The Entity-Relationship Diagram and its Different Notations

Contents

1. Introduction

This is a supplement that explains the differences and shortcomings of some popular entity-relationship diagram (ERD) notations which you will or may already have encountered in your studies. “The cat sat on the mat” can be said in many different languages: Chinese, Japanese, English, Urdu etc. The meaning of the phrase would be the same whichever language it was spoken. This is the same with the notation that we use for database design, such as Chen, SSADM (Structured Systems Analysis and Design Method) and UML (Unified Modelling Language). The same concepts can be expressed using lots of different notations. It is important that you remember that the underlying concepts being represented in any given database design remain the same. This is true regardless of the notation used to express the concepts. However it is important that you do not change notations part way through. For example, saying a phrase such as “The cat sat on the mat” would be hard to follow if you changed languages half way through. We will also see that some notations, like some languages, can express some ideas more clearly than others.

2. Relationships in an Entity-Relationship Diagram

We will start with the general concepts concerning relationships (associations) between entities (data objects). The properties of a relationship can be broken down into the following:

- One relationship, two directions
- Each direction, two questions
- Each question, one answer

Each of these points is dealt with in the following three sections.

One relationship, two directions

Any given relationship has meaning in two directions. First, reading from one entity to the other, and then reading back again. It is important to realise that the reading of each direction is independent of the other. If you try to read a relationship in both directions at once then sooner or later you will make mistakes in your interpretation, or representation of that relationship.

Each direction, two questions

Each direction has an entity at the beginning and at the end and these entities are reversed in the other direction. For each direction that you read the relationship, you can ask two questions. Given any entity occurrence at the beginning of the relationship:

1. Does the thing at the beginning *have to* be related to any of the things at the end of the relationship?
2. *How many* things at the end of the relationship can be related to the thing at the beginning of the relationship?

Each question, one answer

There is only a choice of two possible answers to each question. The answer to the first question is either **yes** or **no**. If the answer is yes, the minimum number of things that an entity occurrence at the beginning of the relationship can be related to is 1. If the answer is no, the minimum number of things that an entity occurrence at the beginning of the relationship can be related to is 0.

The answer to the second question is either **one** or **many**. If the answer is one then the maximum number of things that an entity occurrence at the beginning of the relationship can be related to is 1. If the answer is many then the maximum number of things that an entity occurrence at the beginning of the relationship can be related to is 1 or more.

Summary

Each question is independent of the other and for each relationship there will be a total of four questions, two in each direction. Because there are two possible answers for each question, there are 16 possible combinations of answers for each relationship (2x2x2x2). Other ways of asking the same questions are:

1. Are there any of these things that aren't associated with those things?
2. Are there any of these things that are associated with more than one of those things?

3. Different Notations for the same Relationship

All the notations we will consider use a labelled rectangle for the entity. It is the depiction of the relationship itself which varies between the different notations.

For each notation we will build up to the depiction of the complete relationship in steps. The example that will be considered is a Department which employs Employees. A department can employ several (i.e. one or more) employees. We can also have a department that currently has no employees assigned to it. Employees can only ever work for one department at a time. Furthermore, employees must be assigned to a department; they cannot be "floating". Therefore, the following questions are answered as follows:

1. Does a department *have to* employ employees? No
2. *How many* employees can be employed by a department? Many
3. Does an employee *have to* work for a department*? Yes*
4. *How many* departments can an employee work for? One

These answers can be combined to create what is known as Enterprise Rules; one for each direction of the relationship:

1. A department employs zero, one or many employees.
2. An employee works for one and only one department.

We will take this relationship and look at how it would be represented in a series of different notations. We will consider: Chen (as used in David Howe (2001)), SSADM v3, SSADM v4 and the UML Class Diagram.

4. Chen Notation

‘The Chen notation (as used in David Howe (2001)) shows a relationship as a lozenge (‘diamond’ shape) connected with lines to the entities. The name of the relationship is shown in the lozenge. The name of the relationship is usually written to make sense reading right from left, or top to bottom. With this notation we are left to infer the name of the relationship reading in the other direction. For example, in figure 1.1 the relationship reads “Department *employs* Employee” from right to left, then we are left to infer that it may read “Employee *works for* Department”.

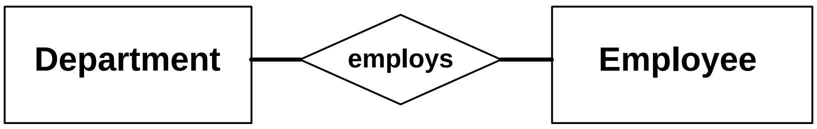


Figure 1.1: Chen ERD showing an ‘employs’ relationship between the two entities: Department and Employee

The ERD depicted in figure 1.1 is not yet complete. Previously we concluded that we can have a department that currently has no employees assigned to it. In other words, a department does not *have to* employ any employees. We show this on the Chen ERD by putting a dot outside of the Department entity box at the start of the relationship (as we are reading it). This dot is shown in figure 1.2. In other words we are reading from department to employee, therefore the dot is put just outside of the department rectangle.

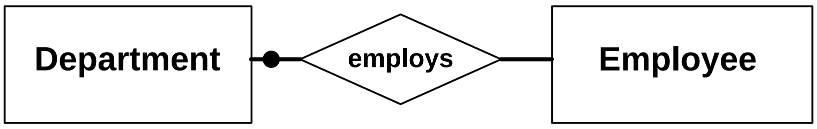


Figure 1.2: Chen ERD with dot showing that a department might not employ any employee.

However, if we concluded the opposite, that is, a department cannot have no employees assigned to it, or in other words, every department must have at least one employee, then the dot is placed inside the department entity box. This is shown in figure 1.3. Notice to emphasise the fact that the dot is on the inside the box a small box is sometimes shown around the dot.

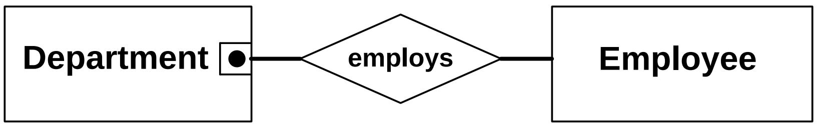


Figure 1.3: Chen ERD with dot showing a department must have at least one employee.

Continuing in the same direction, from department to employee, we also concluded that a department can employ several (i.e. one or more) employees. This is shown as an “m” at the far end of the relationship as we are reading it. Strictly speaking “m” is a place holder for any number greater or equal to one, but is often used as shorthand for meaning ‘many’. Figure 1.4 shows this rule (and only this rule).

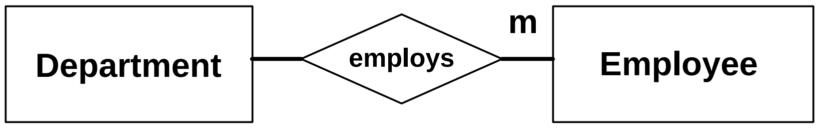


Figure 1.4: Chen ERD with “m” showing that a department might employ one or more employees.

However, if we concluded the opposite, that is, a department cannot employ more than one employee, or in other words, a department can employ at most one employee, then a “1” is placed at the far end of the relationship instead. This is shown in figure 1.5.

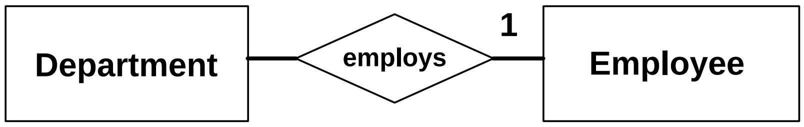


Figure 1.5: Chen ERD with “1” showing that a department employs at most one employee.

We eventually combine the rules on the same diagram. To do this, all the appropriate annotations are simply put on the relationship at the same time. The findings so far are shown in figure 1.6. This completes the relationship in the direction from department to employee and represents the enterprise rule in that direction. It states that a department employs none, one or more employees. The “none bit is represented by the dot and the “one or more” is represented the “m” symbol.

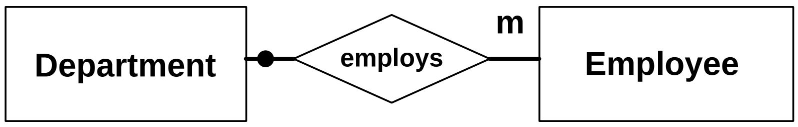


Figure 1.6: Chen ERD showing that each department employs none, one or more employees.

In the other direction, from employee to department, there is a different enterprise rule. It states that an employee works for one and only one department. An employee “must” work for a department and so a dot is placed inside the employee entity box at the starting end of the relationship as we read it (right to left). The “and only one” part of the rule is shown as a “1” at the far end of the relationship as we are reading it. Figure 1.7 shows this enterprise rule from employee to department.

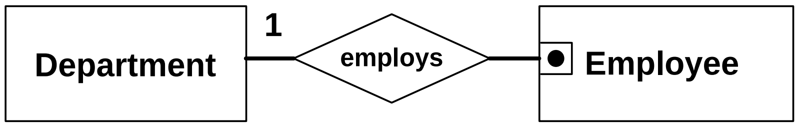


Figure 1.7: Chen ERD showing an employee works for one and only one department

The ERD is completed by showing both directions at the same time. To do this, all the annotations are added to the diagram. In other words the intermediate figures (in particular figures 1.6 and 1.7) are merged together. The result is shown in figure 1.8.

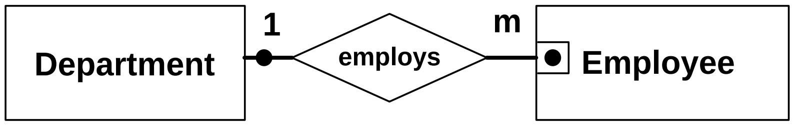


Figure 1.8: Chen ERD completed showing both enterprise rules.

Problems with the Chen Notation

We have already encountered the inconvenience of having to infer the name of the relationship in the other direction from which it is stated. In some other notations, the name of the relationship is given for both directions.

Furthermore, for each direction, the “1” or “m” is at the other end of the dot. This separation can make the standard reading of an enterprise rule difficult. To overcome this problem, you may find it easier to read the enterprise rule in a slightly different way. Currently we are saying “Each… none, …”, and “Each … one …” for the minimum multiplicity (the dot). Instead, let’s use ‘may’ or ‘must’. In this form we would read the relationship in figure 1.8 as:

1. A department might employ one or more employees.
2. An employee must work for only one department.

This way we are reading the annotations as we encounter them. Looking as figure 1.8, we start at Department, read may or must, then the name of the relationship (employs), then ‘one’ or ‘one or more’ then Employee. Then we go back and read the relationship in the other direction.

5. SSADM version 3 Notation

The SSADM version 3 notation uses a solid line to show the relationship between two entities. The role an entity plays in a relationship (its relationship name) is shown at the start end of the relationship for the direction we are reading it in. See figure 2.1.

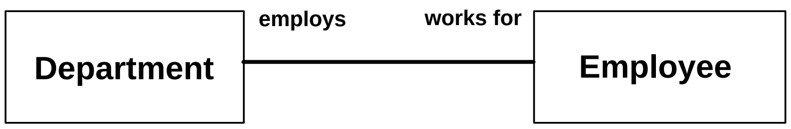


Figure 2.1: SSADM v3 ERD showing an 'employs' relationship role from Department to Employee and a 'works for' relationship role from employee to Department

We need to add rules to the ERD depicted in figure 2.1. We can have a department that currently has no employees assigned to it. We show this by putting a small circle at the far end of the relationship (as we are reading it). This rule (and only this rule) is shown in figure 2.2.

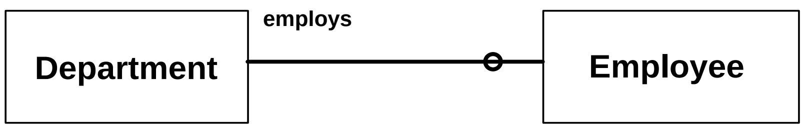


Figure 2.2: SSADM v3 ERD with circle showing that a department might not employ any employee.

However, if we concluded the opposite, that is, every department must have at least one employee, then a vertical pipe is placed at the far end of the relationship instead. This is shown in figure 2.3.

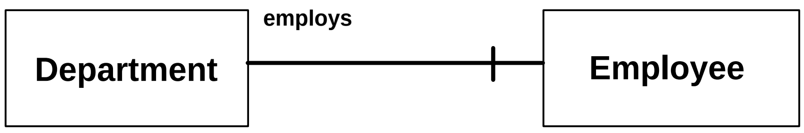


Figure 2.3: SSADM v3 ERD with vertical pipe showing that a department must have at least one employee.

Continuing in the same direction, from department to employee, we also concluded that a department can employ one or more employees. This is shown as a "crows-foot" at the far end of the relationship as we are reading it. Figure 2.4 shows this rule. Note that version 4 of SSADM also uses this "crows-foot" notation for the same purpose.

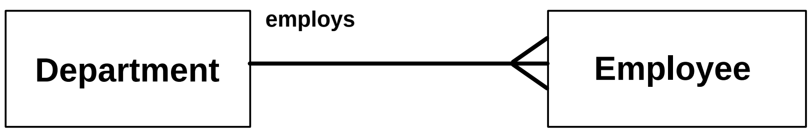


Figure 2.4: SSADM v3 ERD with a "crows-foot" showing that a department might employ one or more employees.

However, if we concluded the opposite, that is, a department can employ at most one employee, then a vertical pipe is placed at the far end of the relationship instead. This is shown in figure 2.5. Notice how very similar this is to figure 2.3 which has a different meaning.

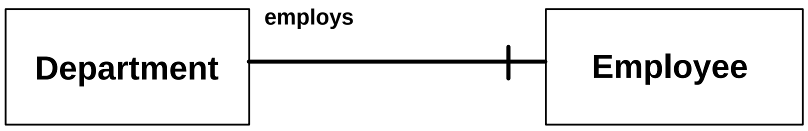


Figure 2.5: SSADM v3 ERD with vertical pipe showing that a department employs at most one employee.

Figure 2.6 shows the enterprise rule in the department to employee direction. It states that a department employs none, one or more employees. The "none" bit is represented by the circle and the "one or more" is represented the "crows-foot". Note that all the multiplicity is shown at the far end of the relationship for the direction we are reading it in, so here the circle is just before the "crows-foot".

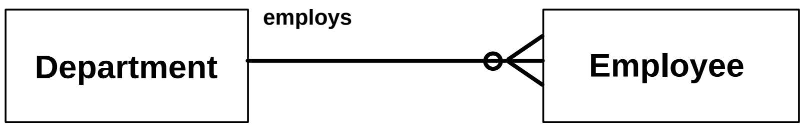


Figure 2.6: SSADM v3 ERD showing that each department employs none, one or more employees.

You may have already noticed that SSADM version 3 uses a vertical pipe to indicate both a minimum multiplicity of one and a maximum multiplicity of one. Thus the "one and only one" annotation will consist of two vertical pipes at the far end of the relationship as you are reading it. Figure 2.7 shows the enterprise rule from employee to department depicting the "one and only one" rule.

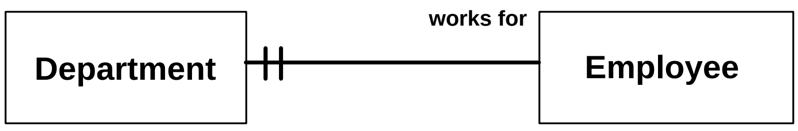


Figure 2.7: SSADM v3 ERD showing an employee works for one and only one department.

Again, the ERD is completed by showing both directions at the same time. In other words the figures 2.6 and 2.7 are merged together to produce figure 2.8.

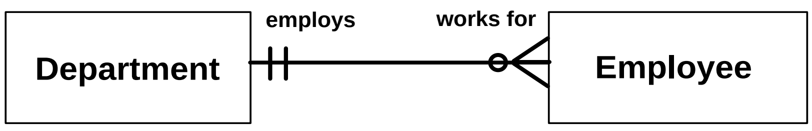


Figure 2.8: SSADM v3 ERD completed showing both enterprise rules.

Problems with the SSADM version 3 Notation

The SSADM version 3 notation includes a source of ambiguity. The vertical pipe is used to indicate both a minimum multiplicity of one and a maximum multiplicity of one. Notice how very similar figures 2.3 is to figure 2.5 (in fact they could be identical). However, figure 2.3 and 2.5 are meant to depict different meanings (different rules). Consequently, during the design process of building the ERD, misunderstandings can occur until the ERD is complete.

6. SSADM version 4 Notation

Other notations use solid lines to depict the existence of a relationship between two entities. In SSADM version 4 there is no notation to depict a relationship without working out its multiplicity. However, as with version 3, the role an entity plays in a relationship (its relationship name) is shown at the start end of the relationship for the direction we are reading it in (figure 3.1).

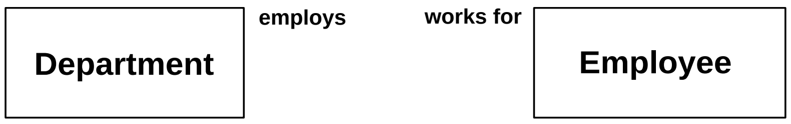


Figure 3.1: SSADM v4 ERD showing an ‘employs’ role from Department to Employee and a ‘works for’ role from employee to Department.

Adding minimum multiplicity to figure 3.1 will connect the two entities together. The rule that we can have a department that currently has no employees assigned to it represents a minimum multiplicity of none. Instead of a dot, circle or zero, version 4 of SSADM uses a dashed line at the starting end of the relationship reaching the midpoint between the two entities (figure 3.2). This symbolises the partial participation of department in the relationship because some occurrences of department do not currently employ employees.

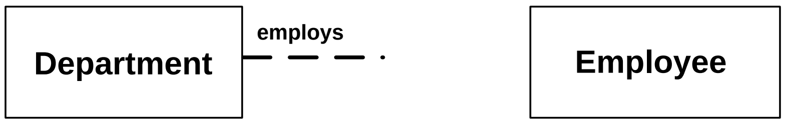


Figure 3.2: SSADM v4 ERD with dashed line showing that a department might not employ any employee.

However, say alternatively we wanted a minimum multiplicity of one, that is, every department must have at least one employee. Instead of a dot inside the entity, or a vertical pipe, or a number one, a solid line (which depicts the existence of a relationship in the other notations) is used at the starting end of the relationship. This is shown in figure 3.3. The solid line symbolises full participation of department in the relationship because every occurrence of department must have an employee. Compare figure 3.3 to figure 3.2.

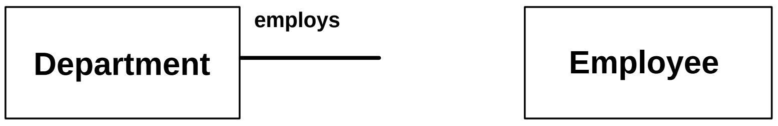


Figure 3.3: SSADM v4 ERD with solid line showing that a department must have at least one employee.

Continuing in the same direction, from department to employee, we also concluded that a department can employ one or more employees. Just as in version 3, this is shown as a “crows-foot” at the far end of the relationship as we are reading it (figure 3.4). This represents a maximum multiplicity of one or more.

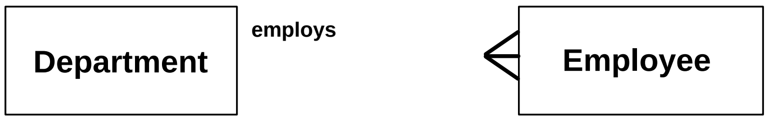


Figure 3.4: SSADM v4 ERD with a "crows-foot" showing that a department might employ one or more employees.

However, say alternatively we wanted a maximum multiplicity of one, that is, a department can employ at most one employee. Instead of the presence of a number one, or a vertical pipe, the "at most one" is implied by the absence of a "crows-foot" (which would indicate the opposite: "one or more"). This absence is depicted in figure 3.5! Compare this to figure 3.4.

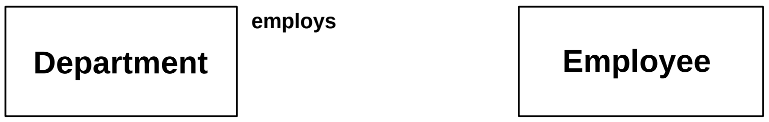


Figure 3.5: SSADM v4 ERD with a role name but the absence of a "crows foot" indicating that a department employs at most one employee!

Figure 3.6 shows the enterprise rule in the department to employee direction. It states that a department employs none, one or more employees. The "none" bit is represented by the dashed line and the "one or more" is represented the "crows-foot". Unlike version 3, but like the Chen notation, for each direction the minimum and maximum multiplicity are separated; each depicted at the other end of the relationship.

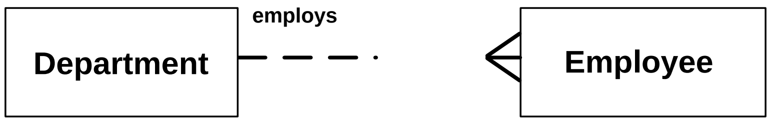


Figure 3.6: SSADM v4 ERD showing that each department employs none, one or more employees.

In the other direction, from employee to department, the enterprise rule is that an employee works for one and only one department. Every department must have at least one employee and so a solid line is drawn at the starting end of the relationship as we read it (right to left). The "at most one" part of the rule is implied by the absence of a "crows-foot" at the far end of the relationship. Figure 3.7 shows this enterprise rule.

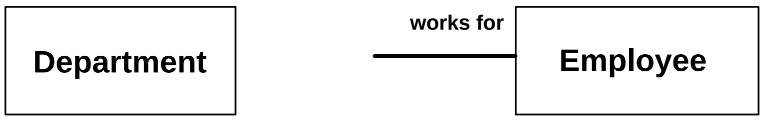


Figure 3.7: SSADM v4 ERD showing an employee works for one and only one department.

The ERD is completed by showing both directions at the same time. In other words the figures 3.6 and 3.7 are merged together to produce figure 3.8. Notice that it is only now, after the multiplicity in both directions has been decided, that the two entities are connected by lines.

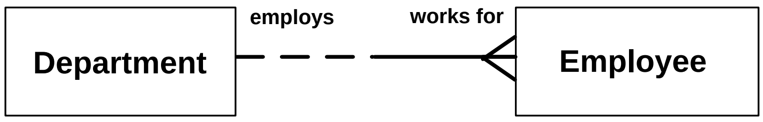


Figure 3.8: SSADM v4 ERD completed showing both enterprise rules.

Problems with the SSADM version 4 Notation

Some notation (found in Chen and SSADM version 3) is absent in SSADM version 4. There is no notation to depict a relationship without working out its minimum multiplicities. Consequently, this adds complications during the design process of building the ERD because an initial step is to identify a relationship and name it before deciding on its multiplicity.

Also there is no notation to represent a maximum multiplicity of one. This is implied by the absence of a “crows-foot” (which would indicate the opposite: “one or more”). This absence can be a source of ambiguity during the design process because a maximum multiplicity of one (depicted as nothing) could be interpreted as a rule has not yet been decided. Conversely, if the absence does in fact mean that a rule has not yet been decided then it might be interpreted as a maximum multiplicity of one!

Furthermore, like the Chen notation, for each direction the minimum and maximum multiplicity are separated; each depicted at the other end of the relationship. This separation can make the reading of an enterprise rule difficult. As with Chen, this problem can be overcome by using the words ‘may’ and ‘must’ instead of ‘none’ and ‘one’ when reading back the rules. See section on problems with the Chen notation for more details.

7. UML Class Diagram Notation

The UML class diagram, as the name implies, is not strictly an entity relationship diagram because it depicts classes, not entities. However the multiplicity notation on a UML class diagram is read in the same way as the Chen and SSADM notations. The popularity of UML has meant that its class diagram has been used for database design as well as, its original purpose, object-oriented programming design. This will work as long as the purpose of the design at any stage is not confused.

With the UML class diagram the relationship is shown as a solid line between the two boxes. As with the SSADM version 3 diagram, the roles that each data object plays in the relationship is shown at the start end of the relationship for the direction you are reading it (figure 4.1)..

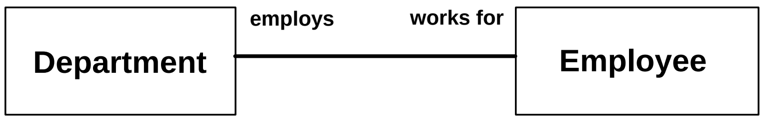


Figure 4.1: UML Class Diagram showing an ‘employs’ role from Department to Employee and a ‘works for’ role from employee to Department.

The rule that we can have a department that currently has no employees assigned to it is represented as a minimum multiplicity of none. Instead of a dot, circle or dashed line, UML uses a zero followed by a few full-stops at the starting end of the relationship. This rule (and only this rule) is shown in figure 4.2. This symbolises a range of values starting at zero. It is zero because some occurrences of department might currently employ zero employees.

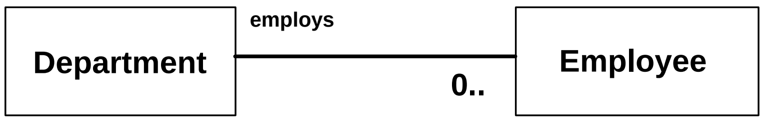


Figure 4.2: UML Class Diagram with start of range zero showing that a department might not employ any employee.

However, if we concluded the opposite, that is, every department must have at least one employee then this is represented as a minimum multiplicity of one. In UML this is depicted as a one followed by a few full-stops at the starting end of the relationship. This is shown in figure 4.3. This symbolises a range of values starting at one. Therefore the range cannot contain zero because every occurrence of department must employ at least one employee.

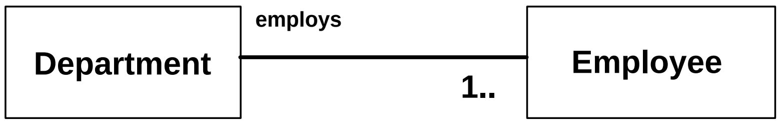


Figure 4.3: UML Class Diagram with start of range one showing that a department must have at least one employee.

Continuing in the same direction, from department to employee, we also concluded that a department can employ one or more employees. This is represented as a maximum multiplicity of one or more. In UML this is shown as an asterisk preceded by a few full-stops at the starting end of the relationship. This symbolises a range of values ending at one or more (many). See figure 4.4.

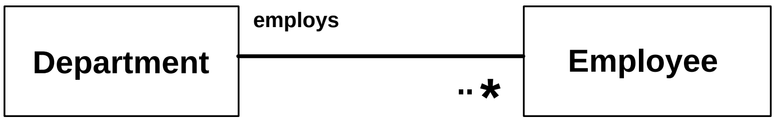


Figure 4.4: UML Class Diagram with an end of range asterisk showing that a department might employ one or more employees.

However, if we concluded the opposite, that is, a department can employ at most one employee, then this is restricted to a maximum multiplicity of one. This is shown as a one preceded by a few full-stops at the starting end of the relationship. Comparing figure 4.3 and 4.5 you can see that the range delimiter ".." is very important for distinguishing minimum and maximum multiplicity in UML class diagrams.

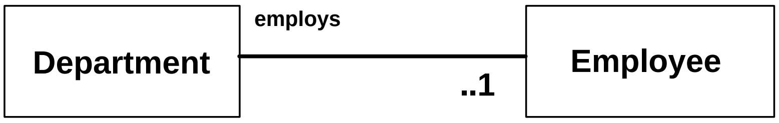


Figure 4.5: UML Class Diagram with an end of range one showing that a department employs at most one employee.

It is time to combine the rules on the same diagram. In other notations we would simply put all the appropriate annotations on the same relationship. If we did this for the Class Diagram we would merge "0.." from figure 4.2 and "..*" from figure 4.4 to make the range "0..*". This would state that a department employs none, one or more employees. The "none bit is represented by the zero and the "one or more" is represented the asterisk. As in SSADM version 3, all the multiplicity in one direction is shown together at the far end of the relationship as we read it. However, in UML "0..*" is shortened to just * as a shorthand reference to the range (figure 4.6). This must not be confused with "..*". This is another reason why the presence or absence of the range delimiter is important.

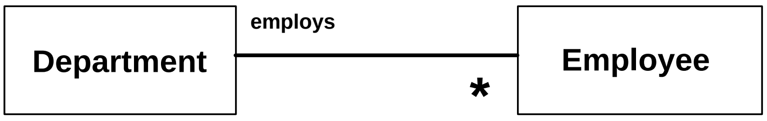


Figure 4.6: UML Class Diagram with an asterisk showing that each department employs none, one or more employees.

In the other direction, from employee to department, the rule states that an employee works for one and only one department. We could merge "1.." from figure 4.3 and "..1" from figure 4.5 to make the range "1..1". However, UML has a second shorthand notation where "1..1" is shortened to simply "1". This is shown in figure 4.7. Note that the other ranges: "0..1" and "1..*" are not shortened.

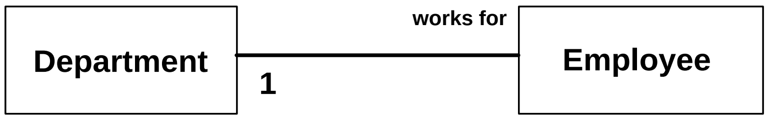


Figure 4.7: UML Class Diagram showing an employee works for one and only one department.

Finally, the ERD is completed by showing both directions at the same time. In other words the figures 4.6 and 4.7 are merged together to produce figure 4.8.

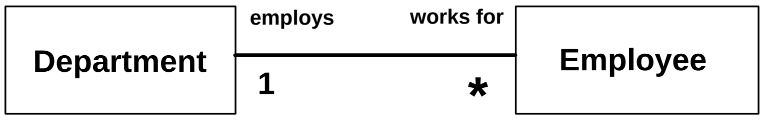


Figure 4.8: UML Class Diagram completed showing both enterprise rules.

Problems with the UML Class Diagram Notation

The UML class diagram is predominantly used to design associations between classes. Classes are very different to entities. Classes are a means of organising software not specifically data. It is very important that a UML class diagram whose purpose was to design object-oriented programming code is not confused with the design of a relational database. The posting of identifiers in a class diagram can be very different to that in an entity-relationship diagram because it depends on the pattern required for organising the object-oriented code, not normalisation. Therefore, the purpose of a UML class diagram must not be confused.

The UML notation is very subtle. For example, the range delimiter ".." is very important. If it is omitted by mistake then it can lead to misinterpretations. "1", "1.." and "..1" all have different meanings, as do "*" and "..*".

8. Selective Summary of All Notations

A one-to-one, one-to-many and many-to-many relationship have been selected to summarise the notations. Of these three multiplicities, the one-to-many combination is by far the most common in practice. One-to-one relationships are rare and many-to-many relationships must be decomposed to produce one-to-many relationships. The relationships will be summarised in terms of their enterprise rules. For further details refer back to earlier chapters.

One-to-one, optional at one end, mandatory at the other

We will look at a one-to-one where it is optional at one end but not optional at the other. The enterprise rules are as follows:

- Each department employs none, or one employee.
- Each employee works for one and only one department.

This is represented in figures 5.1, 5.2. 5.3 and 5.4, one figure for each of the different notations. Colour coding has been added to aid the comparison between figures.

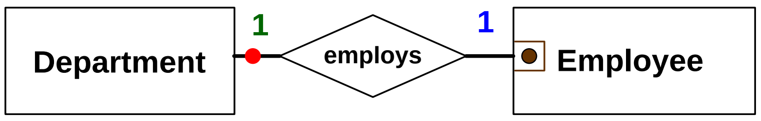


Figure 5.1: Chen ERD notation showing that a department employs none, or one employee, and an employee works for one and only one department.

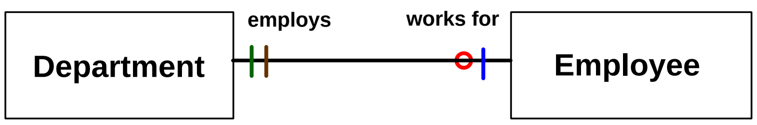


Figure 5.2: SSADM v3 ERD notation showing that a department employs none, or one employee, and an employee works for one and only one department.

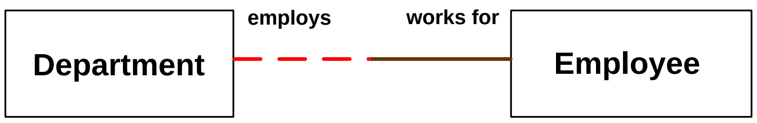


Figure 5.3: SSADM v4 ERD notation showing that a department employs none, or one employee, and an employee works for one and only one department.

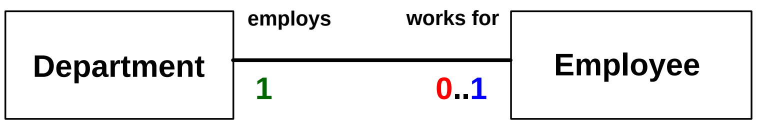


Figure 5.4: UML Class Diagram notation showing that a department employs none, or one employee, and an employee works for one and only one department.

One-to-Many, mandatory at one end, optional at the other

We will look at a one-to-many where it is mandatory at one end but optional at the other. This is a very common combination. The enterprise rules are as follows:

- Each department employs none, one or more employees.
- Each employee works for one and only one department.

This is represented in figures 5.5, 5.6. 5.7 and 5.8, one figure for each of the different notations.

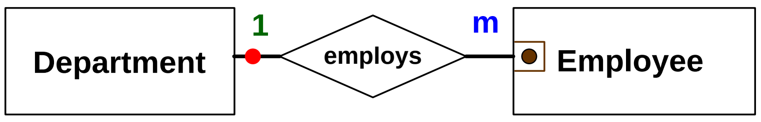


Figure 5.5: Chen ERD notation showing that a department employs none, one or more employees, and an employee works for one and only one department.

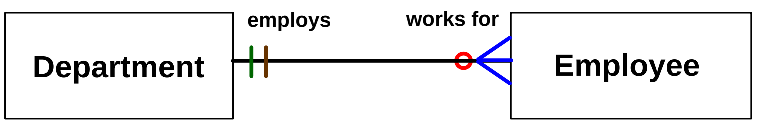


Figure 5.6: SSADM v3 ERD notation showing that a department employs none, one or more employees, and an employee works for one and only one department.

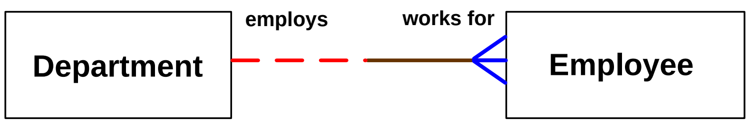


Figure 5.7: SSADM v4 ERD notation showing that a department employs none, one or more employees, and an employee works for one and only one department.

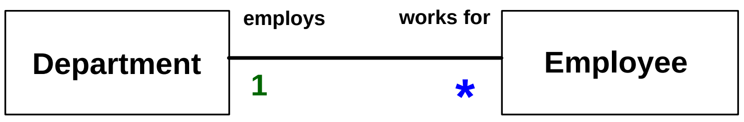


Figure 5.8: UML Class Diagram notation showing that a department employs none, one or more employees, and an employee works for one and only one department.

Many-to-Many, optional at both ends

We will look at a many-to-many relationship where it is optional at both ends. Although many-to-many relationships are fairly common during the design process, in the final model each must be decomposed into two one-to-many relationships. The enterprise rules are as follows:

- Each department employs none, one or more employees.
- Each employee works for none, one or more departments.

This is represented in figures 5.9, 5.10. 5.11 and 5.12, one figure for each of the different notations.

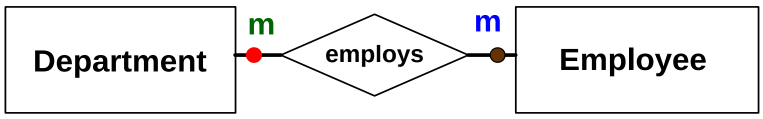


Figure 5.9: Chen ERD notation showing that a department employs none, one or more employees, and an employee works for none, one or more departments.

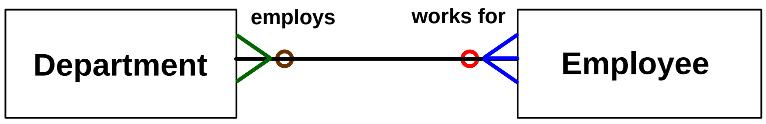


Figure 5.10: SSADM v3 ERD notation showing that a department employs none, one or more employees, and an employee works for none, one or more departments.

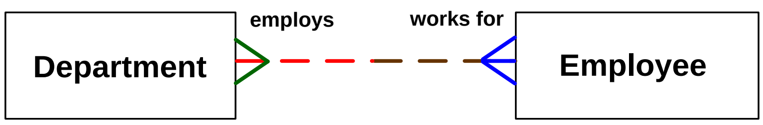


Figure 5.11: SSADM v4 ERD notation showing that a department employs none, one or more employees, and an employee works for none, one or more departments.

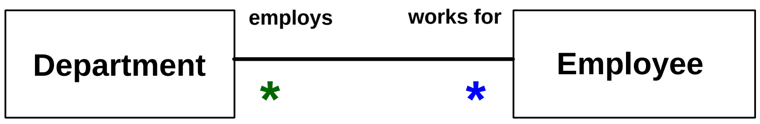


Figure 5.12: UML Class Diagram notation showing that a department employs none, one or more employees, and an employee works for none, one or more departments.

9. Exercises

Consider the six different examples of entities and relationship shown in figures 6.1 to 6.6. The figures all use the same entities and relationship, but show varying multiplicity combinations. When you have looked at, and understood each of the figures, answer the following questions

1. Write down the two enterprise rules of each relationship. Remember each direction represents one enterprise rule.
2. Use the multiplicity presented in figure 6.5 to answer the following questions:
    - Does an employee have to work on a project?
    - Could an employee work on more than one project?
    - Could there be a project which has no employees working on it?
    - Could there be a project with 15 employees working on it?
3. Which two figures are equivalent, that is, show the same multiplicity in each direction and therefore represent the same enterprise rules.
4. Draw four different ERDs, one for each of the different notations, to represent the following enterprise rules:
    - Each dog has one and only one day.
    - Each day may be for none, one or more dogs.

Specimen answers are available at the end of this document.

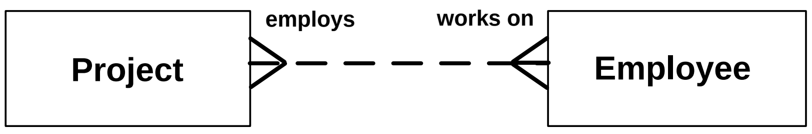


Figure 6.1: SSADM v4 ERD notation.

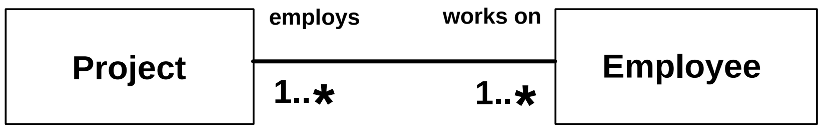


Figure 6.2: UML CD notation.

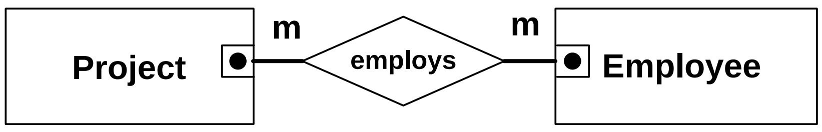


Figure 6.3: Chen ERD notation.

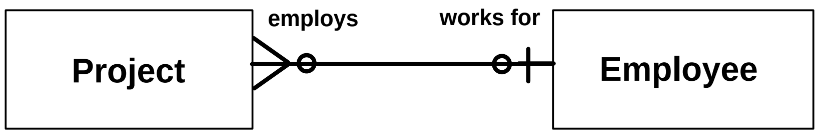


Figure 6.4: SSADM v3 ERD notation.

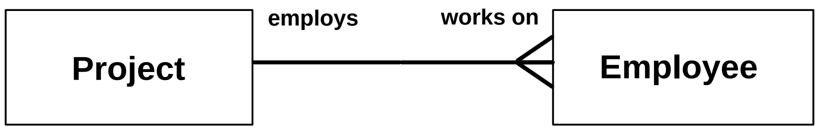


Figure 6.5: SSADM v4 ERD notation.

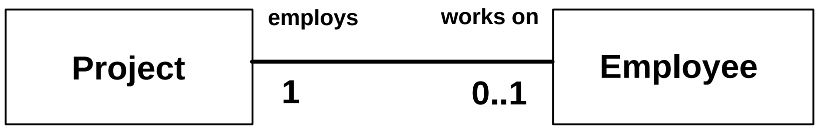


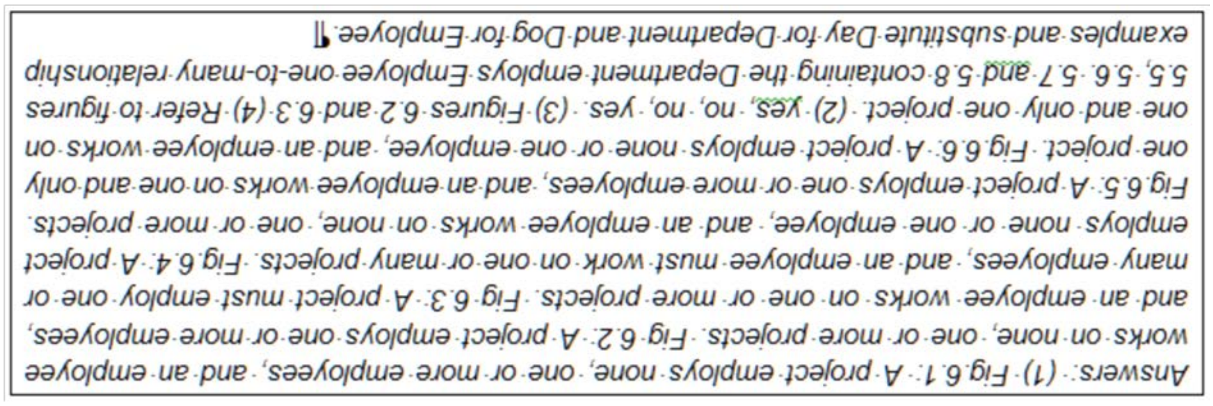
Figure 6.6: UML CD notation.

10.Bibliography

Bennett, S., McRobb, S. and Farmer, R. (1999) *Object-Oriented Systems Analysis and Design using UML,* McGraw-Hill.

Downs, E. (1992) *Structured Systems Analysis and Design Method Application and Context*, Prentice Hall.

Howe, D. (2001) *Data Analysis for Database Design*, Butterworth Heinemann.



Answers: (1) *Fig 6.1: A project* employs none, one or more employees, and an employee works on none, one or more projects. *Fig 6.2: A project* employs one or more employees, and an employee works on one or more projects. *Fig 6.3: A project* must employ one or many employees, and an employee must work on one or many projects. *Fig 6.4: A project* employs none or one employee, and an employee works on none, one or more projects. *Fig 6.5: A project* employs one or more employees, and an employee works on one and only one project. *Fig 6.6: A project* employs none or one employee, and an employee works on one and only one project. (2) yes, no, no, yes. (3) Figures 6.2 and 6.3 (4) Refer to figures 5.5, 5.6, 5.7 and 5.8 containing the Department employs Employee one-to-many relationship examples and substitute Day for Department and Dog for Employee.¶