**数据库设计说明书**

目录

**一．引言2**

1.1编写目的2

1.2背景2

1.3定义2

1.3定义2

1.4参考资料2

**二．外部设计3**

2.1 标识符和状态 3

2.2 命名约定 3

2.2.1 基本规范3

2.2.1.1可用字符3

2.2.1.2可用字符3

2.3 设计约定 4

**三．结构设计5**

3.1 概念结构设计 5

3.2 逻辑结构设计 11

**四．运用设计15**

4.1 数据字典设计 15

4.2 安全保密设计 15

1. **引言**
   1. **编写目的**

1、本数据库设计说明书旨在详细描述轻量级团队协作办公解决方案中的数据库结构和设计，包括逻辑结构设计、数据字 典设计、安全保密设计等，清晰明确并结构化地将本项目系统的数据库结构和设计方案传递给相关项目参与人员，加深开发者 对项目的了解程度，并且可以有助于尽早发现和减少项目的需求错误，帮助成员更好的融入到项目中。

2、本数据库设计说明书读者：用户、系统设计人员、系统测试人员、系统维护人员。

3、本数据库设计说明书是系统详细设计、用户验收的依据之一

* 1. **背景**

针对广大考研学子的需要，在对于考研没有老师带领，多数都是自主学习，自主寻找资料的前提下，我们为其提供了一个方便便捷的平台。以及方便网站升级和管理。随着网站制作条理清晰化进程，为免制作途中结构化的寸步出错。客观性的要求该报告的分析去进一步规划网站的构建与数据库信息存储与管理。

* 1. **定义**

本文用到的术语符合国家标准《软件工程术语（GB/T11475-1995）》。

* 1. **参考资料**

[数据库设计说明书](https://wenku.baidu.com/view/1965b66a0029bd64793e2c7a.html)

1. **外部设计**

# 2.1 标识符和状态

数据库软件名称：Mysql

数据库的名称：test

# 2.2 命名约定

## 2.2.1 基本规范

### 2.2.1.1可用字符

数据库、数据库表、字段所有名称的可用字符范围A-Z，a-z，0-9和下划线\_。在一般情况下，统一为大写或小写的名称，两个单词间用\_下划线分隔，SQLServer使用 Pascal或Camel方式命名。这些不仅仅是为了数据库设计的可读性，也是为了最终生成代码的可读性。数字方式用于区分标题相同的字段，例如多图片存储用image1，image2，image3进行区分。

### 2.2.1.2 命名方式

命名的方式统一使用英文单词或其缩写，禁止使用汉语拼音，且均使用单数名，如客户customer而不是customers。名称应该清晰，准确表达事物的含义，遵循见名知意的原则。

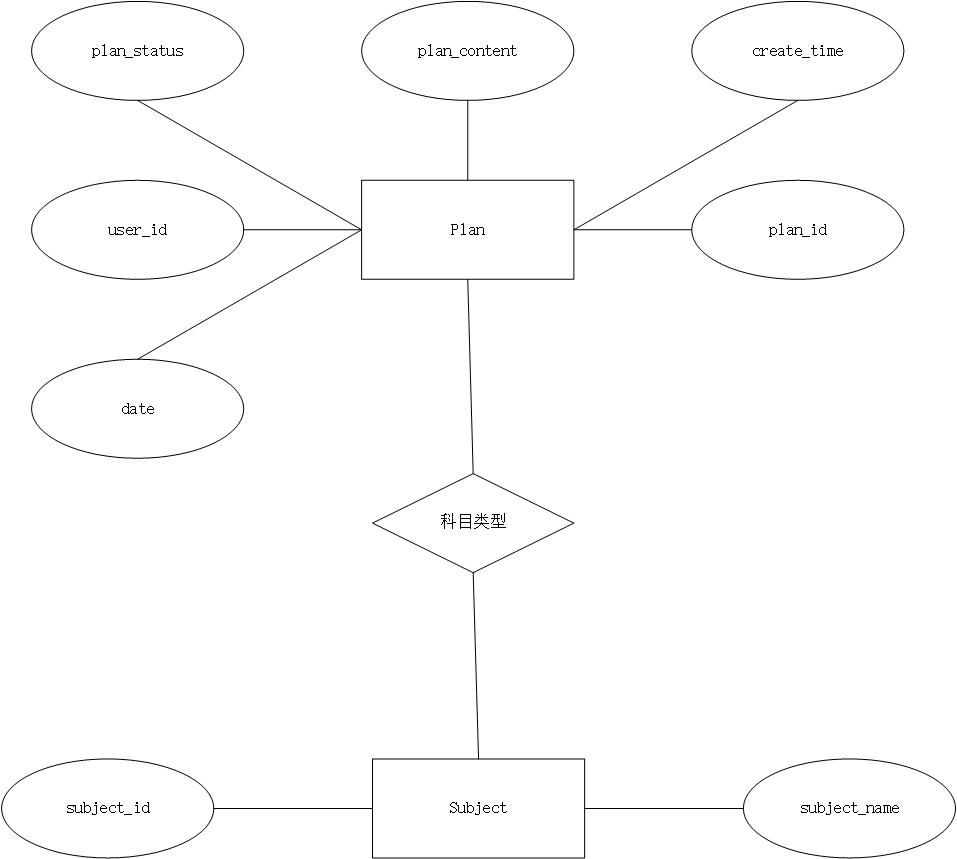
# 2.3 设计约定

在本系统中，数据库设计采用visio进行设计，并且采用面向对象的设计方法，首先及进行对象实体设计，最后将对象持久化到数据库中，所有的表和表之间的关联（ER图）都采用标准的visio设计工具进行，这样能够将整个系统的设计和数据的设计有机的结合起来。

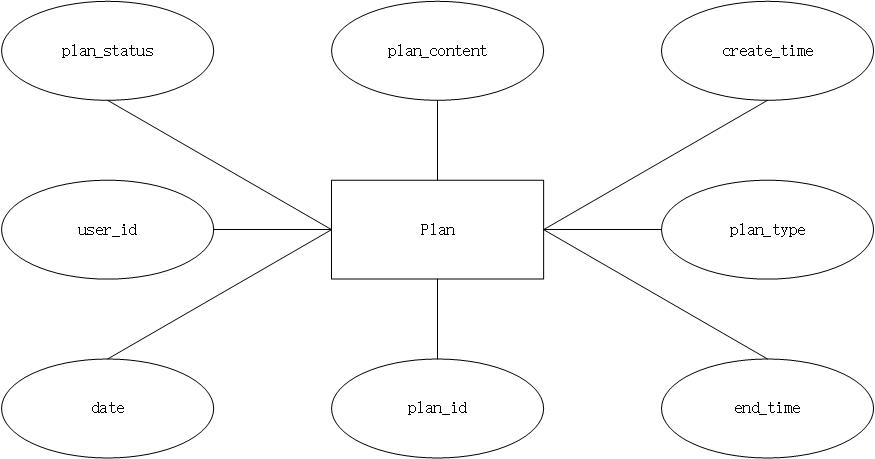
1. **结构设计**

# 3.1概念结构设计

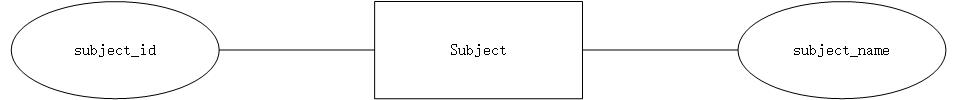
plan局部ER



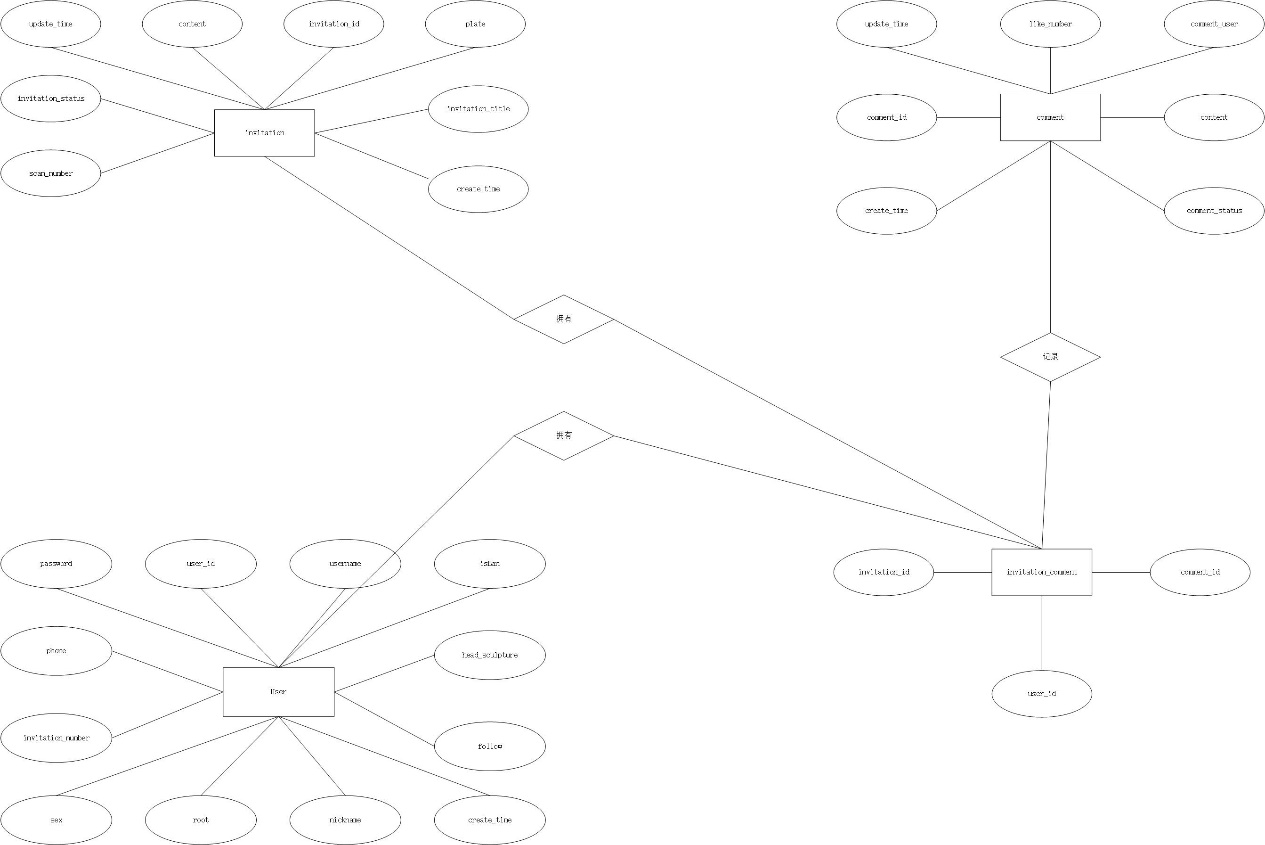
Plan

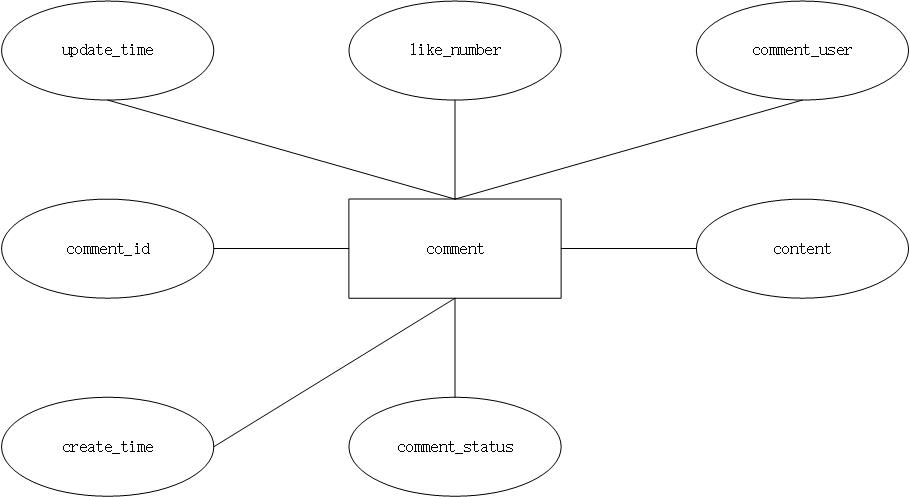


Subject

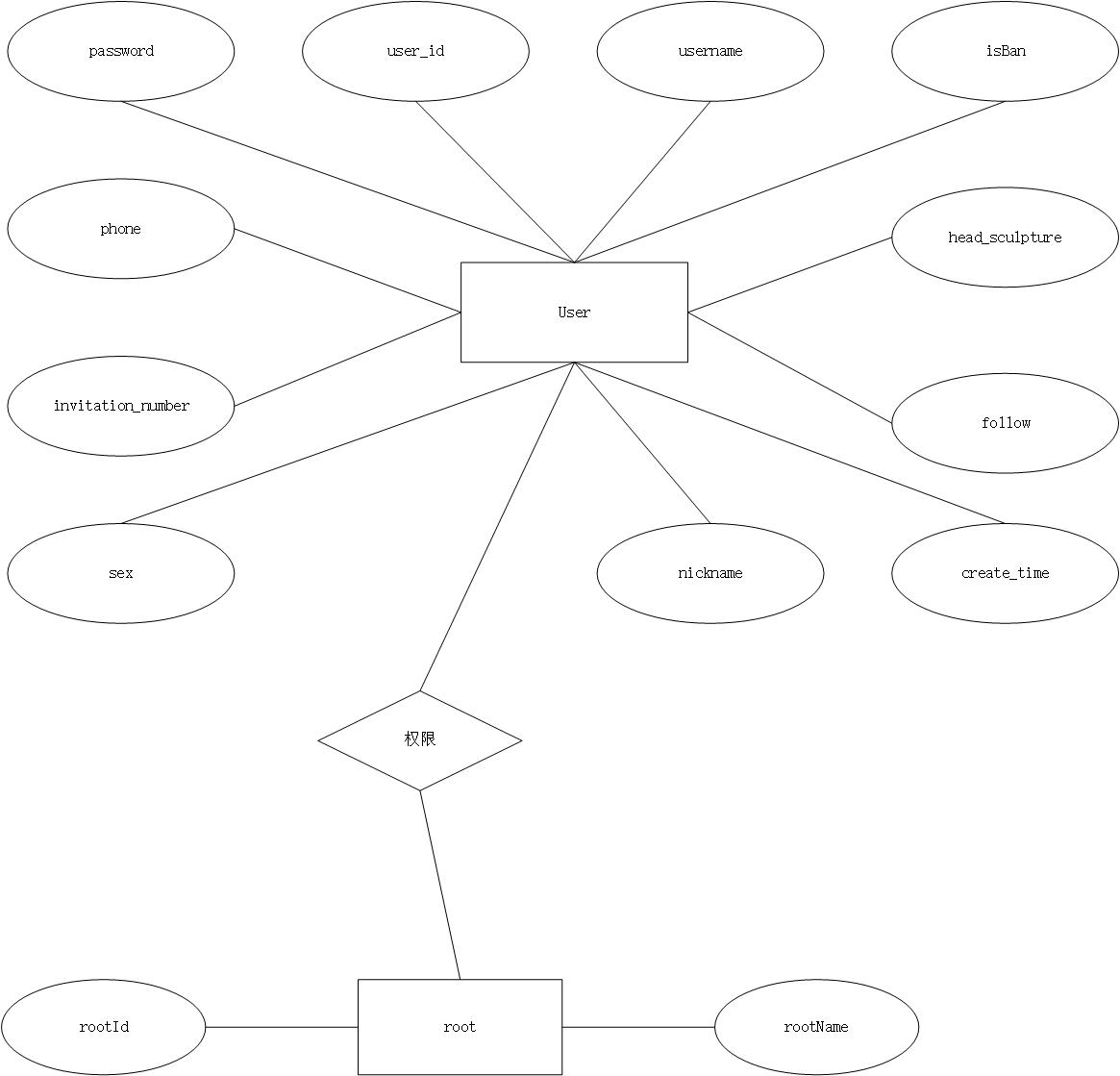


Comment局部ER

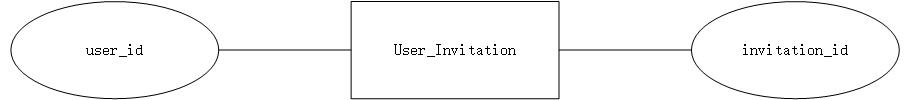




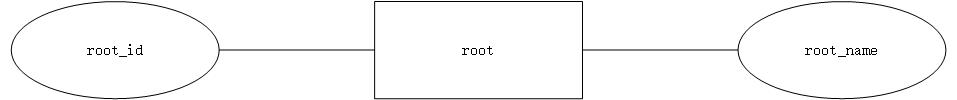
User局部ER



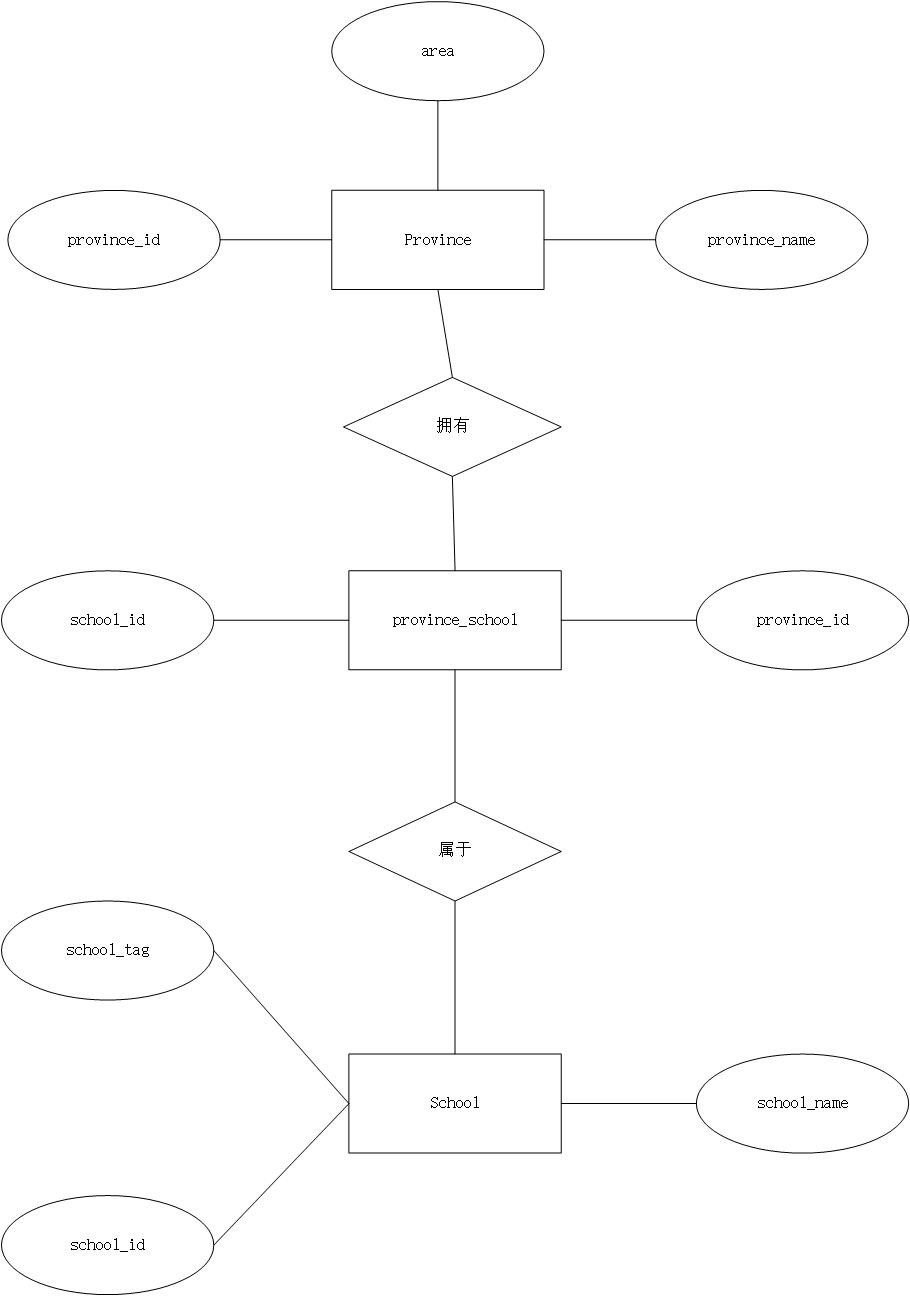
userRoot

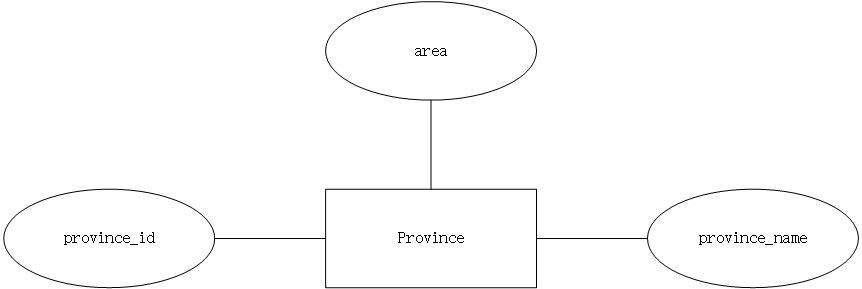
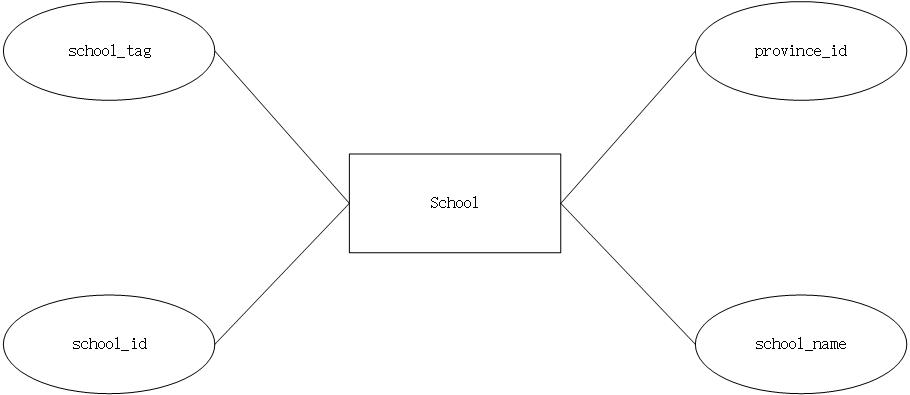
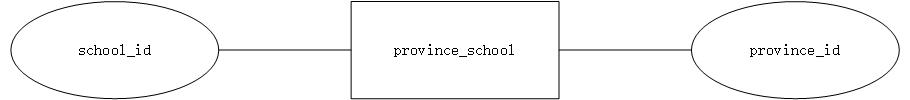


user\_invitation

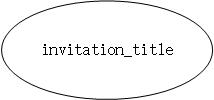


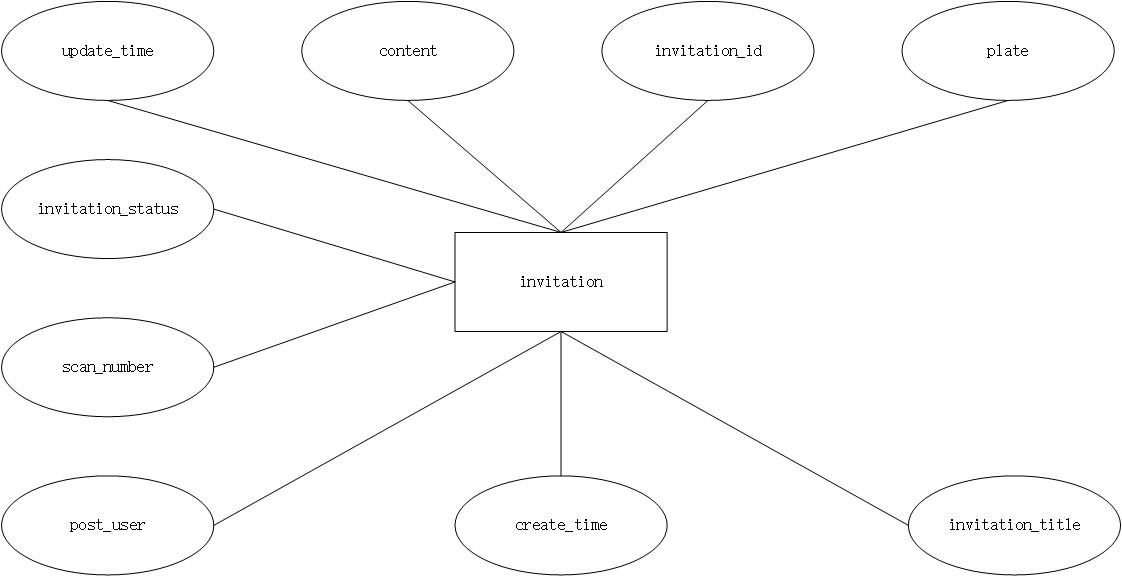
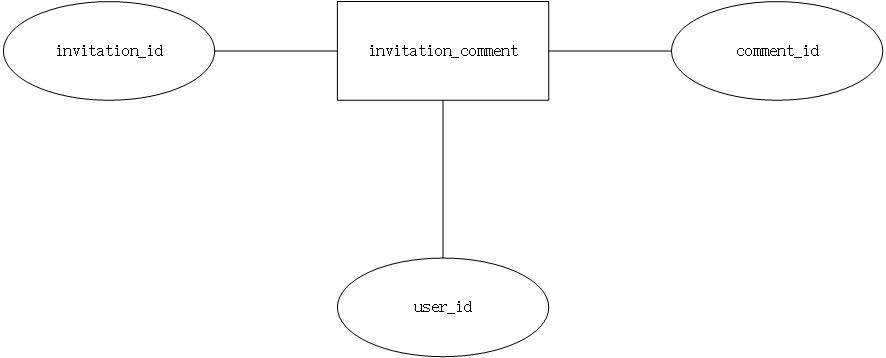
School局部ER





Invitation局部ER





# 3.2概念结构设计

User用户信息表记录用户登陆账号和联系方式，权限状态等

root用于表现权限0表示游客1表示管理员

sex用于表示用户性别0表示女1表示男

isBan用于表示用户封禁状态0表示未封禁1表示封禁



userRoot权限表

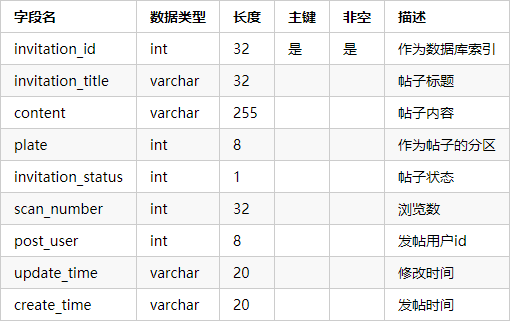
目前权限只有0游客1管理员



Invitation帖子表

status表示帖子状态0表示显示1表示隐藏

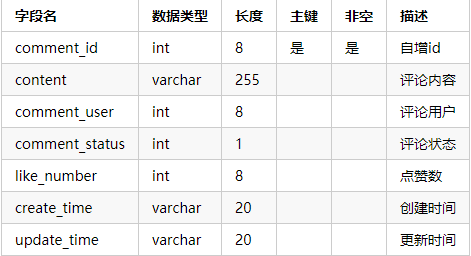
post\_user用于连接user表



comment评论表

comment\_user用于连接user表

comment\_status表示评论状态0为显示1为隐藏



like点赞表



plan计划表

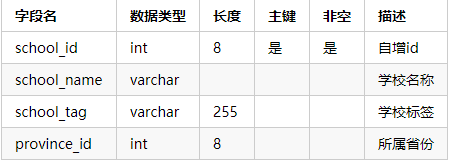
plan\_status表示计划的状态0表示未完成1表示完成



province省份表



school学校表



> subject学科表



user\_invitation

每个用户维护一张属于自己的发帖表



invitation\_comment

每个帖子维护一个属于自己的评论表

每个用户维护一个属于自己的评论表



province\_school

每个省份标记该省所有的学校



1. **运用设计**

# 4.1 数据字典设计

数据字典的设计有两种

一种是，把主体的属性代码化放入独立的表中，不是和主体放在一起，主体中只保留属性的代码。这里属性的数量是不变的，而属性取值的数量可以是变化的。

另外一种是，用一个表来放结构相同的所有属性信息，不同属性的不同取值统一编码，用“类型”来区别不同的属性，主体中保留属性代码的列表。这样主体所拥有的属性数量就是可变的了。

第二种数据库设计较为一般，更通用，由于本系统数据库结构较为简单，所以采用第一种数据字典设计方法。

例如，学科表就是一个数据字典，本身存放学科表的属性信息（学科id 学科名称），在其他表中以学科id的形式使用。如果需要修改学科名，无需修改每一个使用学科的表，而只需要修改学科表中相应学科id的学校名称。

# 4.2 安全保密设计

安全性保护

采用前后台相结合的安全性保护措施共同维护数据库应用系统的安全。

用户身份鉴别：对于后台数据库，一般情况下不允许远程登陆服务器。若是要使用远程登陆则必须采用身份验证方式，在连接数据库给每一个合法用户简历用户名和密码，用户连接数据库时，需要验证，只有通过验证才能进入数据库。后台登录只允许管理用户登录，在数据库表userRoot权限表中对游客和管理员进行区分，游客只能访问前台页面，管理员可以管理后台页面。二者使用均用数据库中的统一账号管理，不能使用第三方登录。

完整性保护

保证数据源的正确性和一致性。

数据库的完整性可分为：实体完整性，域完整性，相关完整性和用户定义完整性。

实体完整性是一行记录为某个特定表的唯一实体，每个实体都有唯一性标识。

本数据库中每一个数据表都有唯一一个字段id用于区分不同的记录。

域完整性是关于一个给定列的有效入口，即对数据类型，格式，范围，精度等规定。

在本数据库的设计过程中均以对各个字段做出较为明确的规定。

相关完整性确保数据的一致性，增强输入数据的有效性。

例如在plan计划表内输入不存在的user\_id将会引发错误。

用户定义完整性是针对某一个具体应用所设计的数据应满足的要求，系统提供定义。

和检验这类完整性的机制，使用统一的方法来处理，而不是由应用系统完成这项工作。

实现完整性校验的方法主要有两种，一种是由数据库定义和规范设计的，如表中非空约束，唯一性约束，外键约束，主键约束，缺省、索引和触发器。定义简单，安全可靠。

另外一种是在实际应用的编程中进行的规范，例如用下拉框和选择框替换输入框，保证可枚举的类型内容正确性。使用系统规范的日期输入框，避免用户自定义输入造成的日期格式无法统一等问题。

并发控制

保证多个用户能共享数据库，并保证数据源的正确性和一致性。

以事务为单位管理用户程序的并发访问。

事务具有ACID

原子性(atomicity)：数据库的逻辑单位，要么都做，要么都不做

一致性（consistency）:一致性指的就是当数据库只从某种一致性状态到另一种一致性状态，数据库只包含成功事务提交的结果。

隔离性（Isolation）:一个事务执行不受其他事务干扰。

持续性（Durability）:一个事务一旦提交，它对数据库中的数据的改变就是永久的。

并发冲突可能带来的不一致性：

1. 丢失修改：两个事务同时读取同一数据并修改,结果后提交事务提交的修改覆盖了前提交事务提交的修改,导致前一个事务的修改丢失；效果等同于串行执行事务。

解决方法：事务要识别提交修改时刻库中的数据是不是与原读取时刻库中的数据一致。

1. 不可重复读

前后读取的数据内容不一致或数目不一致。

1. 脏读

一个事务读取另一个事务未提交的数据。

主要原因就在于破坏了事务的隔离性。

并发控制中主要技术有：封锁locking、时间戳timestamp、乐观控制法optimistic scheduler、多版本并发控制multi-version concurrency control

本系统所采用的前台以读取内容为主，后台以写入内容为主。后台管理员目前数量为1，只有一个管理员能够进行数据的写入，并不存在写入并发性问题。单考虑到系统的扩展性，将来可能不止需要一个管理员，所以应当对数据库进行并发控制。

锁定的控制有两种，悲观锁和乐观锁。

悲观的锁定可以让用户知道他们的数据更新不会被阻塞，当他们更新数据库中的数据是不会遇到冲突

乐观锁定则会带来倘若用户执行数据更新操作时，如果数据在操作执行已经改变了，系统就必须通知终端用户操作不能执行。

本系统目前使用的并发控制技术为乐观锁定。

对并发冲突采取乐观的态度，认为事务并发冲突度不高，对拟将更新的记录在查询时不锁定，在真正更新时再作检查，检查通过后方锁定。

更新前在应用中存储所要操作行的“前映像”，更新时对操作行的当前值与存储的旧记录进行比对，如果数据一致则表明没有并发冲突，就提交更新；否则则需要根据业务逻辑来作进一步处理。

在基表上使用一个能够追踪数据版本的特殊列。添加一个版本控制的字段，或者添加一个时间戳字段，每次需要写入时对比时间戳；并发事务在更新提交的时候检查当前数据库中数据的时间戳和自己更新前取到的时间戳的一致性，如果一致则就提交，否则就回滚放弃。

、