Assignment 1

Stage 1 (Design)

Last updated: ...
Most recent changes are shown in red ... older changes are shown in brown.

[Please read this entire document. There are important notes at the end.]

Aims

This assignment aims to give you practice in

- cooperating in a virtual environment
- analysing/refining problem requirements
- designing ER data models based on requirements
- mapping ER data models to SQL schema definitions

The ultimate goal is to build a database schema (using SQL) to represent an events/ticketing site like EventBrite.

Assignment Structure

This assignment is done individually, will run in two stages:

- **Stage 1**: Perform design exercises. Think through the data requirements of the application, developed a complete ER model based on the requirements. At the end of this stage, you will submit an ER diagram as a picture file.

- **Stage 2**: Perform implementation exercise. I will post a "standard" ER design that captures the best aspects of the designs, plus any other components that I think were missed in the discussion in Stage 1. You should then develop a PostgreSQL schema to accurately implement this ER model. In Stage 2, the PostgreSQL schema is the only thing that needs to be formally submitted (details on how to do this will be provided along with the Standard ER Design when Stage 2 is released).

Timeline

Tuesday 11 June	Stage 1 opens - Design spec released
Thursday 20 June (5pm)	Stage 1 closes (stage 1 submission due)
Monday 24 June (10am)	Stage 2 opens - Standard ER model released
Thursday 4 July (5pm)	Stage 2 closes - Deadline for submission of SQL schema

Note that since we practically release the stage 1 'answer' on Monday June 24, we will not accept any submissions for Assignment 1 (Stage 1) after this time. If you missed out on Stage 1 for some reason, you may continue to work on Stage 2 based on the standard ER model.

Submission

Stage 1: Through WebCMS3 assignment page, submit the diagram(s) of your design as a single PDF file (it could contain multiple pages if necessary). You can choose your own drawing tool, the ER notation MUST be the ones we used in the lecture notes. It is important that we use 'COMP9311 standard' notations so that we do not create confusion around marking. All tutors will follow 'COMP9311 standard notations'.

Stage 2: You will submit the SQL schema through WebCMS; details to follow.

Assessment

The assignment one is worth 15%: Stage 1 is worth 50% of Assignment One, Stage 2 is worth 50% of Assignment 1.

The Problem Domain

Organising an event, especially if it involves purchase of tickets, is a task that in the past has been the province of large online ticketing companies like Ticketek, TicketMaster, etc. More recently, event/ticketing startups such as EventBrite, EventBee, TicketBooth and TryBooking, have allowed small organisations and not-for-profit organisations to set up their own events and ticketing online via a simple-to-use Web interface. Our goal in this assignment is to develop a data model that could be used to implement an online event/ticketing site like this.

EventBrite is a typical example of an online ticketing/event site, and it would be worth investigating how it looks to a user and organiser. In order to do this, you'd need to create an account on EventBrite. It is not esssential to do this, since the requirements the system are given below, but playing with the site will give you a better idea of what kinds of facilities such sites provide and what data they deal with. You don't need to provide large amounts of private information and you don't need to actually create an event. You can get a good feel for what's available by doing a minimal registration and then going to the "Create X" pages and seeing what information is collected, but not going through with the final "Save" step.

The aim of this assignment is to develop a database design that can support the *core* functionality of a site like EventBrite. Since a site like EventBrite offers too much to be feasibly modelled in an assignment of this size, we'll restrict ourselves to a subset of its actual functionality. Note that we are dealing with *data representation* only; we are not actually going to implement any of the functionality. However, whatever data structures you design must be rich enough to enable the functionality below to be realised.

Let's call our website `et.org` (**e**vents and **t**ickets). The site has to deal primarily with people, events and tickets. Some people (e.g. ones who simply buy tickets) are not known in much detail. On the other hand, we need more detail about users of the site who organise events.

The front page of the `et.org` site gives a list of upcoming events, in reverse chronologic order of start time. At the top of the page, there is a scrolling list of featured events. There is also a search box so that visitors can find events that are relevant to them. There is a login link, but a casual customer will not need to log in. Each event has an associated "Buy Tickets" link; clicking on this link takes the user to a page where they are given information about the types of tickets available, their prices, and the number of remaining tickets. They can select how many of each tickets to buy, enter their email address and name, and will then be taken to a separate secure site to enter payment details. They will then be emailed a URL which will get them back into the site to check their ticket purchases. A user can also choose to register on the `et.org` site, in which case they can subsequently log in and check their purchases.

When a user logs in to `et.org`, the first thing they see is a page that lists upcoming events that they might be interested in. There are a number of tabs at the top of the page: Events, Tickets, Organiser, Account. The Events page is the default page on login, and also provides a search mechanism. Search is by keyword and searches on event titles, as well as event location and category. Search results are listed in reverse chronological order of the event starting times. When an event is displayed in a list on the Events page, it has a link, which takes the user to a page to purchase tickets. If the user is the organiser of the event, there would also be a link to view the ticket sales.

The Tickets tab gives a user access to a list of all of their purchased tickets, including summary event details and links to print the tickets or forward them to their smart-phone. The Organiser tab allows a user to create events. They first create an "organiser", an online "persona" to manage the events. An organiser could be the name of an organisation or an artist or any other way to useful indicate who is running the event. Organisers create events by providing details such as a title and description for the event, and the location and time of the event. They also set up ticketing by specifying one or more classes of tickets, their prices and their available quantities. Finally, under the Accounts tab, a user can maintain their contact information, password, etc.

Here are more details on the data items that might need to be stored in the back-end of the `et.org` web site:

People

- for every person associated with the site, we need to know at least their email address and their name

- we'll keep their given names and family names separate so that we can sort lists of people by their family name
- everyone must have at least one given name, but some people may have no family name (e.g. "Prince")

Users

- users are people who can log in to the site and organise events
- each *user* needs to be registered with the system and must provide an email address and a password
- their email and password are used for authentication when they log in to the system
- logging in takes them to their "My Events" page, which displays any events that they have purhcased tickets for
- once logged in, they can modify information related to themselves via their "Account" page
- users may provide additional information about themselves, including home and mobile phone numbers, their personal website, their gender and their birthday
- in addition, they may want to provide a home address and a billing address (for `et.org` to send them bills)
- users can also create contact lists, which are essentially just named groups of people (note that the people in a contact list are not necessarily *users*)

Organisers

- each event is created by an *organiser*
- an organiser is really just a "public face" of a user
- users can create as many organisers as they wish, one for each of the organisations that they might create events for
- each organiser has a name and, possibly, some descriptive text about them
- since an "organiser" is most likely going to be viewed as an organisation, they may also want to supply a logo
- to allow organisers to personalise the appearance of their pages, each organiser can specify colours for: main background, text, heading background and heading text, borders, boxes and links
- `et.org` provides a collection of templates giving colour combinations and users can choose one of these or they can develop their own custom colour scheme

Events

- *events* happen at a certain place at a certain time
- people may attend an event if they "buy" tickets (which may cost nothing)
- events may be public or private
  - if public, anyone may buy a ticket by giving their email address
  - if private, only people who are invited may buy tickets
- the actual payment for tickets is handled off the main `et.org` site (e.g. by PayPal), so we do not need to record purchasers credit card, etc. details
- so that people can decide whether or not to attend an event, the event organiser should provide at least the following information:
  - a title and description of the event
  - starting date, starting time, ending date (generally the same as starting date), ending time,
  - location of the event, given either as a text description or by a complete address
  - optional GPS coordinates that can be used to give a Google map showing where the event is held
- organisers may also specify a set of categories for the event (e.g. festival, concert, lecture, party, food/wine, music, etc.) to help users who are searching for events
- events may be one-off or they may be repeated at various intervals (daily, weekly, monthly)
  - for daily events, we need to say how frequently the event occurs (every day, every second day, every third day, etc.)
  - for weekly events, we need to say which day of the week it occurs on and whether it is every week, every second week, etc.
  - for monthly events, we need to say which day of the week and which week of the month (e.g. first monday of each month, second saturday of each month, last tuesday of each month, etc.)
- despite the name, we're not planning to run `et.org` as a charity, and so we collect a small payment for each ticket sold; organisers can choose to have this fee ~~displayed~~ treated

separately to the ticket cost or to bundle it in with the ticket price that a buyer sees
- in order to assist with publicity, `et.org` generates a unique URL for each event and gives a copy of this to the organiser

Tickets

- there may be several classes of ticket associated with an event (e.g. premium, budget, etc.)
- a certain quantity of each class of ticket will be available; the system needs to keep track of sales so that it doesn't oversell
- each class of ticket has its own price; prices will need to include a currency, to allow for world-wide use
- for each ticket class, we may want to limit the number of tickets that one individual can buy at any given time
- in each ticket sale, a person can buy tickets from one or more ticket classes, up to the limit for each class
- each sale has a unique ID associated with it, which is used in the URL supplied to the purchaser

The above should provide sufficient information to get started designing your data model. The important point about the data model you produce is that it is rich enough to support all of the functionality mentioned above. If you want any of the above to be further elaborated, then post a message under the "Assignment 1" topic on the main WebCMS MessageBoard.

How to approach this exercise ...

You can discuss your ideas and questions through WebCMS messageboard. I encourage discussions during this design phase, as a design activity is better done through thinking broadly. You will, through the discussions, form your own opinions and draw your final model for submission.

I'd also think about the requirements carefully, i.e., the details of what the system will provide. This will form the basis of your data model. Then the ER design determines what entities/attributes exist and how they are related. You may go through several revisions of the ER diagram before a final version is produced.

Let the fun begin! ... Helen