Developing Entity Relationship Diagrams (ERDs)

Introduction

This document seeks to give expanded explanation and examples of how to produce entity relationship diagrams.

It is based on material adapted from a previous CQU course web page (http://infocom.cqu.edu.au/Courses/spr2000/95169/Extra_Examples/ERD.htm) and material used in Lecture 7 for COIS20025 in Term 2, 2006.

Why ERDs?

Entity Relationship Diagrams are a major data modelling tool and will help organize the data in your project into entities and define the relationships between the entities. This process has proved to enable the analyst to produce a good database structure so that the data can be stored and retrieved in a most efficient manner.

By using a graphical format it may help communication about the design between the designer and the user and the designer and the people who will implement it.

Components of an ERD

An ERD typically consists of four different graphical components:

1. Entity.
   A data entity is anything real or abstract about which we want to store data. Entity types fall into five classes: roles, events, locations, tangible things or concepts. E.g. employee, payment, campus, book. Specific examples of an entity are called instances. E.g. the employee John Jones, Mary Smith's payment, etc.
2. Relationship.
   A data relationship is a natural association that exists between one or more entities. E.g. Employees process payments.
3. Cardinality.
   Defines the number of occurrences of one entity for a single occurrence of the related entity. E.g. an employee may process many payments but might not process any payments depending on the nature of her job.
4. Attribute.
   A data attribute is a characteristic common to all or most instances of a particular entity. Synonyms include property, data element, field. E.g. Name, address, Employee Number, pay rate are all attributes of the entity employee. An attribute or combination of attributes that uniquely identifies one and only one instance of an entity is called a primary key or identifier. E.g. Employee Number is a primary key for Employee.

Figure 1 is a very simple, example ERD with each of the four components labelled.

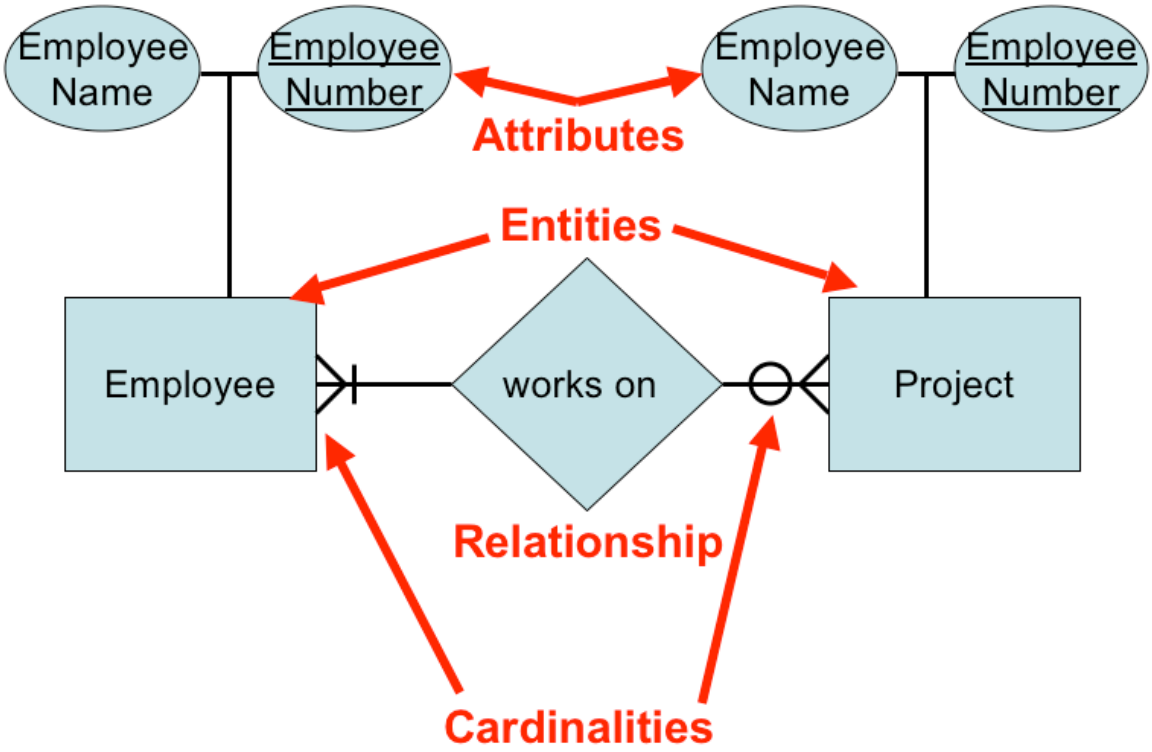


Figure 1
A simple, example ERD

Different ERD styles

As with many data modelling tools there are a number of different styles used to create ERDs. This web page (http://www.smartdraw.com/tutorials/software-erd/erdcardinality.htm) lists four different styles for caradinalities.

The style used in this course will be the one labelled "Information Engineering".

One Methodology for Developing an ERD

Typically you will start with a case study or perhaps a logical model of the system to be developed. This document will demonstrate how to use the following process to convert that information into an ERD.

The process has ten steps:

1. Identify Entities
   Identify the roles, events, locations, tangible things or concepts about which the end-users want to store data.
2. Find Relationships
   Find the natural associations between pairs of entities using a relationship matrix.
3. Draw Rough ERD
   Put entities in rectangles and relationships on line segments connecting the entities.

4. Fill in Cardinality
   Determine the number of occurrences of one entity for a single occurrence of the related entity.
5. Define Primary Keys
   Identify the data attribute(s) that uniquely identify one and only one occurrence of each entity.
6. Draw Key-Based ERD
   Eliminate Many-to-Many relationships and include primary and foreign keys in each entity.
7. Identify Attributes
   Name the information details (fields) which are essential to the system under development.
8. Map Attributes
   For each attribute, match it with exactly one entity that it describes.
9. Draw fully attributed ERD
   Adjust the ERD from step 6 to account for entities or relationships discovered in step 8.
10. Check Results
    Does the final Entity Relationship Diagram accurately depict the system data?

A Simple Example

The above process will be illustrated by working through the following example.

> A company has several departments. Each department has a supervisor and at least one employee. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects. The important data fields are the names of the departments, projects, supervisors and employees, as well as the supervisor and employee number and a unique project number.

Each of the following sections corresponds to one of the stages above.

Identify entities

In this stage, you look through the information about the system and seek to identify the roles, events, locations, concepts and other tangible things that you wish to store data about. One approach to this is to work through the information and highlight those words which you think correspond to entities.

> A **company** has several **departments**. Each department has a **supervisor** and at least one **employee**. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a **project**, but an employee may be on vacation and not assigned to any projects. The important data fields are the names of the departments, projects, supervisors

and employees, as well as the supervisor and employee number and a unique project number.

This example is quite simple in that the last couple of lines actually tell you what data is being stored and that makes it somewhat easy to identify the entities.

You may notice that "**company**" has been highlighted. It is **not** an example of an entity. A single company will use the system we are designing to keep track of its departments, projects, supervisors and employees.

A true entity should have more than one instance. Our system will probably contain information about multiple employees, supervisors, projects and departments. But it will only contain one instance of a company.

Find Relationships

In this step the aim is to identify the associations, the connections between pairs of entities. A simple approach to do this is using a relationship matrix. Which is a fancy name for a table that has rows and columns for each of the identified entities.

Table 1 is an example relationship matrix for the above example. With four entities there are four rows and four columns. Each cell is used to indicate whether or not that combination of entities has an association.

	Department	Employee	Supervisor	Project
Department				
Employee				
Supervisor				
Project				

Table 1
An example relationship matrix.

Having created your relationship matrix you should now go through each cell and decide whether or not there is an association. For example, the first cell on the second row is used to indicate if there is a relationship between the entity "Employee" and the entity "Department". Table 2 is an example relationship matrix that has been completed for the current example.

	Department	Employee	Supervisor	Project
Department		Is assigned	Run by	
Employee	Belongs to			Works on
Supervisor	Runs			
Project		Uses		

Table 2
An example complete relationship matrix.

The names placed in the cells are meant to capture/describe the relationships. So you can use them like this

- A Department is assigned an employee
- A Department is run by a supervisor
- An employee belongs to a department
- An employee works on a project
- A supervisor runs a department
- A project uses an employee

Draw Rough ERD

Your completed relationship matrix now contains a list of all the entities and all the relationships between those entities. This is enough information to create a rough ERD.

Draw a diagram and:

- Place all the entities in rectangles
- Use diamonds and lines to represent the relationships between entities.

Obviously, you should lay out the entities so there is no overlap of the relationships. Figure 2 is an example rough ERD that represents the content of Table 2.

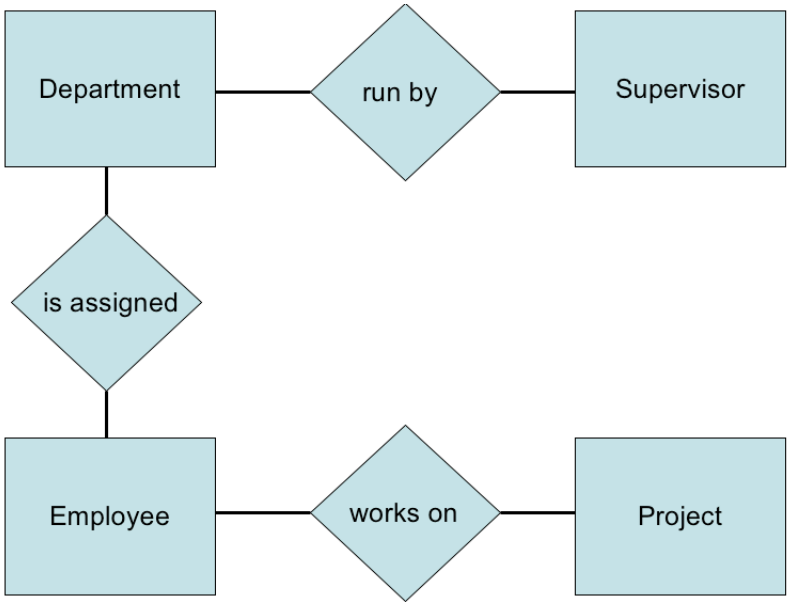


Figure 2
An example rough ERD

Fill in Cardinality

In this step we're aiming to identify the number of occurrences of one entity for a single occurrence of the related entity. For example, how many supervisors can there be for a single department? How many departments can a single supervisor be associated with?

To answer this you can loop through each entity from our rough diagram and ask the question "how many of this entity?" for a single instance of each related entity?

Here's an example drawing on Figure 2 and the case study described above

- Supervisor
  Each department has one supervisor.
- Department
  Each supervisor has one department.
  Each employee can belong to one or more departments
- Employee
  Each department must have one or more employees
  Each project must have one or more employees
- Project
  Each employee can have 0 or more projects.

The cardinality of a relationship can only have the following values

- One and only one
- One or more
- Zero or more
- Zero or one

An ERD is modified to show this cardinality by place some extra marks on the line connecting an entity and a relationship. Figure 3 summarises the marks/notation used in ERDs to indicate cardinality.

Figure 4 is a modified ERD to show the cardinality of the various relations in our example.

Define primary keys

A primary key is an attribute, or collection of attributes, that can be used to uniquely identify a specific instance. My name, "David Jones", is not a primary key as there are many people with that name. If I was a student at CQU, my student number would be a primary key as each student number uniquely identifies one and only one student.

You identify primary keys by examining and evaluating the information about the system. In this example, the following are specified as unique identifiers: Department Name, Supervisor Number, Employee Number and Project Number.

Typically you are looking for a primary key for each entity. You can modify your ERD (e.g. Figure 5) to include the primary keys as attributes to the related entities.

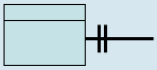
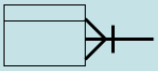
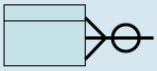
Symbol	Meaning
	One and only one
	One or more
	Zero or more
	Zero or one

Figure 3
Cardinality ERD notation

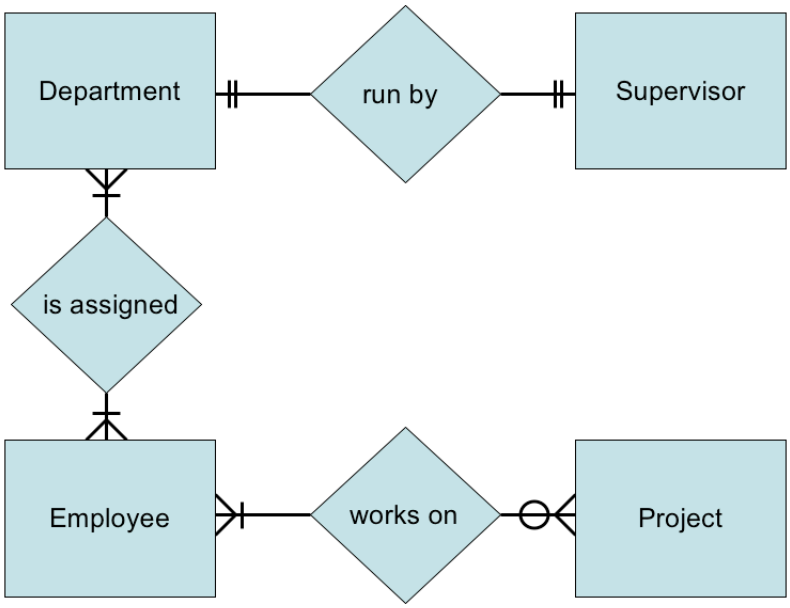


Figure 4
Rough ERD plus Cardinality

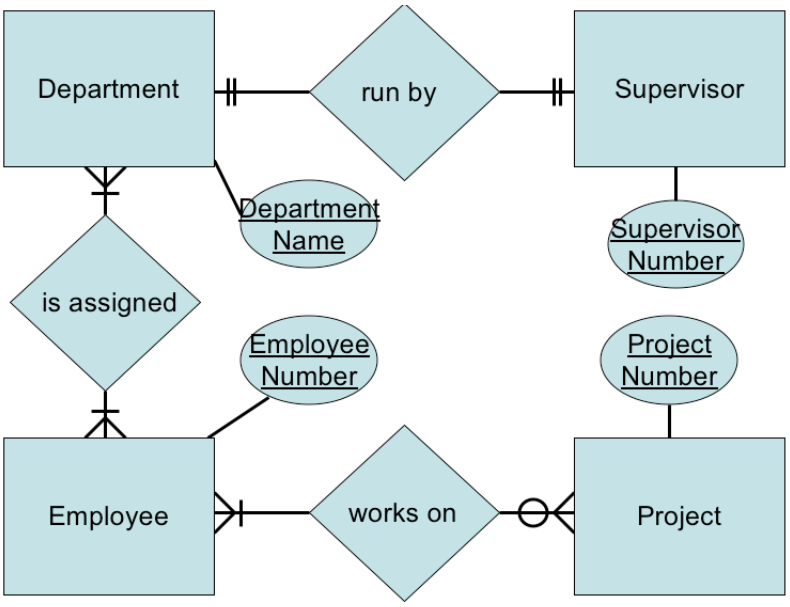


Figure 5
Rough ERD plus Primary Keys

Draw Key-Based ERD

This step cleans up some of the shortcomings of the ERD created so far. In particular, it involves

- Eliminating many-to-many relationships, and
- Adding Primary and Foreign keys for each entity (a task I started in the last step)

What is a many-to-many relationship

A many-to-many relationship is a relationship that has a "many" cardinality on either side of the relationship.

In Figure 5 there are three relationships. The following demonstrates which, if any, of these relationships are a many-to-many relationship.

1. Run by
   The cardinality on both sides of this relationship is "one and only one". This is shown by the two vertical bars on either side (see Figure 3). So this is not a many-to-many relationship.
2. Is Assigned
   The cardinality on both sides of the relationship is indicated by an "arrow" with a single line. According to Figure 3 this means that it is a "one or more" cardinality. The "more" is the same as "many". So both cardinalities include "many". There is a "many" cardinality on both sides of the relationship. **This is a many-to-many relationship.**
3. Works on
   Using Figure 3 you should be able to work out that one cardinality is "one or

more" and the other is "zero or more". Both sides contain "more" which is the same as "many" and so **this is a many-to-many relationship**.

Why are many-to-many relationships bad?

Two of the reasons why many-to-many relationships are bad and should be removed from an ERD include:

1. They can't be represented in relational databases.
2. You often wish to record information about a many-to-many relationship. For example, you may wish to record when an employee starts working on a project. With the current ERD you wouldn't be able to store this information. It is not information that belongs in either the Employee or the Project entity.

At this stage, all you really need know is you have to get rid of them.

How do you get rid of many-to-many relationships?

The simple answer is that you replace the relationship with an associative entity. This splits the relationship into two relationships, each will be a one-to-many relationship.

So the "Department is assigned an employee" many-to-many relationship becomes two separate relationships:

1. A single Department entity is assigned many Employee-Department entities.
2. An Employee-Department entity includes a single Department entity.

Figure 6 is an ERD that represents this change. From this you can see that the Employee-Department entity has a primary key that consists of both Employee Number and Department Name. This combination of two fields uniquely identifies each instance of this entity. So if we wished to store information about the date an employee started working for a department we would store it with this entity.

A similar change happens with the "Works on" relationship. This is shown in Figure 7. Both Figure 6 and Figure 7 would be combined in a final ERD. Shown later on.

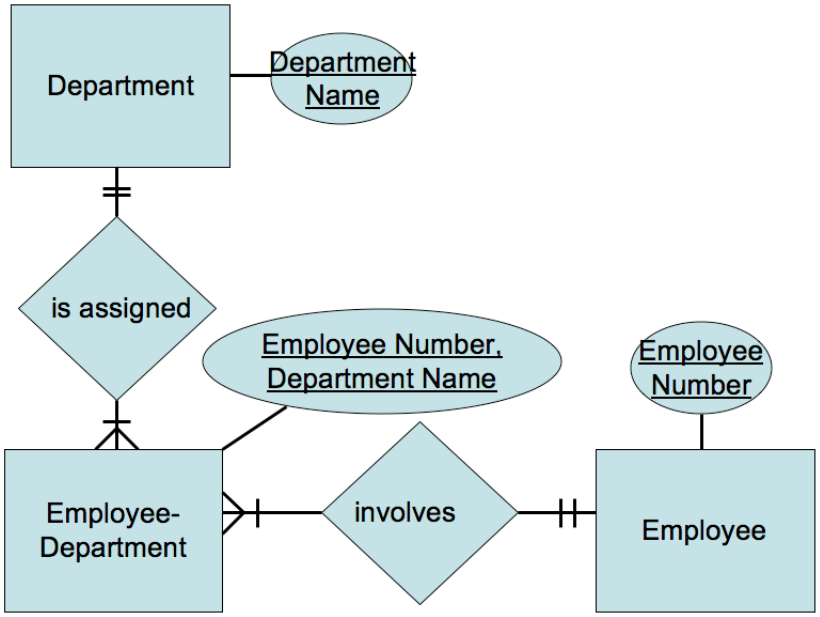


Figure 6
ERD representing Employee-Department Associative Entity

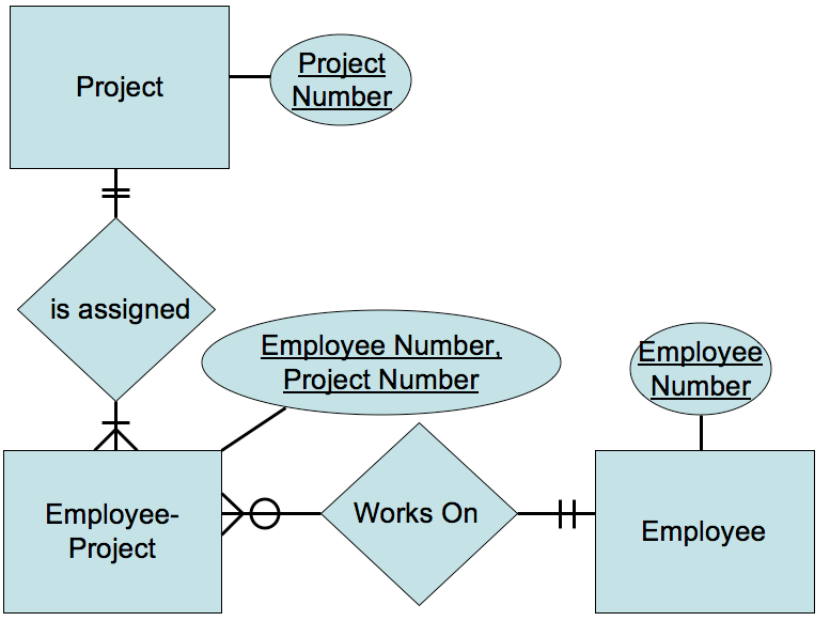


Figure 7
ERD representing Employee-Project Associative Entity

Identify Attributes

A data attribute is a characteristic common to all or most instances of a particular entity. In this step we try to identify and name all the attributes essential to the system we are studying without trying to match them to particular entities. The best way to do this is to study the forms, files and reports currently kept by the users of the system and circle each data item on the paper copy. Cross out those which will not be transferred to the new

system, extraneous items such as signatures, and constant information which is the same for all instances of the form (e.g. your company name and address). The remaining circled items should represent the attributes you need. You should always verify these with your system users. (Sometimes forms or reports are out of date.)

The only attributes indicated are the names of the departments, projects, supervisors and employees, as well as the supervisor and employee NUMBER and a unique project number.

Map Attributes

For each attribute we need to match it with exactly one entity. Often it seems like an attribute should go with more than one entity (e.g. Name). In this case you need to add a modifier to the attribute name to make it unique (e.g. Customer Name, Employee Name, etc.) or determine which entity an attribute "best' describes. If you have attributes left over without corresponding entities, you may have missed an entity and its corresponding relationships. Identify these missed entities and add them to the relationship matrix now.

Attribute	Entity	Attribute	Entity
Department Name	Department	Supervisor Number	Supervisor
Employee Number	Employee	Supervisor Name	Supervisor
Employee Name	Employee	Project Name	Project
		Project Number	Project

Table 3
Attribute-Entity Mapping

Draw Fully Attributed ERD

If you introduced new entities and attributes in step 8, you need to redraw the entity relationship diagram. When you do so, try to rearrange it so no lines cross by putting the entities with the most relationships in the middle. If you use a tool like Systems Architect, redrawing the diagram is relatively easy.

Even if you have no new entities to add to the Key-Based ERD, you still need to add the attributes to the Non-Key Data section of each rectangle. Adding these attributes automatically puts them in the repository, so when we use the entity to design the new system, all its attributes will be available.

Figure 8 is an example of a fully-attributed ERD for the example problem.

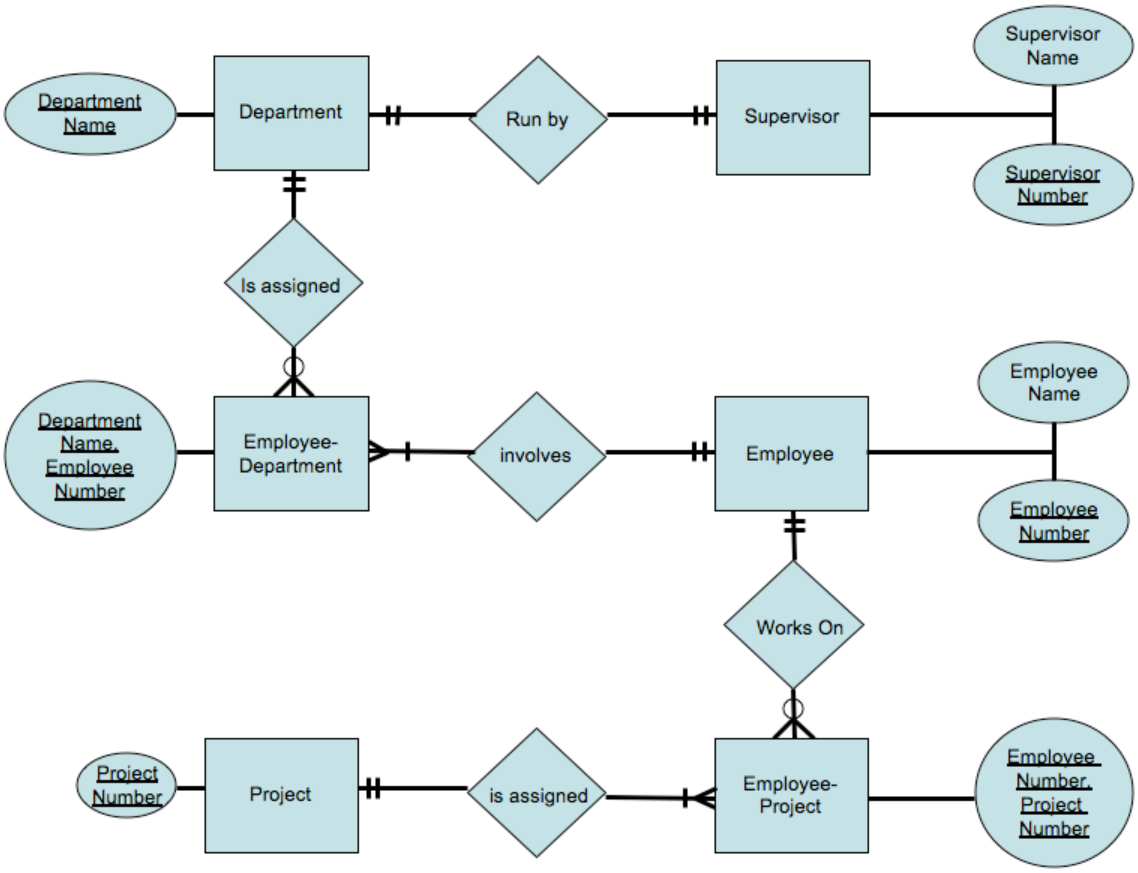


Figure 8
Fully attributed ERD

Check ERD Results

Look at your diagram from the point of view of a system owner or user. Is everything clear? Check through the Cardinality pairs. Also, look over the list of attributes associated with each entity to see if anything has been omitted.