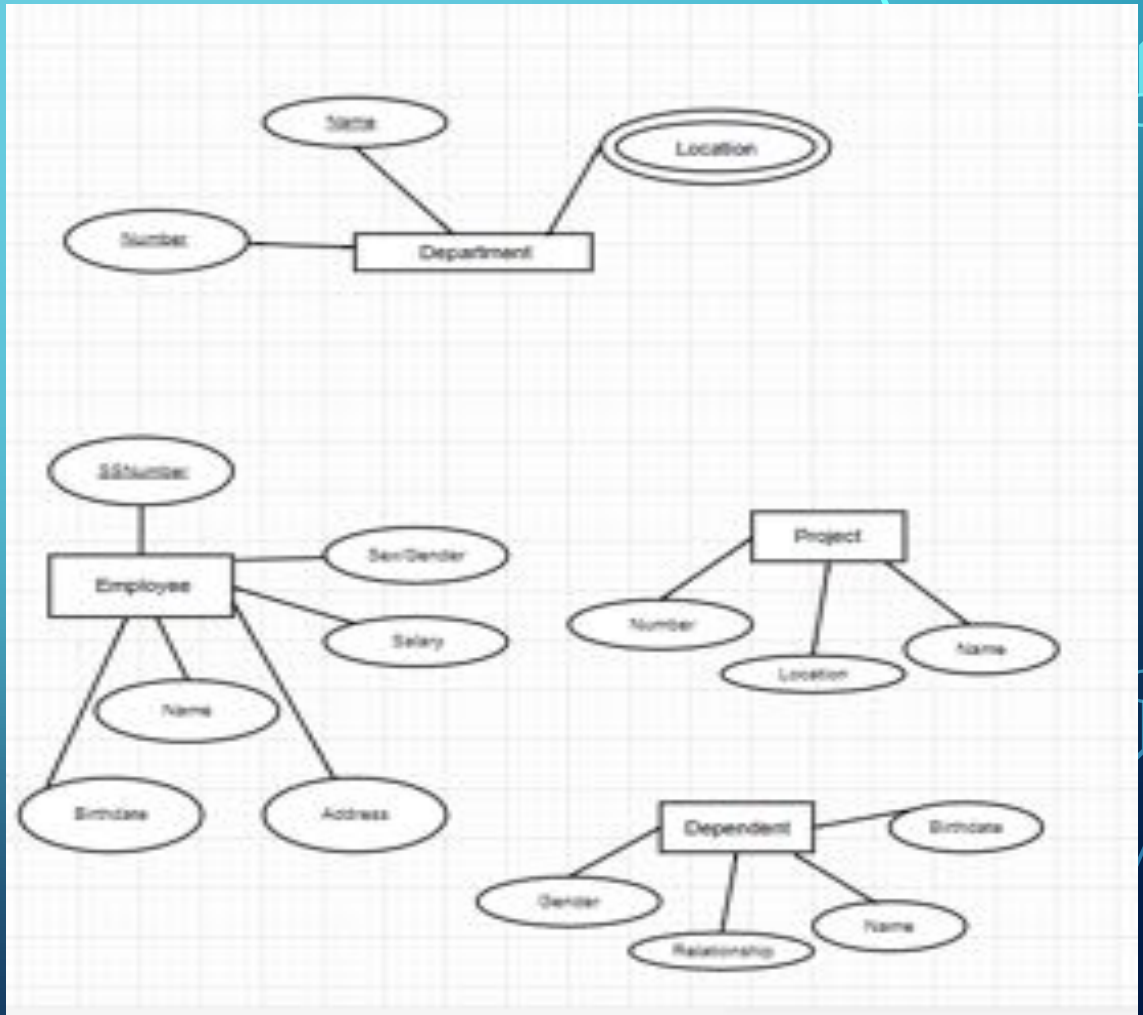


CASE STUDY – CREATING AN ER DIAGRAM

- Suppose we plan to model a company which is organized into departments.
- Each department has a unique name, number and employee who manages it (we want to keep track of when the employee started managing the department)
- A department may have several locations
- A department controls a bunch of projects, each project has a unique number, name and a single location
- Each employee has a name, ssnumber, address, salary, sex and birthdate
- An employee is assigned to only one department but may work on several projects which are not necessarily from the same department
- Keep track of the number of hours each employee works on each project.
- Keep track of the direct supervisor of each employee
- Keep track of the dependents of each employee (name, sex, birthdate and relation)

QUESTION: WHAT IS OUR DIAGRAM SO FAR? (IT IS STARTED BELOW)

Let's use draw.io to finish the diagram.



C++ Programming

Getting Started

Before We Begin

- When programming in C++, always keep in mind its C roots
  - It inherits most of its syntax and structure from C
  - Most (but not quite all) C code is valid C++ code; as a result, C++ is not quite a strict superset of C
  - Entire programs in C++ can be written using only regular functions not defined in any class; in other words, classes are not mandatory
- This is absolutely not the right way to write an object-oriented program; use classes, objects, and methods!

Structure of a C++ Program

- The basic elements of a C++ program are
  - The classes (i.e., a notion of Abstract Data Types),
  - The methods (i.e., functions encapsulated in classes), and
  - The data members (i.e., data fields encapsulated in classes)

Structure of a C++ Program

- Most programs are made up of multiple classes (with methods and data members) and functions
- A main function is required for a program as an entry point to bootstrap the rest of its functionality
  - One main function must exist
  - No more than one can exist in the same program

The Simplest C++ Program

```
int main()
{
}
```

C's Hello World is a C++ Program

```
#include <stdio.h>

/* Simple Hello World program. */

int main()
{
   printf("Hello World!\n");
}
```

A More C++-ish Hello World

```
#include <iostream>

// Simple Hello World program.

int main()
{
   std::cout << "Hello World!" << std::endl;
}
```

A Slightly Better More C++-ish Hello World

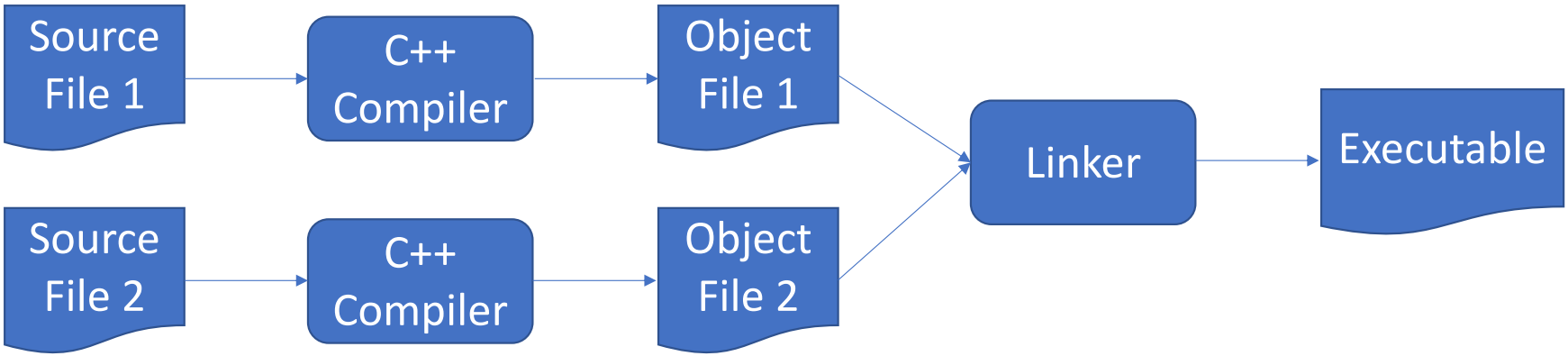
```
#include <iostream>
using namespace std;

// Simple Hello World program.

int main()
{
   cout << "Hello World!" << endl;
}
```

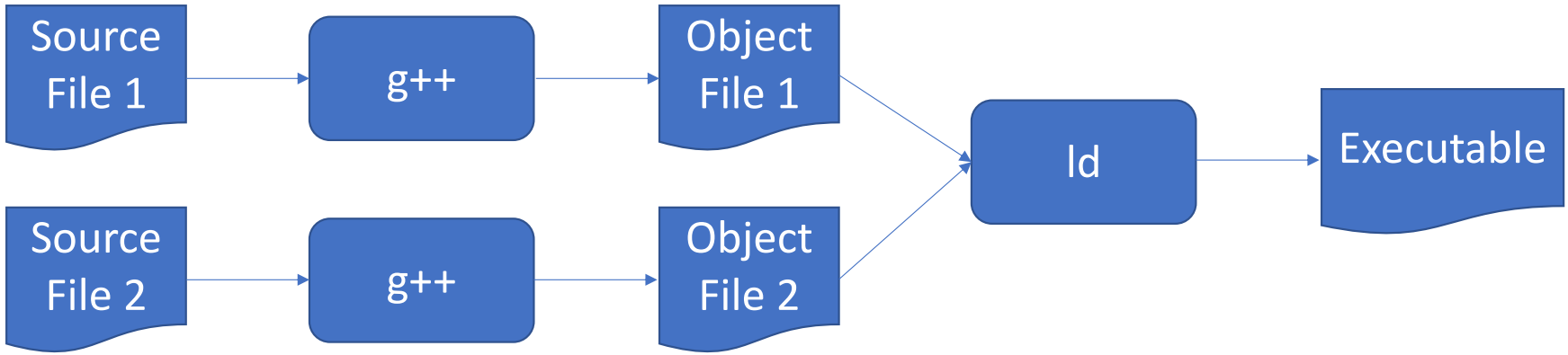
Building C++ Programs

- C++ is a compiled language and so to run a C++ program, its source code must be compiled and linked to produce an executable



Building C++ Programs

- On Linux and most Unix-like systems, things would typically be built using g++ and ld (though sometimes c++ is used instead, and ld is often hidden)



Building C++ Programs

- From a command line, building in one step would look like:

```
> g++ HelloWorld.cpp -o HelloWorld
> ./HelloWorld
```

- Alternatively, you can build and keep the object files and do things in multiple steps like:

```
> g++ -c HelloWorld.cpp
> g++ HelloWorld.o -o HelloWorld
> ./HelloWorld
```