COMP9311 22T3 Database Systems

Assignment 1 Stage 2 (Standard ER Design)

Last updated: **Thursday 6th October 9:41am**
Most recent changes are shown in red ... older changes are shown in brown.

Introduction

This document contains the standard ER design for Stage 2 of Assignment 1. You must convert this design into a PostgreSQL relational schema (a collection of `create table` statements) and submit it via the Assignments link on the course web site. In performing the conversion from the ER design to a relational schema, you should follow the approach given in the lecture notes on “ER to Relational Mapping”.

Submission

You will be asked to submit a single file named `ass1.sql', containing the relational schema definitions. We will open a link for submitting the file in WebCMS3 in due course.

Requirements

The schema you submit will be primarily marked by a program (auto-marked). In order for the program to recognise what you've done as being correct, your SQL **must adhere to the following requirements**:

- all entities and attributes must be given the same names as in the design (although case does not matter since SQL is case-insensitive) 所有实体和属性的名称必须与设计中相同
- single-attribute foreign keys should be named after the **table** that they are referring to OR named after the **relationship** that they implement 单属性外键应该以它们引用的表或实现的关系命名
- all tables must have an appropriate primary key defined; all foreign keys must be identified
- wherever possible, not-null, unique and domain constraints must be used to enforce constraints implied by the ER design 在可能的情况下，必须使用非空约束、唯一约束和域约束来强制实施ER设计所隐含的约束
- wherever possible, participation constraints should be implemented using the appropriate SQL constructs 只要可能，应该使用适当的SQL构造来实现参与约束
- all relationhips should be mapped using the approaches described in the lecture notes; in particular, you should avoid “over-generalising” your SQL schema relative to the ER design (e.g. a 1:n relationship should *not* be mapped in such a way that it can actually be used to form an n:m relationship) 例如，1:n关系不应该以这种方式映射，使其实际上可以用于形成n:m关系
- when mapping an n:m relationship `R` between two tables `S` and `T` called the resulting table `SRT` 当映射两个表S和T之间的n:m关系R时，称为结果表SRT
- when mapping the People/Users class hierarchy, use the **ER-style** mapping (i.e. one table for each entity class) 当映射People/Users类层次结构时，使用er风格的映射(即每个实体类一个表)
- when mapping the Events class hierarchy use the **single-table style** (i.e. one table for the whole hierarchy) 当映射事件类层次结构时，使用单表样式(即整个层次结构为一个表)
- when mapping composite attributes, name the new attributes by concatenating the basename and the component name (e.g. addressStreet, addressSuburb, addressCity)
- when mapping multi-valued attributes, name the new table by concatenating the entity and attribute names (e.g. eventsAlarms) 当映射多值属性时，通过连接实体和属性名称来命名新表(例如eventsAlarms)

所有的表都必须定义一个合适的主键;必须标识所有的外键

所有的关系都应该使用课堂讲稿中描述的方法来绘制;特别是，您应该避免相对于ER设计“过度泛化”SQL模式

当映射复合属性时，通过连接基本名和组件名来命名新属性(例如addressStreet, addressSuburb, addressCity)

Place the schema in a file called **ass1.sql** To give you a head-start, a template for the schema is available.

The reason for insisting on strict conformance to the above is that your submission will be auto-marked as follows: 我们坚持严格遵守上述规定的原因是，您的提交将被自动标记如下:

- we will create an initially empty database (no tables, etc.) 我们将创建一个初始的空数据库(没有表等)，
- we will load your schema into this database 我们将加载你的模式到这个数据库中
- we will use PLpgSQL functions to extract the schema in a fixed format 我们将使用PLpgSQL函数来提取固定格式的模式
- we will try to reconcile this with the expected schema 我们将尝试使其与预期的模式保持一致

Following the instructions above is considered to be a requirement of this assignment. If you stray from the expected schema, your submission will be marked as incorrect. Our auto-checking scripts have a little flexibility, but not much, so don't rely on it.

遵循上面的说明被认为是这个作业的要求。如果您偏离了预期的模式，您的提交将被标记为不正确。我们的自动检查脚本有一点灵活性，但不是很多，所以不要依赖它。

Please don't try to second-guess or “improve” the standard design below. Even if you think it's complete rubbish, just translate it as given. If you want to give opinions on the standard schema use the Ed Forum “Assignment 1”. That is, we can discuss the improvement, but we will still use the given schema as the basis for the Stage 2 exercise.请不要试图猜测或“改进”下面的标准设计。即使你认为它完全是垃圾，只要把它翻译成给定的。如果你想对标准模式发表意见，请使用Ed论坛“作业1”。也就是说，我们可以讨论改进，但是我们仍然将使用给定的模式作为第二阶段练习的基础。

Design

This ER design gives one possible data model for the CSEcal on-line calendar application introduced in the first stage of this assignment. The design here is based on the spec and on my experience with using vairous Calendar systems. This isn't necessarily the design that would be used in practice; it has been modified slightly to make Stage 2 of the assignment more interesting (i.e. to give you experience with a range of modelling constructs and translation mechanisms).

To make the presentation clearer, the design is broken into a number of sections. Note that an entity will have its attributes and class hierarchy defined exactly once. If an entity is used in a later section of the design (e.g. to show relationships), it will simply be shown as an unadorned entity box (and you should assume all of the attributes and sub/super-classes from its original definition).

The development of any significant design requires assumptions. Assumptions specific to particular entities and relationships are presented below. 为了使演示更清晰，设计被分成多个部分。注意，实体的属性和类层次结构只定义一次。如果一个实体在设计的后面部分被使用(例如用于显示关系)，它将简单地显示为一个无修饰的实体框(并且你应该从原始定义中假设所有属性和子/超类)。

Data Types

Several specialised kinds of data exist in the system:

Dates and Times

We assume that the calendar will only be used by people in and around the UNSW Kensington campus, and so they'll all be operating in the same time-zone. If we wanted to allow multiple campuses in different time-zones, we could add time-zone as an attribute of each calendar. Note that `time` is a built-in SQL data type and would be suitable to use here.

我们假设日历只会被新南威尔士大学肯辛顿校区及其周围的人使用，因此他们都将在相同的时区运行。如果我们想允许不同时区的多个校园，可以为每个日历添加一个time-zone属性。注意，time是一个内置的SQL数据类型，适合在这里使用。

Dates are also available as a standard data type `date` in SQL. PostgreSQL also provides a `timestamp` data type which combines a date and time (down to microsecond accuracy). This if often useful, but is not relevant for this particular application.

date在SQL中也可以作为标准数据类型date使用。PostgreSQL还提供了一个时间戳数据类型，结合了日期和时间(精确到微秒)。这通常很有用，但与此特定应用程序无关。

Alarms

An alarm is defined simply by a time-interval before an event when the user will be alerted to the event's occurrence. There are two different ways of alerting a user: via email, or by popping up an alert on their screen if they're logged in to the calendar system (we haven't gotten into things like SMS to your mobile phone yet).警告简单地定义为事件之前的时间间隔，当事件发生时，用户将收到警报。提醒用户有两种不同的方式:通过电子邮件，或者在用户登录日历系统时在屏幕上弹出一个提醒(我们还没有介绍过手机短信之类的东西)。

In determining what kind of alarm to use, where *diff = eventTime – currentTime*:

在确定使用哪种类型的告警时，其中diff = eventTime - currentTime:

- if the person is not a user and *diff ≥ 1 hour*, send them email
- if the person is not a user and *diff < 1 hour*, do nothing
- if the person is a user and *diff > 30 mins*, send them email
- if the person is a logged-in user and *diff ≤ 30 mins*, use a pop-up alert

如果该用户不是用户且diff≥1小时，则发送邮件给他们，如果该用户是用户且diff＜1小时，则什么都不做，如果该用户是用户且diff≤30分钟，则发送邮件给他们，如果该用户是登录用户且diff≤30分钟，则使用弹出警告

Note that `interval` is a built-in PostgreSQL data type for describing intervals of time and would be suitable to use here.请注意，interval是PostgreSQL的内置数据类型，用于描述时间间隔，适合在这里使用。

Access levels

Different users have different access rights to calendars/events. Access rights are always assigned at the Calendar level, and apply to all of the events contained in the calendar. The owner of a calendar specifies the access rights for users and groups, and also sets default access rights. Access rights are specified via an access level; possible access levels are:

不同的用户对日历/事件不同的访问权限。访问权限总是在日历级别上分配，并应用于日历中包含的所有事件。日历的所有者指定用户和组的访问权限，并设置默认访问权限。访问权限通过访问级别指定;可能的访问级别有

- no access at all (don't even know that events exist)
- time-block access (can see events as “busy” only, no details)
- read-only access (can see details of event)
- read-write access (can see and change event details)

每个日历都有一个默认的访问级别，当不存在组或用户访问权限时，该级别将应用。

Each Calendar has a default access level which applies when no Group or User access rights exist. Access rights for calendars can be assigned to groups, in which case all members of the

日历的访问权限可以分配给组，在这种情况下，组的所有成员都继承组访问级别

日历的访问权限也可以分配给个人。用户所看到的访问级别由以下步骤确定:首先检查其个人访问级别(如果有)，然后检查其所属的任何组的访问级别(如果多个组对日历具有访问权限，则选择最高的访问级别)，最后，如果没有指定的用户或组访问权限，则使用日历的默认访问级别。访问权限与非用户没有关联，因为他们不能直接访问日历。

group inherit the group access level. Access rights for a calendars can also be assigned to individuals. The access level as seen by a user is determined by first checking their individual access level (if any), then checking the access levels from any groups they are a member of (if several groups have access rights to the calendar, then choose the highest access level), and finally, if they have no specified user or group access, the calendar's default access level is used. Access rights are not associated with people who are not users, because they do not have direct access to the calendars.

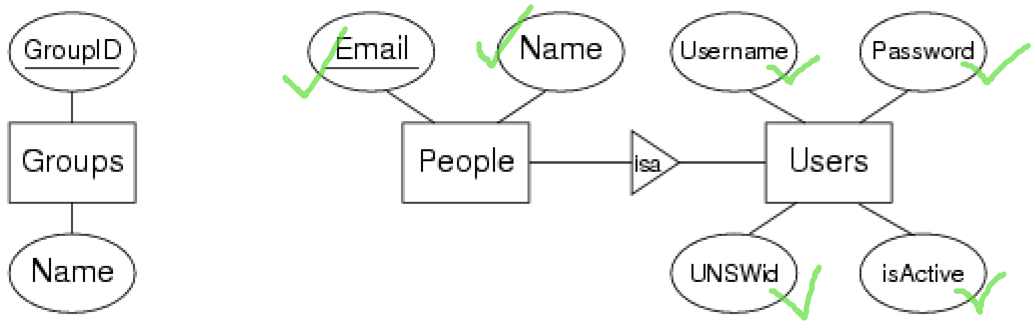
第一次创建日历时，默认的访问级别是 “ no-access ” 。如果日历的所有者将日历的默认访问级别设置为非 “ 禁止访问 ” ，那么它将成为公共日历。

When a Calendar is first created, the default default access level is “no-access”. If the owner makes the default access level to a Calendar anything other than “no-access”, then it becomes a public calendar. In the most lenient case, giving a Calendar default access level of “read-write” means that any user can add events to that calendar, unless they are blocked by having a lower access level applied specifically to them, or to all Groups that they are a member of.

The template defines a suitable `AccessLevel` type using the `create domain` statement. See the PostgreSQL manual for details.

在最宽松的情况下，给日历默认的访问级别 “ 读写 ” 意味着任何用户都可以向该日历添加事件，除非它们被应用于它们的特定的较低访问级别或它们所属的所有组的较低访问级别所阻止

People



Comments: 访问CSEcal的人有两种:新南威尔士大学内部的人和与新南威尔士大学无关的人(注意，CSE计划慷慨地允许整个新南威尔士大学使用其日历系统)。

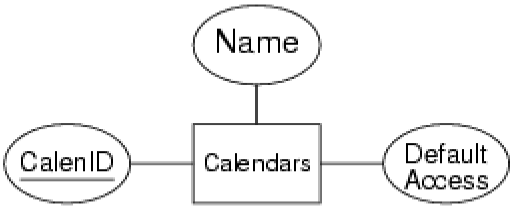
- there are two* different kinds of people who access CSEcal: people within UNSW and people not associated with UNSW (note that CSE plans to be generous and allow the whole of UNSW to use its calendar system)
- people within UNSW can create and manage calendars and events; they are also called “users” and all have a unique UNSW id
- people from outside UNSW may be invited to events inside UNSW, and so might need to be referred to from within the system (e.g. we might want to send them an invitation or reminder)
- people from outside UNSW clearly don't have a UNSW id
- we assume that everyone has an email address (so that e.g. we can send them invitations), and that email addresses are unique, so that email can be a primary key for all people in the system
- since it is more convenient for users to login using a mnemonic username (rather than an email address or UNSW id), we include a username
- for authentication, a real implementation of this system would most likely use an external system such as UniPASS; however, we include a password field so that authentication can be handled entirely within the calendar system
- in order to give the system administrator some control over users, each user has an associated “active” flag which indicates whether they are allowed to log in or not
- groups need a name, which is displayed to users of the system
- however, group names are unlikely to be unique across all users (e.g. many people might have a “My friends” group)
- thus, we introduce a numeric attribute to form a key

为了给系统管理员一些对用户的控制，每个用户都有一个相关联的 “ 活动 ” 标志，表示他们是否被允许登录

* In fact, there is a second type of “user” for CSEcal: the CSEcal administrator. The administrator will have additional privileges, such as adding users, creating groups, and so on. As far as the data model is concerned, however, the administrator is just a “user” and their additional privileges will be implemented via switches in the code for the system.

Calendars

事实上，CSEcal还有第二种 “ 用户 ” :CSEcal管理员。管理员将拥有额外的权限，例如添加用户、创建组等。然而，就数据模型而言，管理员只是一个 “ 用户 ” 及其用户额外的权限将通过系统代码中的开关实现。



然而，日历名不太可能对所有用户都是唯一的(例如，许多人可能有一个“ My appointment ”日历)。

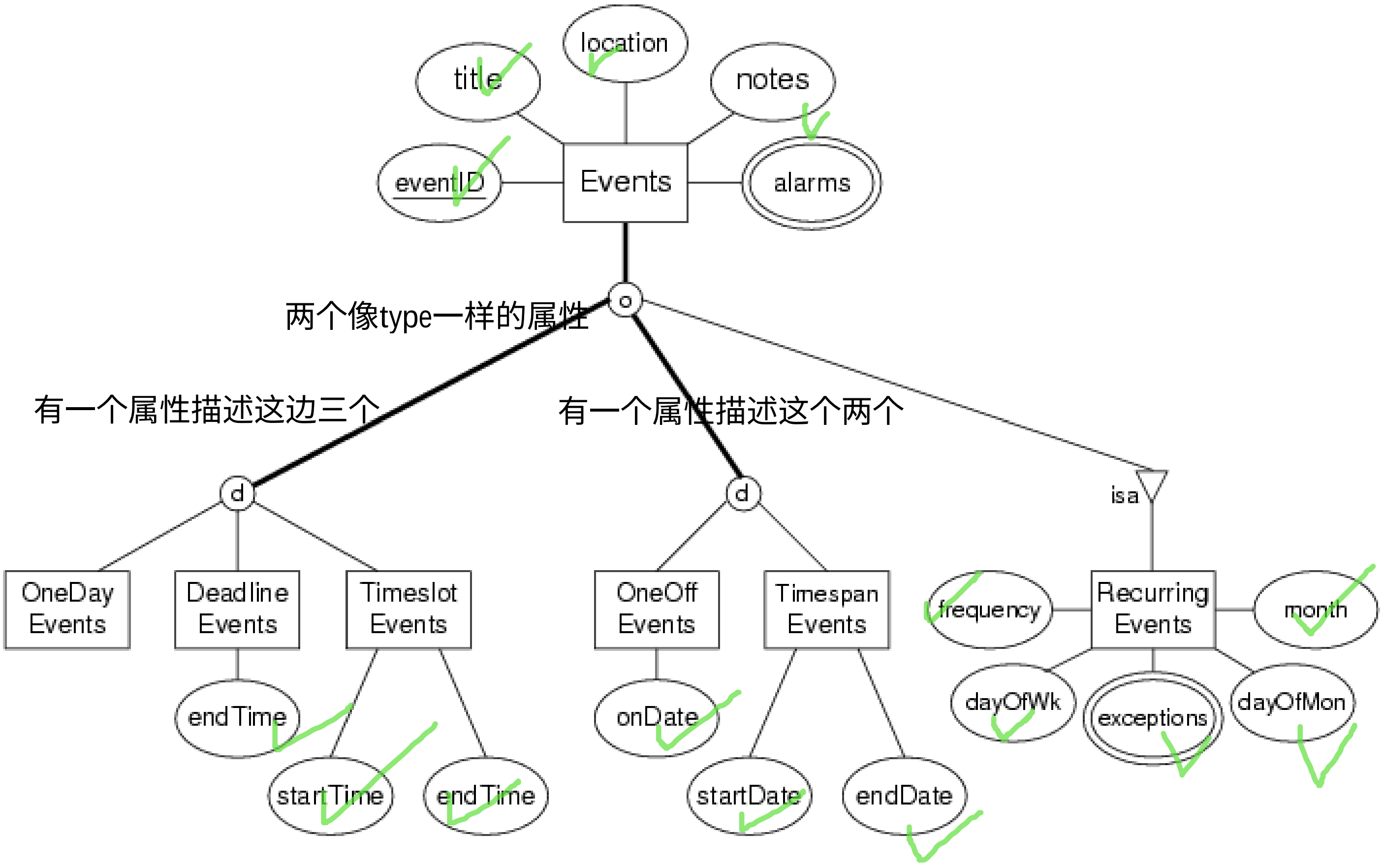
Comments:

日历需要一个名称，它会显示给使用日历系统的人

- calendars need a name, which is displayed to people using the calendar system
- however, calendar names are unlikely to be unique across all users (e.g. many people might have a “My appointments” calendar)
- thus, we introduce a numeric attribute to form a key
- since calendars form the basis for access rights to events, each calendar also needs to specify a default access level (users can make their calendars “public” via this mechanism)

由于日历是事件访问权限的基础，因此每个日历还需要指定默认的访问级别(用户可以通过这种机制使其日历“公开”)

Events



Comments:

- all events need a title (for display on the calendar)
- we also provide the option for any event to have a location and some notes
- since no combination of (title,location,notes) gives a useful primary key, we introduce a numeric attribute for this purpose
- there are many different “styles” of events, e.g. anniversaries (same date each year), meetings (specific time on specific date), period (range of dates e.g. Week 3 22T3), classes (same day each week for a specified period), school meeting (11am on the first Thursday of each month), etc. etc.
- one possible approach might be to think of every possible event style, and have a separate sub-class for each 一种可能的方法是考虑每种可能的事件样式，并为每种样式创建一个单独的子类
- our approach is to factor out the time, date and recurrence aspects of the event specification into three separate sub-class hierarchies 我们的方法是将事件规范的时间、日期和递归方面分解为三个单独的子类层次结构
- note that every event *must* belong to one of the OneDay, Deadline or Timeslot subclasses; every event must also belong to one of the OneOff or TimeSpan subclasses; events may or may not be recurrent
- by combining different pairs from the time/date class hierarchies, and, optionally, a recurrence event, a range of effects can be achieved (e.g. a OneDay event + a Timespan event over a range

注意，每个事件必须属于OneDay、Deadline或Timeslot中的一个子类;每个事件还必须属于OneOff或TimeSpan的一个子类;事件可能会重演，也可能不会重演

通过组合时间/日期类层次结构中的不同对，以及可选的递归事件，可以实现一系列效果

一天的事件+年范围内的时间跨度事件+每年出现频率的重复事件和月份中的特定月份和日期，组合起来实现上述周年类型

of years + a Recurring events with yearly frequency and a specific month and date in month, combine to implement the Anniversary type mentioned above)

- maybe some combinations don't make sense, but these can be handled by the interface preventing these entity combinations from being inserted into the database (ideally, of course, the database itself should prevent nonsensical data from being entered)

也许有些组合没有意义，但是这些可以通过接口处理，防止这些实体组合被插入到数据库中(当然，理想情况下，数据库本身应该防止输入无意义的数据)

- for recurring events, we have effectively collapsed a further subclass hierarchy into a single class; we could potentially have used sub-classes for regular daily/weekly/monthly/annual events on particular dates in particular months or particular days of the week or things like “first Tuesday of each month”

对于重复出现的事件，我们有效地将进一步的子类层次结构折叠为一个类;我们可以用子类来表示每天/每周/每月/每年在特定月份的特定日期或每周的特定日期或诸如"每个月的第一个星期二"之类的常规事件

- the value of the `dayOfWk` attribute is one of Mon, Tue, Wed, etc.
- the value of the `dayOfMon` attribute is one of 1..31
- the value of the `month` attribute is one of Jan, Feb, Mar, May, ...
- the `onDate`, `startDate` and `endDate` attributes are specific dates (e.g. 21 July 2022)
- the `startTime` `endTime` attributes are specific times of day (e.g. 3:15pm)
- there is also an assumption that recurrent events don't occur more than once per day; thus, exceptions to recurrent events can be specified as dates (the dates on which the event does *not* occur)

还有一个假设是复发性事件每天不会发生超过一次;因此，周期性事件的异常可以指定为日期(事件不发生的日期)。

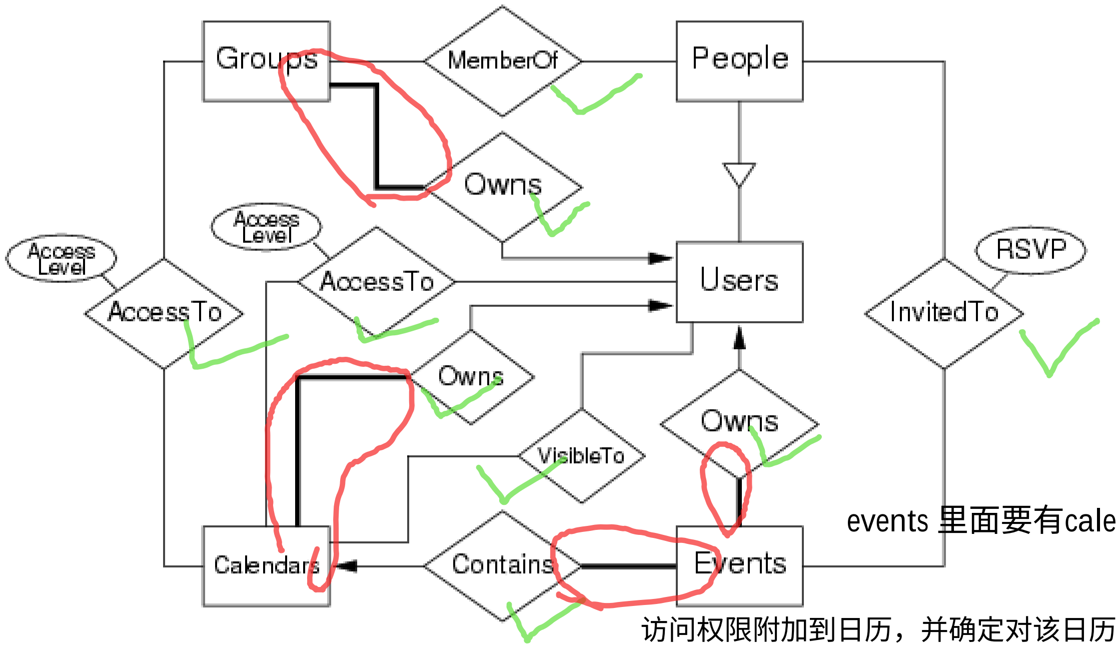
- since there is some overlap amongst the attributes of the subclasses, it seems simplest to use a single-table implementation of the hierarchy, despite the fact that a number of attributes will be “wasted” for non-recurring events

由于子类的属性之间有一些重叠，使用层次结构的单表实现似乎是最简单的，尽管许多属性对于非重复的事件来说是“浪费”的

Important Note: the last point is a requirement: you *must* incorporate all of the sub-class hierarchies into a single `Events` table

重要提示:最后一点是一个要求:您必须将所有子类层次合并到单个事件表中

Relationships



events 里面要有calendars的主键

Comments:

每个组、日历或事件都是由用户创建的，因此属于该用户

- every group, calendar or event is created by a user and thus owned by that user
- access rights are attached to calendars and determine the access to all events on that calendar (see above for a discussion of access rights values)

访问权限附加到日历，并确定对该日历上所有事件的访问权限

- since a user can create as many calendars as they want, they can enforce appropriate access rights to any event they create simply buy putting the event in a calendar with the required access level (if none of their existing calendars have a suitable access level, they can always create a new calendar)

由于用户可以创建任意数量的日历，因此只要将事件放在具有所需访问级别的日历中，就可以对所创建的任何事件实施适当的访问权限(如果现有日历没有合适的访问级别，则总是可以创建一个新日历)。

- invitations are associated with People rather than Users to allow external people to be invited to UNSW events; we assume that the system provides a mail-based interface for them to respond to an invitation

邀请是与人而不是用户相关的，以便邀请外部人员参加新南威尔士大学的活动;我们假设系统为他们提供了一个基于邮件的界面来响应邀请

- the RSVP attribute records the response simply as “Yes” (I will be attending) or “No” (I will not be attending), with `NULL` indicating that the invitee has not yet responded

RSVP属性简单地将响应记录为“是”(我将参加)或“否”(我将不参加)，NULL表示被邀请方尚未响

- logged-in users can specify which calendars they want to appear in their browser at any given time; since these visibility preferences persist over calendar sessions, they need to be stored, and

已登录的用户可以指定在任何时候希望在浏览器中显示哪些日历;因为这些可见性偏好在日历会话中持续存在,因此需要存储它们，因此它们在图中通过用户和日历之间的VisibleTo关系表示

so they are represented in the diagram via the VisibleTo relationship between Users and Calendars

- note that users can only specify visibility on calendars that they have access rights to, but this is not specified in the diagram注意，用户只能在他们有访问权限的日历上指定可见性，但在图中没有指定这一点
- if a calendar is VisibleTo some user, then the events in that calendar are displayed in their CSEcal window; 如果某个日历对某些用户可见，那么该日历中的事件将显示在他们的CSEcal窗口中
- if a calendar is not VisibleTo a user, then none of its events are displayed
- the user can toggle visibility via checkboxes in the interface

如果日历对用户来说是不可见的，那么它的任何事件都不会显示。用户可以通过界面中的复选框切换可见性

If any aspect of this specification requires further clarification, ask for it under topic “Assignment 1” on the course forum.

Don't forget to have fun!