Entity-Relationship Modelling

3 Entity-Relationship Modelling

9. Identify entities. The first step is to find entities that we wish (or that the customer asks us) to represent in the database.
Typical examples:

- **People:** staff, clients/customers, patients, members, owners, contacts, other individuals, ...
- **Objects:** stock items, real estate, offices, ...
- **Organisations:** firms (suppliers or customers), departments, charities, clubs, committees, ...
- **Object classes:** recordings, films, books, types of stock, biological species, work roles, ...
- **Events:** concerts, examinations, lecture courses, consultations, sales, ...
- ⋮

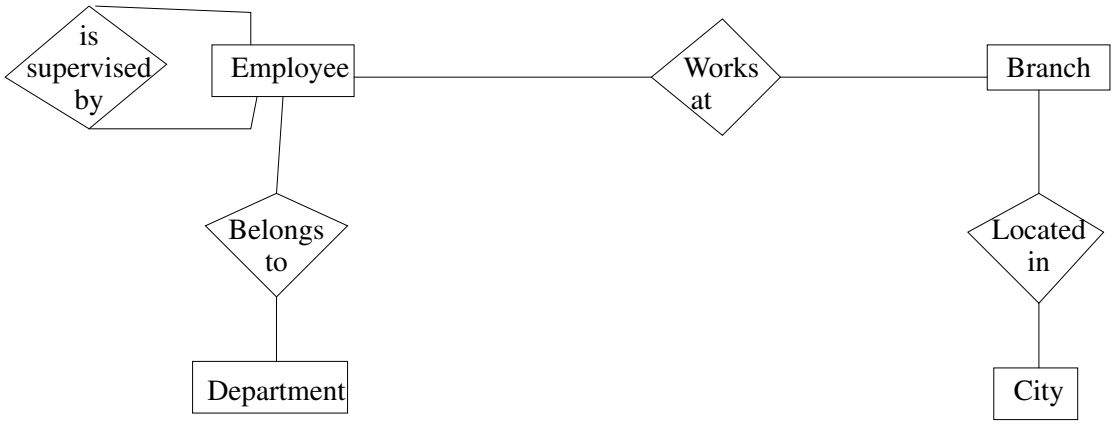
Often, the main entities appear as *nouns* in a natural language specification. However, a *typical mistake* of beginners in ER-modelling is to include all nouns as database entities. In each case, you must check whether there is more than one instance of the entity — otherwise, why use a table instead of a constant? Even if there are several instances, they may be totally fixed in advance, so for example, the days of the week are an unlikely entity set. Finally, you must check whether there is anyone really interested in the instances — otherwise we are cluttering the database with irrelevant information.

10. Identify relationships. Find relationships between entities that need to be recorded:
Typical examples:

- **Ownership:** person *owns* object
- **Lines of command:** person *supervises* person
- **Participation:** person *participates in* event
- **Part of relationship:** item *is part of* order
  person *belongs to* organisation
- **Location:** house *is located in* region
- **Personal:** person *is married to* person
  person *is parent of* person
- ⋮

Often, relationships are expressed by *verbs*.
Relationships can involve more than two entities, for example, an examination involves an examiner, a course, and a student; a property sale involves a seller, a buyer, a property, an estate agent, and two solicitors. These *higher-order* relationships are rare.
Sometimes it may be difficult to decide whether to use a relationship or an entity. The good thing about this style of modelling is that *it does not matter* — it will usually result in the same tables anyway.

11. ER diagram. Draw a diagram where entities are represented as rectangles, relationships as diamonds. Small example:



Note: Beauty is not an objective here, so don't bother to waste time with a drawing program. Hand-drawn diagrams will always be fine. Clear layout, however, is very important, so try to avoid crossing lines etc.

12. Collect attributes for entities.

- Together, the set of attributes *must identify* the real-world entity that is represented. Typically, therefore, entities have *many* attributes.
- Additionally, you may wish to introduce an artificial identifier as the primary key.
- Attributes can be complex: *date* consists of day of the month, month, and year; *address* consists of street, house number, parish, city, region, postcode, country, continent, planet, solar system, galaxy, etc.; *duration* consists of start date and end date. Don't worry about complex attributes at this stage in the design.
- Attributes can be of varying structure: *children* can be a list of persons of varying length; *shipment* can consist of a varying number of items. Don't worry about this at this stage.

13. Collect attributes for relationships.

- Do *not* include keys of participating entities because these will be added automatically at a later stage.
- Often, the additional attributes refer to time: worked at branch *since* . . . ; was married to . . . *from* . . . *until*
- It is normal to have *very few* attributes attached to a relationship. Indeed, it is often the sign of an erroneous model if you find that a lot of information should be recorded in a relationship table.

14. Recording attributes in the design. Some textbooks recommend to have attributes represented in the diagram; only do this if it clarifies the meaning of an entity (or relationship) name. Otherwise, list the attributes separately in the following form:

> Employee(<u>eid</u>, first_name, middle_name, last_name, date_of_birth, home_address,
> national_insurance_number, first_day_of_employment, . . .)

(Underline the attribute(s) that you wish to use as primary key.)
Note that we do not include attributes which refer to other entities in the database, such as, department, line manager, etc., because the whole point of the modelling phase is to find the correct table where this information should be stored.

15. Cardinalities in relationships. For every entity and every relationship, record the (expected) number of times that a particular instance of that entity will take part in the relationship. We distinguish four different cases:

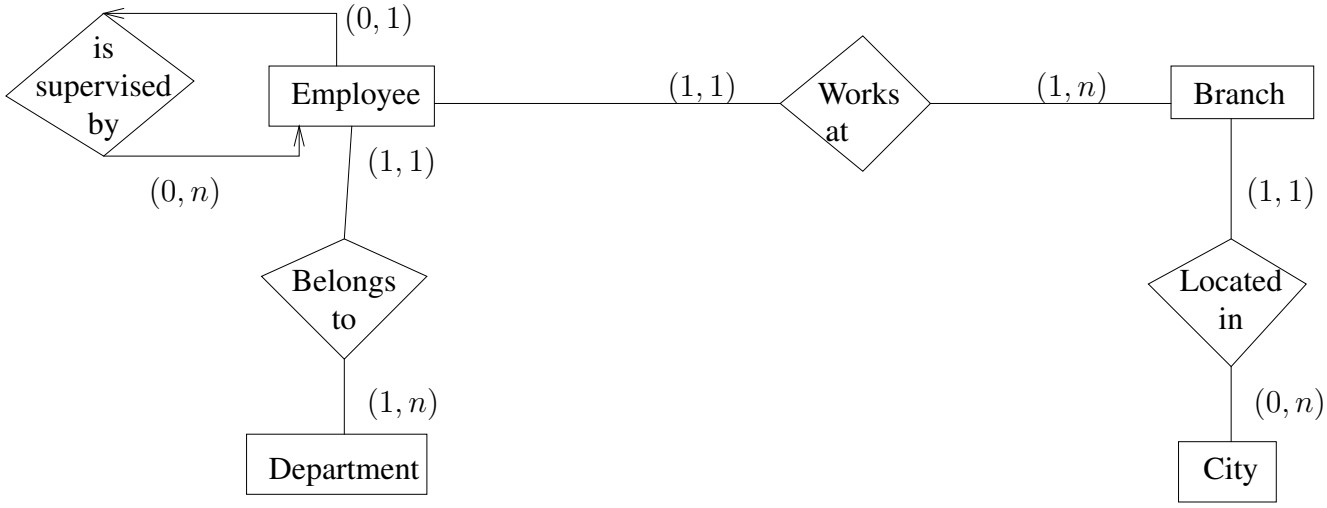
(1,1) The instance must and will appear exactly once in the relationship. For example, every order was issued by exactly one customer. We record this multiplicity as $(1, 1)$, meaning, "at least once and at most once".

(0,1) The instance appears at most once in the relationship. For example, although every person has exactly one father and one mother we may not be able to record their identities. We record this as $(0, 1)$, meaning, "possibly not, but at most once".

(1,*n*) The instance appears at least once but can appear many times. As an example, every piece of music has at least one performer but there could be many musicians involved in its recording. Likewise, a bank account has at least one owner but often a spouse is recorded as second owner with equal rights. This is recorded as $(1, n)$ where we usually don't bother to make the n more specific even if we know that there is an upper bound. It would not make any difference to the tables generated.

(0,*n*) The instance may appear many times or not at all. This is the most common situation: A customer may have placed many or no order in a given time interval; a couple may have many or no children at all, etc. We record this as $(0, n)$.

Adding multiplicities to the diagram above we get:



Let's read these multiplicities and see whether they capture the real world situation correctly.

- An employee may or may not be supervised but if (s)he is, then there is only one supervisor. This is an *assumption* about this organisation which may not be valid in other companies.
- An employee may or may not be supervisor to other employees. If (s)he is, then there can be many subordinates.
- Every employee works at precisely one branch. A branch has at least one employee working there.
- A branch is located in precisely one city, and a city may have no branch or several branches of this company.
- Every employee works for a particular department, and a department has at least one member of staff.

Important! Multiplicities will allow us to optimise the database but the price to be paid is a restriction on what the database can store. For example, in the organisation modelled above, it will not be possible to record a second supervisor for an employee.

16. Exercises Below, you are asked to build conceptual schemata for three application domains.[1] In each case, you should record the steps in the development of the schema as explained in this section, and not just return the final diagram.

Your model should contain entities, relationships, and attributes. Indicate which attributes you think should serve as primary keys. Annotate the links between entities and relationships with multiplicity bounds. (Some descriptions are giving very little information on required attributes. Try to make a good guess.)

You will notice that the descriptions contain ambiguities and do not completely specify every aspect of the respective real-world setting. It is an important part of the design process to remove ambiguities and to delineate the scope of the database. In practice, you would do this in cooperation with your customers, but here you will have to make assumptions. Please record these assumptions in your answer.

[1] These text have been taken from the excellent book on *Conceptual Database Design* by Batini, Ceri, and Navathe, of which there are copies in the Main Library.

a. Design a database system for a College of Pharmacy, Division of Clinical Pharmacokinetics. The division has research projects in various stages of development: current, pending, and complete. Each project is funded by a single grant. Usually, the major portion of this research grant is used to pay the study's subjects; the various drugs and equipment used in the experiments are often provided by one or more drug companies. A project studies the effects of the drug on several subjects, some of whom have unequal therapeutic regimes. Several employees of the College of Pharmacy work on each project, which is led by a principal investigator; each principal investigator may control several projects. When a study is completed, a research report is written, describing the results of the study.

b. Design a database system for managing information about routes supported by a bus company. Each route served by the company has a starting place and an ending place, but it can go through several intermediate stops. The company is distributed over several branches. Not all the cities where the buses stop have a branch; however, each branch must be at a city located along the bus routes. There can be multiple branches in the same city and also multiple stops in the same city. One bus is assigned by the company to one route; some routes can have multiple buses. Each bus has a driver and an assistant, who are assigned to the bus for the day.

c. Design the database for a programming support environment. In this environment programmers produce programs, which are written in given programming languages. Each program is written by a given programmer, can call other programs, and can be used by given users. Users are recognised by their log-in name; programmers are recognised by their log-in name and by their code; programs have compound names that include the program's name, the extension, and the programmer's code. Programs have a version number, a date, and a short description; some programs interact with database management systems. Each database management system maintains stored data in the form of relations, with several attributes and one primary key. Each database is defined by a database administrator, who is a programmer specialised in data management.

4 Extensions to the basic Entity-Relationship model

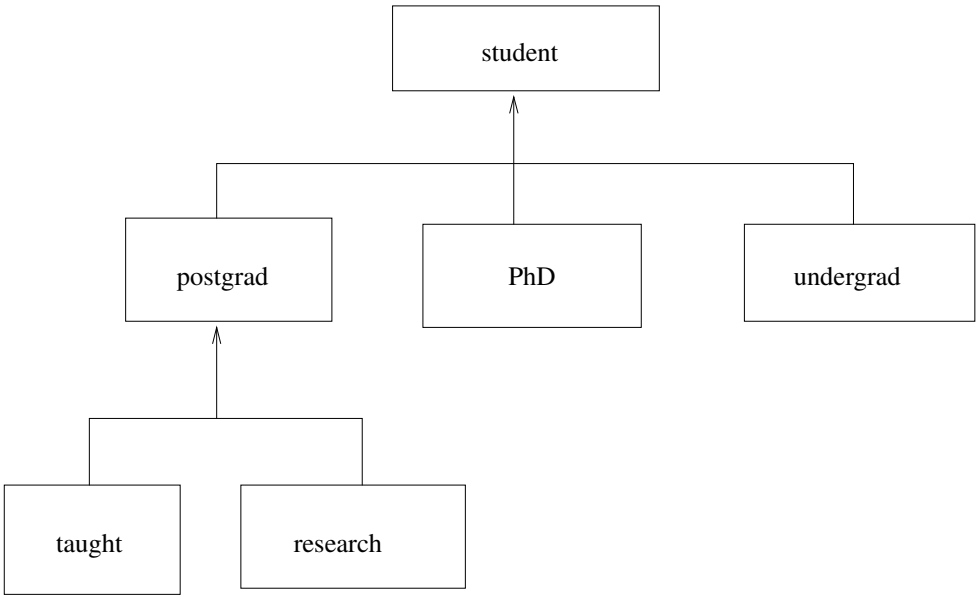
17. Generalisation hierarchies. Often we have to deal with situations where an entity splits into subentities in a natural way: A customer could be private or commercial; a student could be undergraduate, post-graduate taught, post-graduate research, PhD, or occasional; a vehicle could be a bike, a car, a lorry or something else; and on and on.

In each case we have to decide whether subentities are interesting enough from a database point of view to warrant their own representation in the design. So for example, the distinction of human beings into males and females does not necessarily require separate entities, in this case an attribute "`gender`" may well be enough. On the other hand, the separation of the "`student`" entity could be very useful because PhD students have supervisors (and so there is a relationship to members of staff) but undergraduates don't.

Another reason why you might want to make a sub- or superclass explicit is that there could be differences in the set of attributes that they support. In general, the subclass inherits all the attributes of the superclass but could have more. In the "`vehicle`" example, we might want to record engine size for cars but not for bicycles.

If you are unsure whether to make a subclass explicit or not, we recommend that you do. When we translate the diagram into actual tables there is an opportunity to consider this question again in the context of the whole design. If by that time it turns out that the subclass is not really necessary, then we will simply subsume the subclasses into the superclass and use attributes to distinguish instances from each other.

18. Representing generalisation hierarchies in ER diagrams. Sub- and superentities are each represented as rectangles and the relationship between them is indicated by lines which end in an arrowhead towards the superclass:

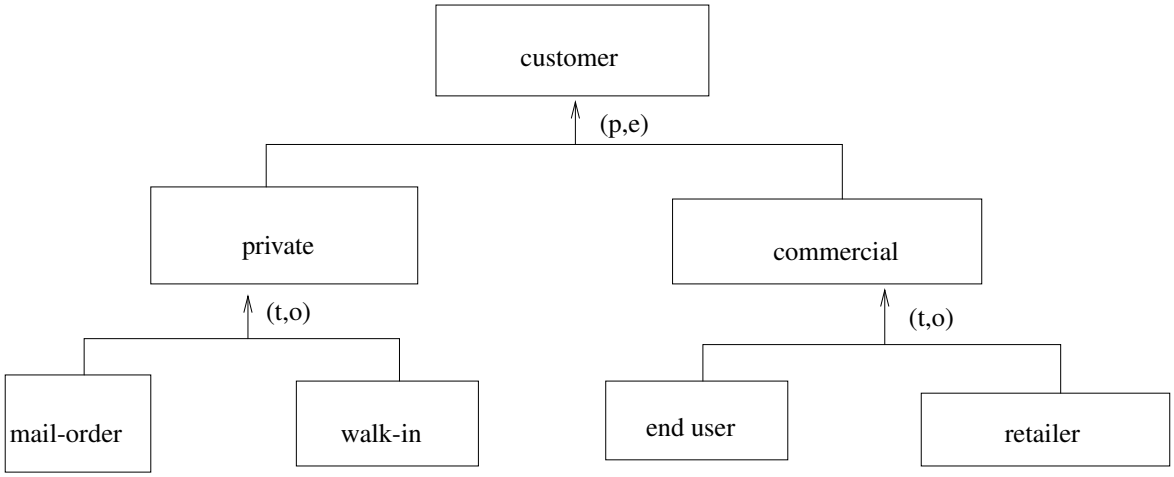


As you can see, such a generalisation tree can have several levels.

19. Annotating generalisation hierarchies. There are two questions about sets of subclasses which will help us to decide on how they should be translated into tables. The first is whether the subclasses together cover all instances of the superclass. In our examples, we can say that males and females cover all humankind. On the other hand, our list of types of vehicles surely does not cover all possibilities. Let's reserve the letter "`t`" ("total") for the first case, and "`p`" ("partial") for the second.

The other issue is whether the subclasses are mutually exclusive or not. Again, we can state with confidence that a person can not be both male and female at the same time. On the other hand, a student on a PhD programme might at the same time take some additional classes in another subject. We record this fact with the letters "`e`" ("exclusive") and "`o`" ("overlapping").

The two letters are attached to the arrowhead in a generalisation hierarchy:



We have annotated the first distinction as "partial" because there is at least one other category of customer that we can think of, namely, government bodies. We have assumed this classification to be exclusive.

In the bottom two classifications we have assumed total coverage but with overlap between the subclasses; on the left because a mail-order customer may well walk into the shop one day and we are unlikely to want to create a new customer record in that situation. Similarly, on the right it may well be that a retailer uses some of the merchandise inside their own business while the majority is sold on (say, a car dealer may use some cars for the in-house rental business).

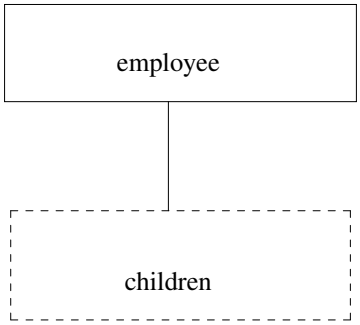
20. Conflicting classifications. There may be more than one way to classify a given entity. For example, while we can classify humans in to male and female, we can also classify them alternatively according to social status. It is all right to record these possibilities in the ER diagram without worrying too much about how they might eventually get translated into separate tables. Remember that ER modelling is all about adapting a design to the real world, not about adapting it to the needs of a computer system.

21. Weak entities. We have emphasised the need for an entity to have a clear and verifiable link to an object (or concept) in the real world on several occasions before in these notes. "Weak entities" are called "weak" precisely because this link is not of this nature. Instead, in order to relate an instance of a weak entity to the corresponding real world object, we need to have the context of another ordinary entity.

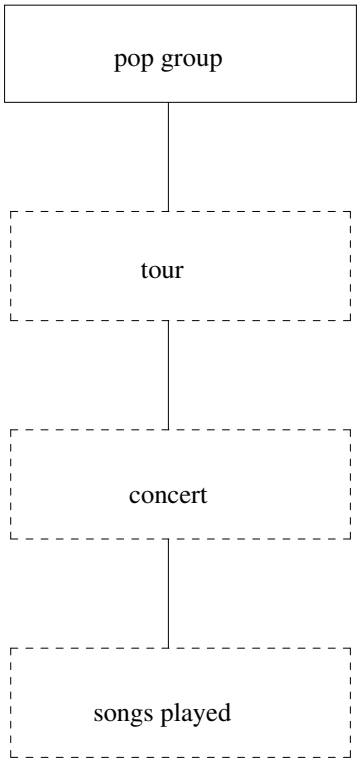
Let's look at an example. Consider a family-friendly employer who keeps a record of the children of his employees and their date of birth. This allows him to award a free afternoon to the parent whenever one of his or her children celebrates their birthday anniversary.[2] For this purpose it is enough to store first name and date of birth of each child of each employee. These two attributes, however, do not identify the children unambiguously because there may well be more than one "Mark" born on the same day. The information becomes identifying only in the context of the parent information, because we can safely assume that a person will not give the same name to identical twins.[3] Eventually, therefore, the table describing the children would have three attributes: first name, date of birth, and employee identifier (the primary key of the employee table).

Another way to view this situation is to start from the parent entity, and to consider the children as an attribute. Because the number of values of that attribute varies from one person to another (depending on the number of children they have), we can not simply have a fixed number of "`child_1`", ... , "`child_n`" attributes.

22. Representing weak entities in ER diagrams. As with all elements of ER diagrams, you will find several conventions in the literature. In this notes, we represent them in the following way:



23. Cascaded weak entities. It may well happen that a weak entity requires the context of another entity which itself requires the context of another entity. There is no limit to how far such a cascade may be nested, and the good news is that it does not cause any problems in the eventual translation into tables. Here is an example:



[2] Obviously, this example is entirely fictitious.

[3] On the other hand, a person may re-marry another divorcee and both may have children from their previous marriages. The real world really is a complicated place...

Instances at the bottom of this cascade refer to songs which were played at a certain concert during a certain tour of a certain pop group, for example, the interpretation of "Sounds of Silence" by Simon & Garfunkel in their Central Park concert on September 19 during their 1981 reunion tour.[4]

24. Exercises

a. Draw the generalisation hierarchy graph for the following concepts. Annotate the graph with coverage letters for "partial vs total" and "overlapping vs exclusive".

   i. Bird, reptile, animal, mammal, vertebrate (has a spine), invertebrate (does not have a spine), male, female, mature, young, pigeon.

   ii. Car, bicycle, motor bike, mountain bike, lorry, unicycle, motor vehicle, road vehicle, means of transportation, sports car, formula one car, human powered vehicle, aeroplane.

b. Suggest strong and weak entity sets for the following application areas. For strong entities draw the generalisation hierarchy graphs annotated with coverage letters where appropriate. For weak entities indicate the parent entity. All in all we would expect at least 12 entities.

   i. People in a hospital (as patients or staff).

   ii. Music-related electronic devices.

c. Consider the following entity sets and suggest possible attributes (at least 10) to describe them. What are the candidate keys? What are the value domains?

   i. Conference participant.

   ii. Customer of a mobile phone company.

d. Consider the following entity sets and suggest possible relationships between them (at least 5). Draw the Entity-Relationship diagrams annotated with multiplicities.

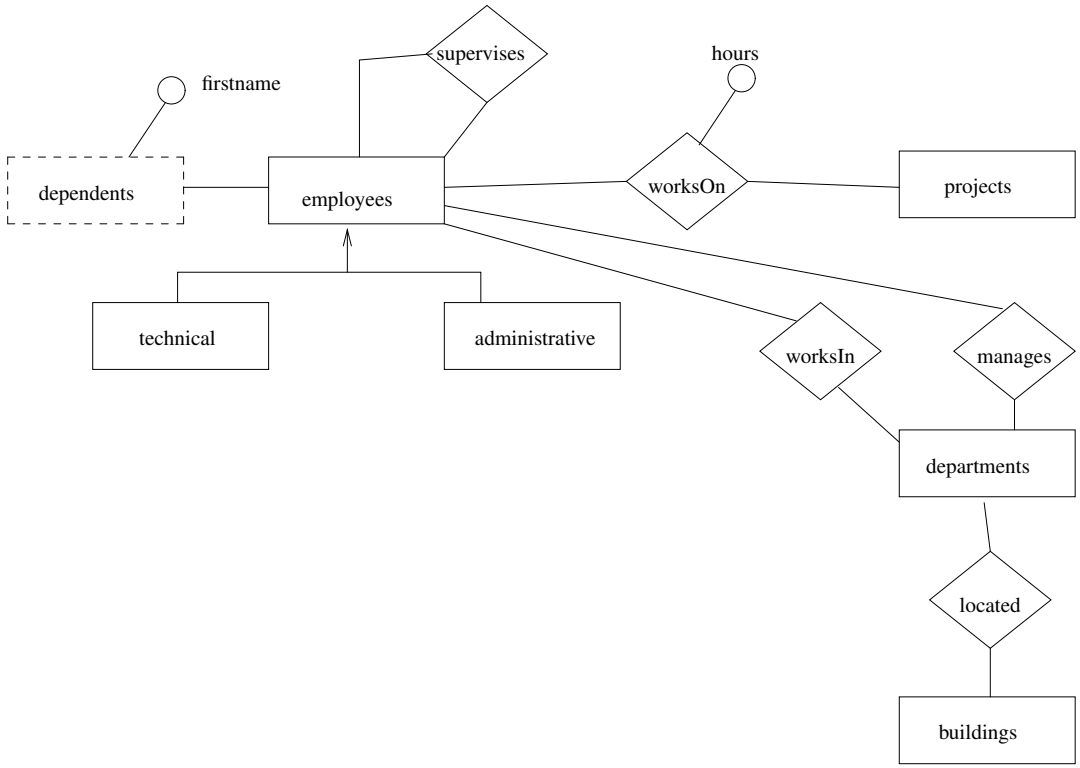
   i. Setting: Insurance company. Customer, policy, type of policy, agent, local office, neighbourhood.

   ii. Setting: Charity. Board of trustees, sponsor, active supporter, fundraising event, request for support, supported "good cause".

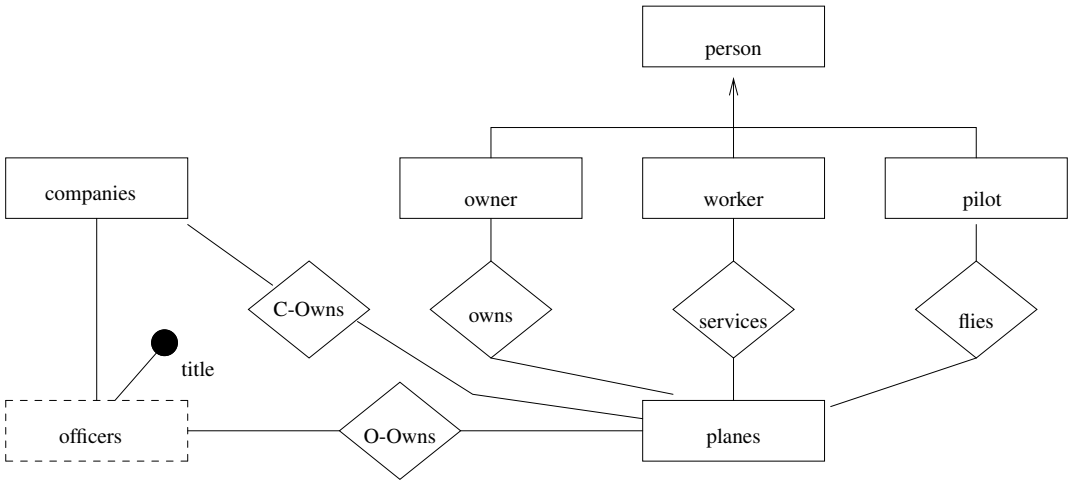
e. Attached is a page with three Entity-Relationship diagrams. Annotate them with multiplicities.

[4] This really shows our age...

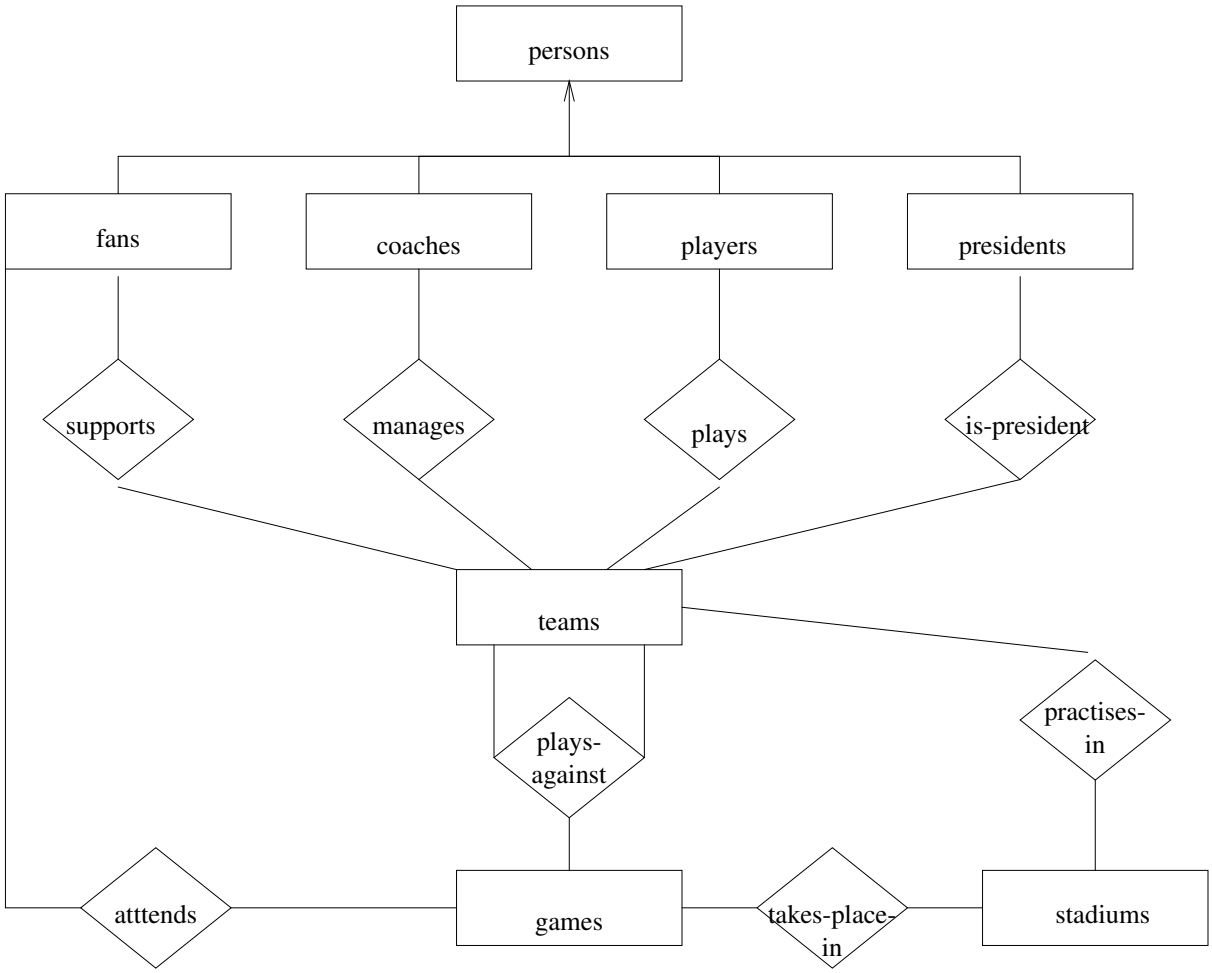
i.


supervises
hours
firstname
dependents
employees
worksOn
projects
technical
administrative
worksIn
manages
departments
located
buildings

ii.


person
companies
owner
worker
pilot
C-Owns
owns
services
flies
title
officers
O-Owns
planes

iii.


persons
fans
coaches
players
presidents
supports
manages
plays
is-president
teams
plays-against
practises-in
atttends
games
takes-place-in
stadiums