数据库设计

# E-R图

用户

用户ID

邮箱

用户名

签名

所在地

性别

生日

好友

消息

消息ID

时间

路径

内容

发消息

评论

内容

时间

评论ID

评论消息

头像

密码

日期

日期

用户ID

图片

消息ID

目标用户ID

# 关系模型

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 用户/user | | |  |  |  |
| 用户ID | 用户名 | 密码 | 邮箱 | 签名 | 头像 | 性别 | 生日 | 所在地 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 好友/friend | |  |  | 消息/status | | |  |
| 用户ID1 | **用户ID2** |  | **消息ID** | 用户ID | 日期 | 时间 | 内容 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 评论/comment | | |  |  |
| 评论ID | 用户ID | 目标用户ID | 消息ID | 日期 | 时间 | 内容 |

|  |  |
| --- | --- |
| 图片/picture | |
| 路径 | 消息ID |

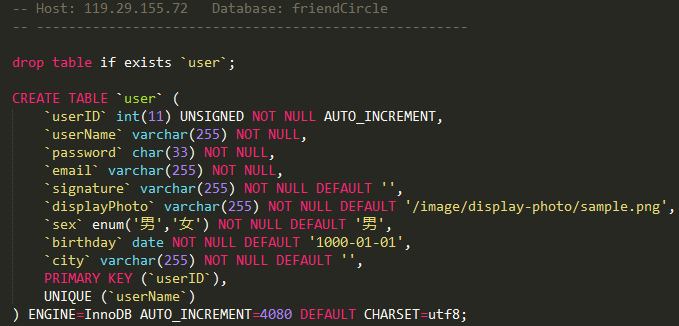
# 数据库设计

共有5张数据表，如下：

|  |  |  |
| --- | --- | --- |
| 数据表名称 | 功能 | 备注 |
| user | 存放所有用户的信息 |  |
| friend | 存放所有的朋友关系 | 每条记录为两个用户ID对 |
| status | 存放所有用户所发的状态/消息 | 可以附加图片状态 |
| comment | 存放所有的评论信息 | 不仅可以评论状态，还可以回复评论 |
| picture | 存放用户状态中的图片信息 | 以路径的形式存放 |

1. user表

创建user表的sql语句如下：

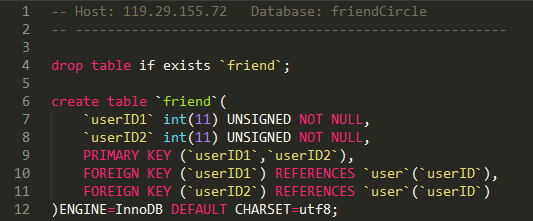


user表保存用户的所有信息，其中

* 1. 以userID作为主键；
  2. userName设置为“unique”，即用户名独一无二，不能重复（用户名是用户注册时设定的，是用户登陆的凭证）；
  3. 保存了用户头像图片的路径；
  4. 所有的内容都不能为空，一些非重要内容设置了默认值；

1. friend表

创建friend表的sql语句如下：

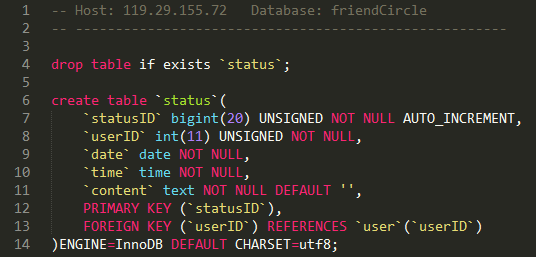


friend表保存所有的朋友“对”，其中：

* 1. 以两个用户的userID的组合作为主键；
  2. 两个用户ID分别设置为外键，这样就与user表关联起来了。

1. status表

创建status表的sql语句如下：

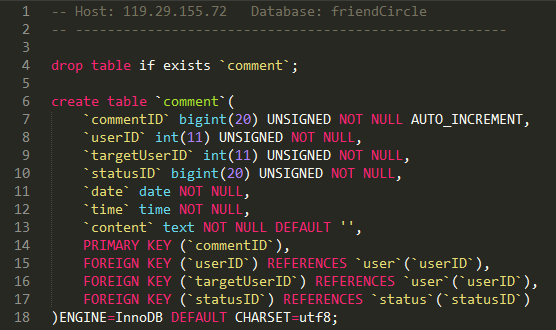


status表保存所有用户的所有状态，其中：

* 1. 以statusID作为主键；
  2. 以发状态用户的userID作为外键，与user表联系起来。

1. comment表

创建comment表的sql语句如下：

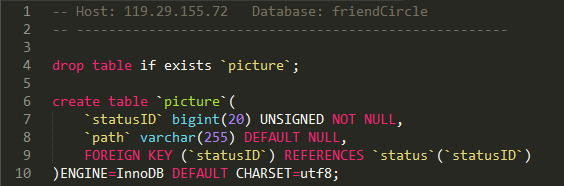


comment表保存所有的评论信息，包括两个方面：①对状态的评论；②对评论的回复，其中：

* 1. 以commentID作为主键；
  2. userID外键，引用user表，指向发起评论的用户；
  3. targetUserID外键，引用user表，指向评论所针对的用户；
  4. statusID外键，引用status表，指向评论所依附于的状态；

1. picture表

创建picture表的sql语句如下：



picture表保存用户状态所附加的图片信息，其中：

* 1. 没有主键，每个图片都通过外键statusID指向一条用户状态；
  2. 状态一旦发出，则不能更改图片（也不能更改文字），如果需要更换图片，只能删除状态，然后重新发布。