Tutorial 1: Entity-Relationship Modelling

Informatics 1 Data & Analysis — Tutorial Notes

Week 3, Semester 2, 2018/19

This worksheet has three parts: tutorial *Questions*, followed by some *Examples* and their *Solutions*.

- Before your tutorial, work through and attempt all of the Questions in the first section. If you get stuck or need help then ask a question on *Piazza* or at *InfBASE*.
- The Examples are there for additional study, practice, and exam preparation.
- Use the Solutions to check your answers, and read about possible alternatives.

You must bring your answers to the main questions along to your tutorial. You will need to be able to show these to your tutor, and may be exchanging them with other students, so it is best to have them printed out on paper.

If you cannot do some questions, write down what it is that you find challenging and use this to ask your tutor in the meeting.

Tutorials will not usually cover the Examples, but if you have any questions about those then write them down and ask your tutor, or post a question on *Piazza*.

It's important both for your learning and other students in the group that you come to tutorials properly prepared. Students who have not attempted the main tutorial questions will be sent away from the tutorial to do them elsewhere and return later.

Some exercise sheets contain material marked with a star $\star$. These are optional extensions.

Data & Analysis tutorials are not formally assessed, but the content is examinable and they are an important part of the course. If you do not do the exercises then you are unlikely to pass the exam.

Attendance at tutorials is obligatory: if you are ill or otherwise unable to attend one week then email your tutor, and if possible attend another tutorial group in the same week.

Please send any corrections and suggestions to Ian.Stark@ed.ac.uk

Introduction

In this tutorial you are required to design an entity-relationship model for a database that will be used by the organisers of a poster exhibition as they keep track of three phases in the exhibition: submission, selection and presentation. You should read the scenario carefully, making a note of all the entities you think might be involved, and of their attributes. Think about the possible relationships between the entities as you do this. The tutorial proceeds with a series of questions about your design, finishing with an ER diagram.

Scenario: A Poster Exhibition

The setting is that you are one organiser of a poster exhibition on "Global Problems of the 21st Century", and you must design a database to keep track of the administration of the exhibition. There are three phases to the administration: submission, selection, and presentation.

Submission Phase: Graphic designers design and submit posters to the exhibition that illustrate one of the chosen global problems. Relevant information on designers includes their name and their affiliation (the organisation they work for) — we assume that these together are enough to uniquely identify each designer. Each poster has a title and is assigned an identification number. Posters may be created by several graphic designers; although each individual designer may only be involved with one poster. Where a group of graphic designers create a poster, we distinguish between the main designer and the co-designers. In case of a single graphic designer, that person is considered to be the main designer of the poster. The main designer is always the point of contact, so should provide an email address.

Selection Phase: All posters created for this exhibition are judged by members of a jury. A judge is a graphic design expert with experience in communication for raising public awareness and for public benefit. Relevant information about each judges includes their name, their affiliation and email. Each poster is judged by three different judges. When judging a poster, a judge gives a decision: accept or reject. A poster is selected for the exhibition only if all three judges give an "accept" decision. The judges cannot compete themselves to appear in the exhibition.

Presentation Phase: All selected posters are presented in the exhibition by their main graphic designers. The poster presentation is allocated a stand and an exhibition session. Each exhibition session takes place at a specific date, and 4 session topics have been announced: human rights, climate change, inequality, and war.

Question 1: Determining possible entity-sets

What are the entity sets you would consider in order to build your ER model? Make a list and then decide your final choice of entity sets. You should include at least *Judge*, *Poster* and *Graphic Designer*.

Things to think about: Is it possible to identify entities directly and unambiguously from the information given, or is there more than one possibility for entity modelling here?

(Tutor Notes) The diagram on page 9 gives a sample solution. The entities in this are: *Graphic Designer*, *Main Designer*, *Co-designer*, *Poster*, *Judge* and *Exhibition Session*. Obviously the names may differ, but they should be recognisable as being these things. There might be other additions: for example, a *Stand* entity or even *Phase*, although that last one doesn't add a great deal to the model.

In the sample model *Main Designer* and *Co-designer* are in an ISA relationship with the *Graphic Designer* superclass. This conveniently accommodates the additional email address field required for *Main Designer*. Other arrangements are possible — for example, leaving out *Co-designer* and declaring those are just any *Graphic Designer* who is not also a *Main Designer*.

Question 2: Defining attributes for a given entity-set

By now you will have decided your sets of entities that can be used to build the ER diagram. For this question, focus on just the two entity sets below:

- *Judge*
- *Poster*

What are the attributes you can assign to each of these given the available information? What are the attribute domains?

Note that in order to draw your diagram you will need to decide the attributes for *all* your ER objects. You do not need to list them all here, but they must appear on your final diagram.

Things to think about: According to the scenario given, a judge gives a decision on a poster: accept or reject. Is this an attribute? If not, what is it? If it is an attribute, does it belong to Judge or to Poster or to neither of them? If it belongs to neither, where does it feature in your ER design?

(Tutor Notes) The *Judge* entity has attributes *name* (string), *affiliation* (string) and *email* (string). The name could be split into *forename* and *surname*, or similar; I'm not sure of any other sensible attributes here.

In the model shown on page 9, the *Poster* entity has attributes *title* (string) and *id* (integer). Another reasonable attribute would be *problem* (string).

The judge's decision on a poster is an attribute of the *Judgement* relationship between the *Judge* and *Poster* entities. Notice that this means that this applies individually to each relationship instance — each judge/poster pairing in the *Judgement* relationship — so for any particular poster there can be different decisions from different judges.

Question 3: Describing relationships

Define relationships between the entities you defined in the first question. What kinds of relationships are these? Are any of them *many-to-one*? *one-to-many*? *many-to-many*? Are there any weak entities in your model? Are there any ISA hierarchies? Are there any key constraints? Do any of these relationships involve total participation?

(Tutor Notes) The diagram on page 9 has relationships *Creates*, *Judgement* and *Presents* — one for each phase of the process.

The fact that posters can have many creators, but each graphic designer can only work on a single poster, makes the *Creates* relationship many-to-one. This is represented in the ER model by a key constraint, shown on the diagram as an arrow-head on the line from *Graphic Designer* to *Creates*. There is also total participation, in that each poster must be created by some graphic designer or designers. This appears on the diagram as the thick line from *Poster* to *Creates*.

The *Judgement* relationship between *Judge* and *Poster* is many-to-many, as each judge may assess many posters, and every poster has multiple judges. There is also a total participation constraint, as every poster must have at least one judge — shown on the diagram as a thick line from *Poster* to *Judgement*

The requirement for exactly three judges per poster is not straightforward to capture in ER diagrams: see the notes later under "Tutorial Discussion" for some ideas on this.

The relationship *Presents* between *Main Designer*, *Poster* and *Exhibition Session* is ternary, with several constraints. Each main designer can present at most once, because they can have only a single poster: this gives a key constraint, shown by an arrow on the line from *Main Designer* to *Presents*. Each poster can be presented at most once, giving another key constraint and an arrow on the line from *Poster* to *Presents*. Neither of these have total participation, though, as not all posters will necessarily be accepted for exhibition. There is, however, a total participation constraint on exhibition sessions, as each session must have some posters. This is shown as a thick line from *ExhibitionSession* to *Presents*.

I don't see any obviously weak entity sets in the scenario — but if anyone has a proposal for one, then I would be interested to know.

There are further constraints which we cannot capture in this ER diagram, so would still need to be recorded in accompanying text. For example, that every poster must have exactly one main designer; that the person presenting a poster must be its main designer; or that a poster should appear in the appropriately themed session.

There are several possibilities for different choices of constraints or whole solutions, but in all cases the relevant assumptions should be clearly stated. For example, we might add an assumption that all judges must judge at least one poster, giving total participation of *Judge* in *Judgement*; or the assumption that every graphic designer must have some poster. One choice might be to not put *MainDesigner* in the *Presents* relationship, on the grounds that it is always possible to work out who should present a poster by identifying its main designer through the *Creates* relationship.

Question 4: Defining primary keys

Choose primary keys for each entity in your model. Which ones did you choose, and why?

(Tutor Notes) In the solution below, *Graphic Designer* has a composite key of *affiliation* and *name*. One alternative would be to introduce a integer *designerId* specifically for this competition; that adds more complication, but avoids any problem if two designers from the same organisation share a name.

The primary key for *Judge* is *email*, but again a composite key of *affiliation* and *name* would do, too.

Based on the scenario, *id* is an obvious primary key for *Poster*. For *ExhibitionSession* we have *title* (might also be called *topic*). If we assume that each session is on a different day, then *date* would do as a key, although I'm not sure it's clear from the scenario text that this is true. Some might reckon that the scenario could include more than one session for each topic, in which case a composite key of *title* and *date* would be needed.

Notice that the subclass entities *Co-designer* and *Main Designer* do not need their own keys, as each instance is already uniquely identified by the key of *Graphic Designer*.

Question 5: Drawing the diagram

Draw the diagram representing your ER model by hand on paper.

(Tutor Notes) See diagram on page 9.

Question 6: Tool support

Draw the ER diagram you created in Question 5 using drawing software of your choice. Print out the result and bring it to your tutorial.

You can use either a general-purpose drawing tool or one with specialised support for ER diagrams. Here three possible examples.

- The online diagram editor *draw.io*. Use the link https://is.gd/da19draw for an ER toolbar specialised to the notation used in this course.
- *LibreOffice Draw* is general-purpose drawing software. (DICE command line: `libreoffice`)
- The *Dia Diagram Editor* is specifically for structured diagrams. Unpack the zip file at https://is.gd/da19dia into a `.dia` subdirectory of your DICE home directory and run `dia` at the command line to get a toolbar for the notation used in this course.

Both LibreOffice and `dia` are already installed on DICE and are also freely available for other platforms. Other diagramming tools include yEd, Calligra Flow, LucidChart, Creately, ...

Having drawn the diagram, what do you think of your chosen tool? Can you identify any advantages or disadvantages for drawing ER diagrams? Is there any way the tool could do more to help? What features would you look for in a tool to support this?

Tutorial Discussion: Thinking around the model

- According to the scenario: 'Posters may be created by several graphic designers; although each individual designer may only be involved with one poster'. Can you model this in your ER diagram? If yes, then how?

(Tutor Notes) Yes, this was done through the key constraint on *Creates*, making that many-to-one.

- Can we model that every poster has a single main designer?

(Tutor Notes) To model this with a key constraint would require having another *Creates* relationship between *Main Designer* and *Poster*. Simply adding this to the existing diagram, though, would leave us with possible inconsistency between the two *Creates* relationships.

One possibility would be to remove the original *Creates* relationship and replace it with two: *MainCreator* and *CoCreator* from each of the subclass entities to *Poster*. With a key constraint on both (no-one can work on more than one poster) and total participation of *Poster* in *MainCreator* (every poster must have a main designer), we capture the constraints given. However, it does then complicate the process of, for example, listing all those who worked on a given poster.

- According to the scenario: 'Each poster is judged by three different judges'. Can you model this in your ER diagram? If yes, how?

(Tutor Notes) This constraint is not straightforward to capture in an ER model as presented in the lecture course. The simplest solution is just to note it as an additional constraint in descriptive text with the diagram. More elaborately, we could split *Judgement* into three relationships: *FirstJudgement*, *SecondJudgement*, and *ThirdJudgement*; with total participation (thick line) and a key constraint (arrow) from *Poster* to each one of the three. However, this would now need an extra note that no-one can judge the same poster more than once.

Some modelling systems extend the basic scheme to put specific numbers on connecting lines, which also solves the problem but at the cost of a more complicated formalism.

- According to the scenario: 'A poster is selected for the exhibition only if all three judges give an "accept" decision'. Can you model this in your ER diagram? If yes, how?

(Tutor Notes) I don't think so, because the constraints given in an ER diagram cannot depend on the values of individual attributes. However, there may be interesting suggestions of ways around this.

This is also because ER diagrams don't express how things change over time, and thinking about this might also suggest weakening some of the constraints given earlier. For example, the participation constraint of *Poster* in *Judgement* might be thought too strong — when first submitted to the exhibition, a poster will have no judges yet, but might still have to go in the database. So although an ER model might show that all posters should eventually be assigned judges, it might not be appropriate to enforce that as a fixed constraint in the underlying database.

- According to the scenario: '...judges are not allowed to compete in "Global Problems of the 21st Century" themselves'. Can you model this in your ER diagram? If yes, how?

(Tutor Notes) The entities *Graphic Designer* and *Judge* are separate, but we don't have any way to enforce that a particular name and affiliation doesn't appear in both.

It might be possible to do this with a more elaborate subclass hierarchy of graphic designers: either with *Judge* as a third subclass, or having a tree with *Co-designer* and *Main Designer* being kinds of *Poster Designer*, and *Poster Designer* and *Judge* being kinds of *Graphic Designer*. All of these would require adjustment of the *Creates* relationship, and identifying which subclasses should be disjoint (no-one in more than one subclass) and complete (everyone in some subclass).

- Based on your answers for the above questions, what do you think about the expressivity of the ER model?

(Tutor Notes) Modelling languages like ER diagrams cannot describe everything in a scenario — and, indeed, being able to abstract away some properties is an important part of modelling. This example, though, does show some things we might like to be able to capture but cannot, or cannot do so very easily.

This is a common trade-off with modelling: keeping a system as simple as possible, while still covering enough cases to be useful. ER diagrams are a widely-used tool for conceptual modelling; but also one with many extensions invented for particular circumstances.

- Based on your answers (and your classmates' answers) for this tutorial, is there a natural "best" solution for this ER model?

(Tutor Notes) I don't think there is any single "best" ER model for this scenario; and students in a tutorial may well bring along multiple different solutions.

- If there is more than one correct answer, how can you be sure that your answer is correct? Any suggestions on how to check?

(Tutor Notes) It can help to go over a diagram and explicitly justify each part of it — each entity, relationship, and constraint — by reference to the scenario. Conversely, given a proposed diagram, go through the scenario and say how each feature mentioned is implemented in the ER model. Make a list of things which the scenario requires but have not been captured in the model.

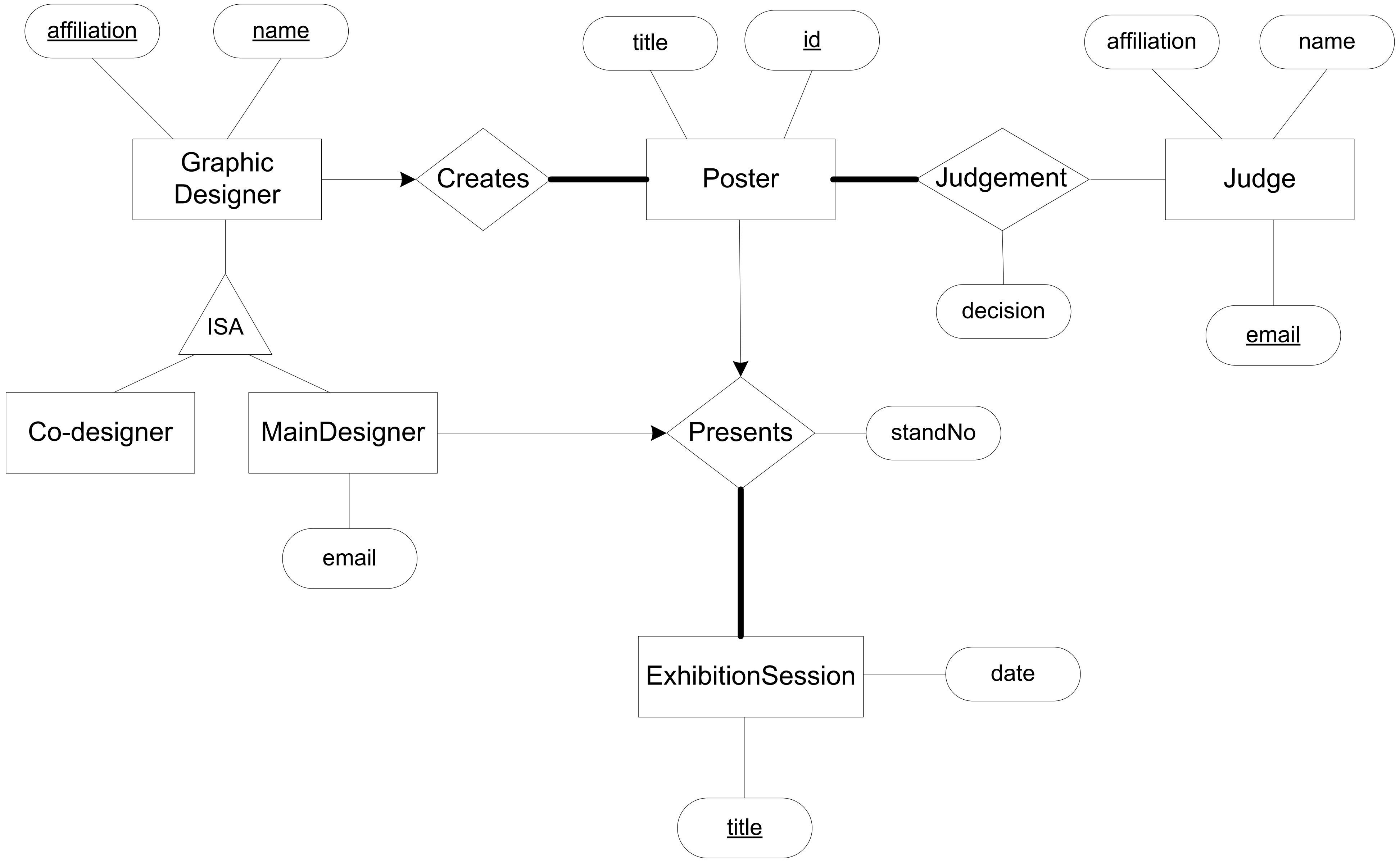
Don't just give a diagram: explain why the diagram is the way it is.

- So far in Informatics 1 you have seen several different modelling formalisms, such as ER models, Propositional Logic and Finite State Machines. What aspects do these modelling systems have in common? How do they differ?

(Tutor Notes) Each kind of model has distinctive situations where it is applicable: ER diagrams for things and the relationships between them; Propositional Logic for assertions about the state of things which may or may not hold, and from which we may find out new facts; Finite State Machines for systems which react over time to events. Sometimes, for a single scenario we may model different aspects using different formalisms.

With any of these modelling formalisms, there can be multiple possible models for a single scenario — no single “right” model, and trade-offs between different ways of presenting things in the modelling language.

All of these modelling systems highlight some facet of a situation — behaviour over time, or the relations between things — and use *abstraction* to focus on what is relevant to that alone. A model is always a simplification of a system, identifying features common with other systems so that we can apply general-purpose tools to specific situations.



An Entity-Relationship model for the poster exhibition database.

Examples

This section contains further exercises on entity-relationship modelling. Examples 1 to 6 run in a sequence very similar to the main tutorial questions, designing an ER model for a given scenario. Examples 7 to 11 are taken from past exam papers.

Following these there is a section presenting solutions and notes on all the examples.

An Inter-University Gliding Competition

For these examples you are required to design an entity-relationship model for a database based on a textual description of a particular scenario. You should read the description carefully, making a note of all the entities you think are involved, and of their attributes. Think about the relationships between the entities as you do this. After this there are a series of questions to help you develop your design and then produce a final ER diagram for it.

The scenario is that you are organising an inter-university gliding competition, and you have decided to design a database to keep track of the administration of the competition.[1]

A number of universities have each entered a team in the competition (known as a gliding *Task Week*), and one of the things you need to keep track of is whether or not they have paid the entry fee. Each university team consists of a variable number of people who will take part in the competition; everybody who competes must be a member of one of the teams.

The pilots will have different levels of experience. Some will be *pre-solo*, which means they can only fly as second pilot (known as *crew capacity* "P2") in a two-seater glider. They can still compete for their team in this capacity, as long as there is an instructor flying with them as pilot-in-charge (crew capacity "P1"). Pilots who are of *cross-country* standard can fly as P2 just like pre-solo pilots, but may also fly on their own (flying "solo") in any kind of aircraft. A pilot flying solo is always P1. Pilots who are *instructors* can fly solo in single- or two-seater gliders, or as P1 in a two-seater with a less experienced pilot. If two instructors are flying together they will simply decide between them who is P1 and who is P2.

There are a number of different types of glider involved in the competition. Some are two-seaters, such as K7, K13, K21 and DG505. The rest are single-seaters; their types include K8, Pirat, DG300, Discus and LS4. There may be more than one glider of a particular type, but every glider can be distinguished by its *callsign* — a short string which is used to identify it in radio communications. Typical callsigns include "MF", "P19", "FNS" and "CPG".

The competition is organised around *tasks*, which are routes that each competing glider must attempt to fly around. On each competition day a task is set for the pilots to fly in their gliders. The task is defined by choosing a set of *turning points* taken from a list available from the BGA (British Gliding Association). There are almost a thousand such turning points defined for the UK, and each has a unique *trigraph* or three-letter acronym to identify it. For example, "STI" is for Stirling and "LOM" is the Lake of Menteith. The competition is to be held at Portmoak Airfield (about 30 miles north of Edinburgh), which is "POR". The task-setter will decide on a suitable task for each competition day, which will involve trying to glide from the starting point at the airfield around one, two or more turning points. For example, a set like "POR, STI, MVN" would define a triangle of just over 100km, with the corners at Portmoak, Stirling and Methven (which is near Perth). For the purposes of this example we will assume that competitors are allowed to fly around the turning points in any order they choose. As well as specifying the trigraph, the BGA list of turning points gives the latitude and longitude of each turning point, so their positions can be precisely identified on a map.

[1]Gliders are aeroplanes without engines, designed for maximal efficiency in extracting the energy freely available in the air, and (on a good day) using it to fly hundreds of kilometres.

Sometimes competitors can gain an unfair advantage by starting off much higher than other gliders, or by happening to pick a better time of day. The task-setter can therefore attach conditions to each task, specifying the maximum starting height allowed and the earliest time at which a glider can start.

Example 1: Determining possible entity-sets

What are the entity sets you would consider in order to build your ER model? Is it possible to map these entity sets directly and unambiguously from the information given, or is there more than one possibility for modelling these entity sets? Choose your final set of entity sets, making sure to include both 'Person' and 'Glider', and justify your answer in as much detail as possible.

Example 2: Defining attributes for a given entity-set

At this point you will have decided your sets of entities that can be used to design the ER diagram. For this question, focus on just these two entity sets:

- Person
- Glider

What are the attributes you can assign to each of these given the available information? What are the domains of these attributes?

The scenario mentioned that a pilot can fly a glider in crew capacity either P1 or P2. Is this an attribute? If not, what is it? If it is an attribute, does it belong to Person or to Glider or to neither of them? If it belongs to neither, where does it feature in the ER design? Explain your reasoning.

Note: In order to draw your diagram you will need to decide the attributes for all of your ER objects. You do not need to list them all here, but they must appear on your final diagram.

Example 3: Describing relationships

Describe the relationships between the entities you defined in the first question and give them names. What kinds of relationships are there? *one-to-one*? *one-to-many*? *many-to-many*? Are there any weak entities in your model? Are there any key constraints? Do any of the relationships involve full participation?

Example 4: Defining primary keys

Choose primary keys for each entity in your model. Explain briefly why you picked those.

Example 5: Drawing the diagram

Draw the diagram representing your ER model.

Example 6: Using dia

Draw the ER diagram you created in Question 5 using the dia application.

Discussion: Thinking around the model

Do not alter your diagram for this question, but describe in outline changes that you might make (such as adding certain attributes to particular entities or relationships).

Imagine you really are running a gliding competition. Is there anything you think is missing from the scenario — any other information that you would like to collect and add to your model? Explain what you would add and why. Could your model be easily altered to include new information or would you need a major redesign?

Example 7

The fictional online service *BitBarrow* provides a repository for shared software development. It uses a database to track various aspects of the service, including the following.

- Users, who are identified by their personal email address and each have a registered name and nickname.
- Projects, identified by a unique project title.
- Which users work on which projects. Each user can work on several different projects, and each project may have multiple contributors.
- For each project, exactly one user who is the project leader.
- Different kinds of project. Projects may optionally be declared as mobile, desktop, or server. Mobile projects need an identified platform, and desktop projects a named operating system.

Draw an entity-relationship diagram that represents this information.

Example 8

A student accommodation service looks after several residential halls, and wishes to integrate management of these into a single database. One small part of this concerns recording the allocation of students to rooms. You have the following information, arising from an initial requirements analysis.

- Each student has a universal username (UUN) that can be used to identify them in the database. This is a unique 8-character alphanumeric code issued by the University.
- Other than that, the database must record every student's name and their year of study.
- Students are allocated to rooms. Each room has a fixed capacity — one, two, three, or even more.
- No student can be allocated to more than one room; but some students in the system might not yet have any room assigned.
- There are several halls, each with its own name and unique Building ID.
- Each room belongs to one hall, and is identified within that hall by its number. There is no coordination in room numbering between different halls: in particular, two rooms in different halls might have the same number.

Draw an entity-relationship (ER) diagram that represents this information. Make sure that you capture the constraints described on the relationships involved, and designate appropriate primary keys for all entities.

Example 9

The craft trading website *Itsy! Bitsy!* is setting up a database to record sellers and their products. This requires recording the following information:

- For each seller, their name, contact email, and postal address.

- For each product, its name, price, and number available.
- Which product is from which seller.
- A unique id number for each product.

Draw an entity-relationship (ER) diagram that represents this information. Make sure to capture the constraints on the relationships involved, and designate appropriate primary keys for the entities.

Example 10

The organisers of the *EXAM 2011* international multi-conference need to keep track of a large collection of workshops associated with the event. Initial requirements analysis brings out the following information about what needs to be recorded.

- Each workshop has a name, and happens on a particular date — or dates, as some workshops last more than one day.
- There are several participants, each of which may sign up to one or more workshops.
- For each participant, it is important to record their name, email address, and the workshops which they wish to attend.
- There are a number of meeting rooms at the conference venue, each of a fixed capacity. Meetings rooms are identified by a floor and room number.
- Every workshop needs an allocated meeting room; where a workshop lasts for two days, it will use the same room on both days.

(a) Draw an entity-relationship diagram suitable for representing this information, in particular the connections between participants, workshops, rooms, and dates.

(b) For each of the following concepts give a brief description of what it means, and give an example from your ER diagram for the previous part.

  (i) Key
  (ii) Composite key
  (iii) Total participation
  (iv) Key constraint

  How is total participation shown in an ER diagram? How is a key constraint shown?

(c) Further analysis reveals additional requirements. However, not all of these can be captured easily in an ER diagram.

  - Each workshop must have an identified organiser among the conference participants.
  - No participant may register for two workshops on the same day.
  - Every participant must register for at least one workshop.

  (i) Identify two of these which can be captured in an ER diagram.
  (ii) For those two, show the additions required to your diagram.

Example 11

A database is to be set up to record details of experts on different subjects. The database will be used to contact experts to be called as witnesses in court cases. The following information needs to be recorded.

- For each expert, their witness identity number, name, affiliation (e.g., "Edinburgh University"), and email address.
- For each field of expertise, the name of the field (e.g., "DNA forensics").
- For each expert, all the fields of expertise that they are expert in.
- For each field of expertise, at most one identified leading expert.

(a) Draw an ER diagram that expresses the requirements for the database. Make sure that you capture all the constraints on the data mentioned above.

(b) Are there any other natural constraints one might impose on the data that are not captured by the requirements above? For each such constraint, say whether it would be possible to modify your ER diagram to include the constraint, and, if so, explain how this would be done.

Example 12

A university wants to set up a database to record details about its staff, and the departments they belong to. They intend to record the following information.

- For each member of staff, their staff identity number, name, job title, and salary.
- For each department, its name and address.
- For each member of staff, all departments that they belong to. It is required that every member of staff belongs to at least one department.
- For each department, the head of department. It is required that each department has exactly one head of department.

(a) Draw an ER diagram that expresses the requirements for the database. Make sure that you capture all the constraints on the data mentioned above.

(b) Are there any other natural constraints one might impose on the data that are not captured by the requirements above? For each such constraint, say whether it would be possible to modify your ER diagram to include the constraint, and, if so, explain how this would be done.

Solutions to Examples

These are not entirely "model" answers; instead, they indicate a possible solution, together with some comments on what features are relevant and where there might be trade-offs between different alternatives.

Remember that not all of these questions will have a single "right" answer. There can be multiple appropriate ways to design an ER diagram, each with particular advantages or disadvantages. Even where these notes include more than one solution, it still cannot cover every possible correct alternative.

If you have difficulties with a particular example, or have trouble following through the solution, then please raise this as a question in your tutorial or on the *Piazza* page.

Solution 1

Figure 1 on page 18 shows a possible ER diagram. That uses the following entities: Team, Person, Glider, Task and TurningPoint. Using different names for the same things — such as University instead of Team, or Pilot instead of Person — would give an equivalent solution.

Solution 2

In the solution of Fig. 1, the Person entity has attributes `personId` (an integer assigned to each person), `name` (string) and `experienceLevel` (string). Other possibilities might be to split the name into first name and last name, or to add further information like an email address. There needs to be enough information to build a key, which is why this solution includes an integer identifier — name alone is probably not enough. There is no need to add a person's university as an attribute, because that follows from their team membership.

For each Glider, Fig. 1 gives attributes for `callSign` (string), `type` (string), and `numSeats` (integer).

The crew capacity in which someone is flying is an attribute of a particular flight: in this diagram, that is an instance of the "Flies" relationship between a Person, the Glider, and the Task being flown. Capacity is not an attribute of the Person because someone may fly as P1 on one day and P2 on another, possibly even in the same Glider.

Solution 3

The relationship between Team and Person is one-to-many: each team can have any number of pilots, but no pilot can be in more than one team. In fact, the constraints on the relationship are slightly stronger, with a key constraint on Person as well as total participation: every Person must be in exactly one team. This should be shown on the diagram as a bold line (total participation) with an arrow-head (key constraint) between Person and the MemberOf relationship.

The other relationships are many-to-many, and there are no weak entities in the model shown here.

Solution 4

In the solution of Figure 1, {`university`} is the primary key for Team. Alternatively, one could choose to generate a unique `teamId` and use that instead. This is what has been done here for Person, with the {`personId`} key. For Glider and TurningPoint, the scenario makes it clear that {`callsign`} and {`trigraph`} are appropriate choices. For Task the model uses {`compDayNo`}, the day number in the competition, as the scenario describes setting one task for each day of the competition. Alternatives might be to use the calendar date of the task, or give each task a unique name.

Solutions 5 and 6

See the ER diagram in Figure 1.

Discussion

One rather serious omission in the model proposed is that there is no provision whatever for keeping score in the competition and working out who won. This was not, however, in the scenario; so in this situation it might be appropriate to go back to the author of that and ask what is required there.

Adding a score to each flight is not, however, completely straightforward: the obvious approach of adding a `score` attribute to the Flies relationship doesn't work satisfactorily, as there will be two entries in the corresponding relation for each two-seater aircraft. One possibility would be to introduce a new, separate relationship between Glider and Task, and add the `score` attribute to that. This would give one score per glider on the task, which is the desired result.

You may be able to suggest other solutions; or identify other data which could be useful to add. These should still be guided by an appropriate scenario: simply adding data without relevant motivation can cause new problems without solving any existing ones.

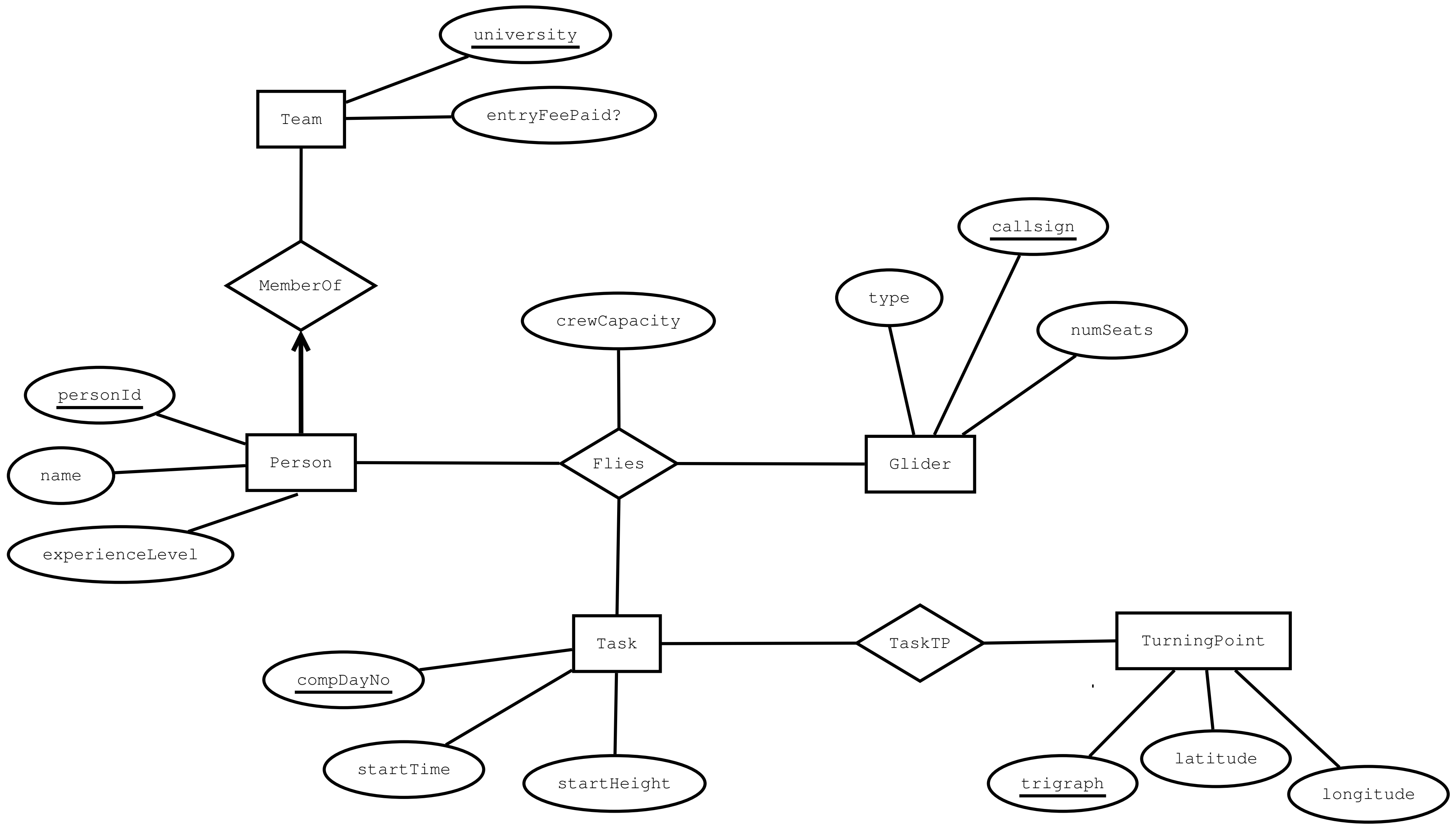
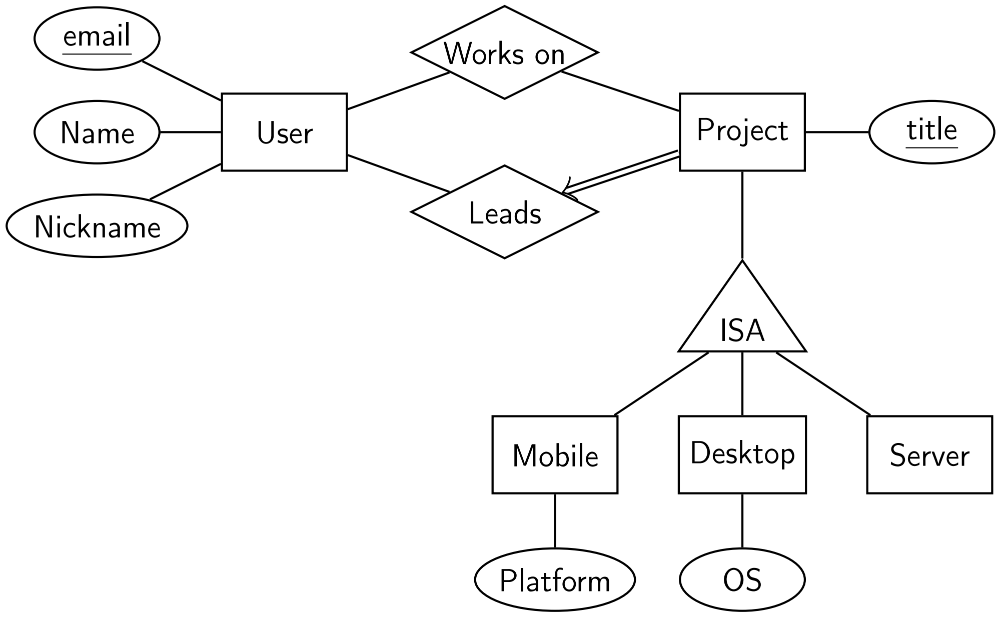


Figure 1: A proposed Entity-Relationship model for the gliding competition database.

Solution 7

The following captures the information described for the *BitBarrow* service.



It's not enough to use an attribute on the Project entity to record the project leader, because that doesn't capture the fact the leader must be someone in the User entity set.

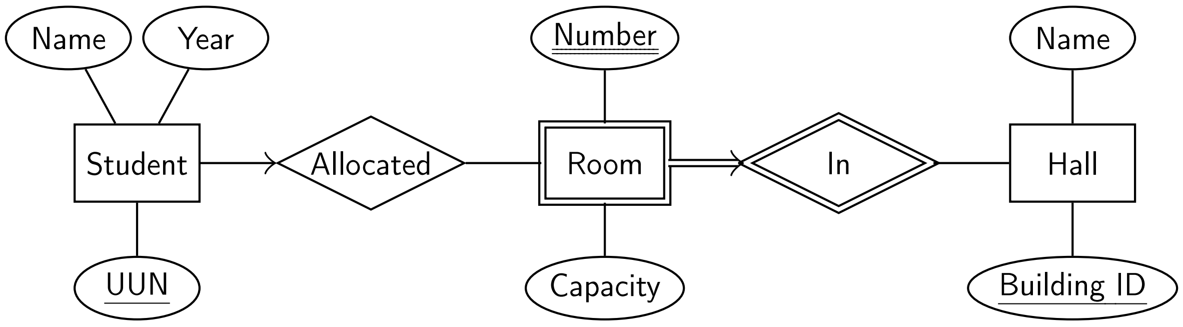
The double line and arrow from Project and Leads indicates that each project must have exactly one user identified as the project leader.

The use of a single line, with no arrow, for both links to the Works on relationship expresses the fact that this is an unconstrained relationship: each user can work on several different projects, and each project may have multiple contributors.

The ISA subclasses classify the different possible kinds of project, with attributes for mobile or desktop projects to record the additional information needed for those.

Solution 8

The ER diagram below represents the scenario described.



All the constraints mentioned in the requirements are captured here.

- The student UUN serves as primary key for the Student entity.
- Student has a key constraint in the Allocated relationship, represented by an arrowhead, indicating that each student can be allocated to at most one room.
- There is no participation constraint on Student (and hence no double line) as the requirements explicitly mention that some students may be recorded in the system but with no specific room assignment.

- Room is a *weak entity*, as the attributes provided, Number and Capacity, may not be enough to uniquely identify a room among all those in different halls. However, In provides an *identifying relationship* for each room, with the relevant Hall serving as *identifying owner*.

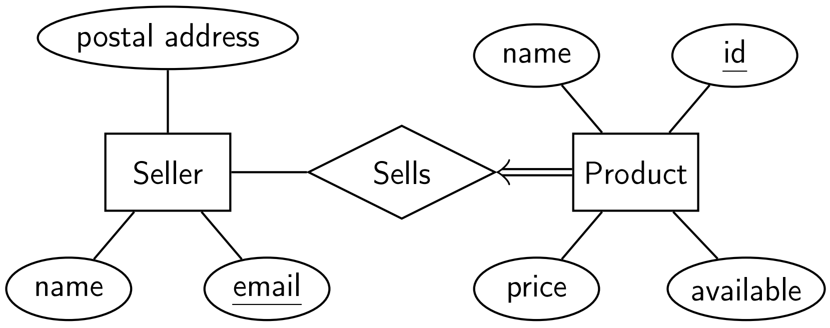
  This is shown in the ER diagram by the double outline on Room and In and the double underline on the room Number attribute — this Number and the related Building ID of the relevant Hall make up a composite key for the Room entity.

  The arrowhead and double line joining entity Room to the relationship In show a key constraint and a total participation constraint, respectively: each room must be in exactly one hall. This is essential for the Hall entity to serve as identifying owner for the weak Room entity.

- The Building entity has two attributes, with the unique Building ID chosen as primary key. Note that both Building and Student have a Name attribute, but there is no conflict in this.

Solution 9

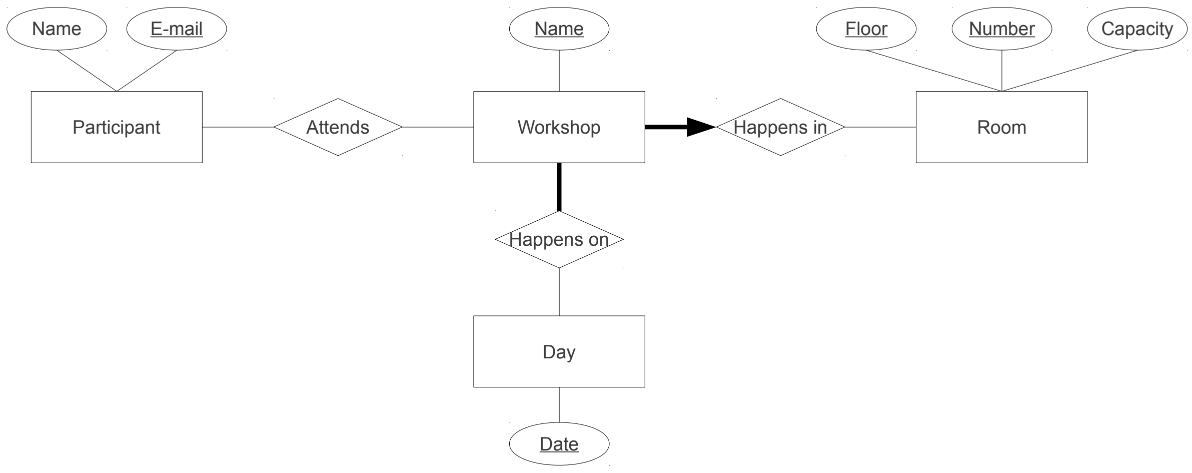
This ER diagram captures the information listed.



The double arrow from *Product* to *Sells* indicates the *key constraint* and *total participation constraint* that every product must have exactly one seller.

Solution 10

(a) Here is a suitable entity-relationship diagram.



One incorrect alternative is to assign a single "date" attribute to workshops; notice the explicit mention of multiple-day workshops in the scenario description.

As well as the entities and relationships, it's important to include the primary keys, the participation constraints, and the key constraint.

(b) **(i)** A *key* is a minimal set of attributes whose values uniquely identify an item in an entity set. For example, the e-mail address of a participant is a key for the corresponding entity set.

(ii) A *composite key* is a key that includes more than one attribute. For example, the floor and room number of a meeting room.

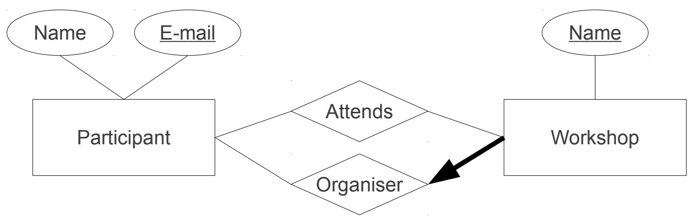
(iii) *Total participation* is a requirement that every element of an entity set must appear at least once in a particular relationship. For example, the requirements that every workshop must happen on at least one date, and in some meeting room.

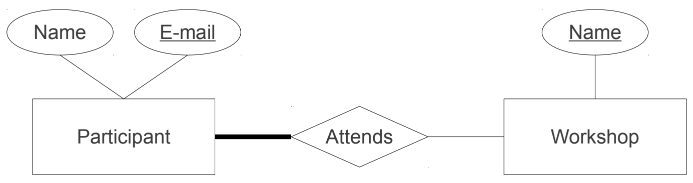
(iv) A *key constraint* is a requirement that each element of an entity set may appear at most once in a particular relationship. For example, the requirement that every workshop must be allocated no more than one room, even if it runs for more than one day.

A total participation constraint is shown by a thick line joining the participating entity to the relationship. A key constraint is shown by an arrowhead on the line joining the entity to the relationship.

(c) **(i)** The requirement for workshop organisers, and for every participant to register for at least one workshop, are both readily presented in an ER diagram.

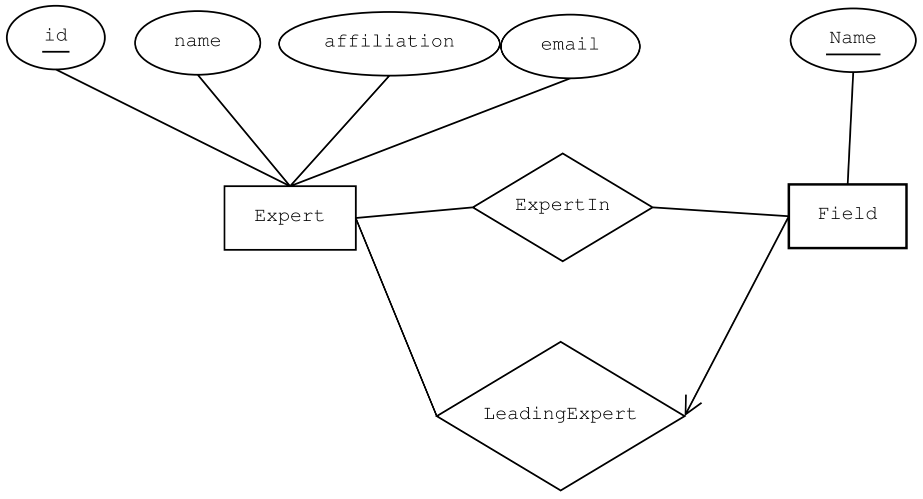
(ii) Recording organisers requires an additional relationship, with both a participation and a key constraint, as shown in the first diagram below. Participant registration for at least one workshop is represented as a participation constraint on the existing *Attends* relationship, as shown in the second diagram below.





Solution 11

(a) Here is a suitable ER diagram:



Notice that the line from entity *Field* to relationship *LeadingExpert* has an arrow indicating a key constraint. This captures the requirement that each field has at most one leading expert recorded.

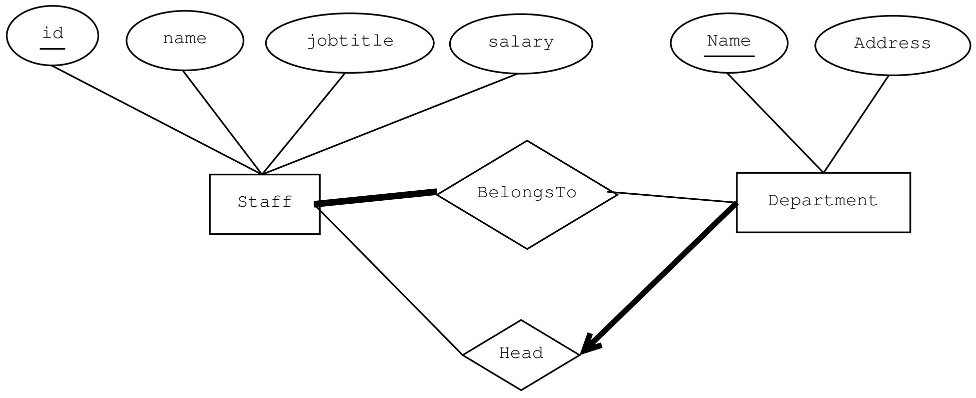
(b) Here are three possible additional constraints:

- Every field should have at least one expert. This could be incorporated by a participation constraint on *Field* in the *ExpertIn* relationship, denoted by a thick or double line for the existing arrow.
- Leading experts should always be listed as being experts in the appropriate field. This cannot be incorporated in the ER diagram.
- Every member of the *Expert* entity set should have at least one field of expertise. This can be incorporated by a participation constraint for *Expert* in the *ExpertIn* relationship, shown by a thick or double line.

You may be able to identify others.

Solution 12

(a) Here is one possible ER diagram:



This includes the following constraints, as required by the specification:

- A thick line of total participation from entity `Staff` to relationship `BelongsTo`, recording the requirement that every member of staff must belong to some department (although still possibly more than one).
- A thick line with an arrowhead from `Department` to `Head`, capturing the constraint that every department has exactly one head of department.

(b) These are some possible further constraints:

- Every Department should have at least one member of staff. This would be incorporated by a participation constraint on `Departments` in the `BelongsTo` relationship.
- Every head of department should belong to the department of which they are head. This cannot be readily incorporated in this ER diagram.
- No member of staff should be head of more than one department. This can be incorporated using a key constraint on `Staff` in the `Head` relationship.

Again, you may be able to identify other reasonable constraints: some of which are represented easily in an ER diagram, some that are not.