数据库建模设计说明书

[数据库建模设计说明书 1](#_Toc452718100)

[一、文档编写说明 2](#_Toc452718101)

[二、关系型数据库建设 2](#_Toc452718102)

[1. 用户信息管理模块 2](#_Toc452718103)

[1.1注册信息表  uers\_registry 3](#_Toc452718104)

[1.2安全信息表 user\_security 3](#_Toc452718105)

[1.3用户生日表 user\_birth 3](#_Toc452718106)

[1.4用户地域分布表 user\_location 4](#_Toc452718107)

[1.5用户经验表 user\_exp 4](#_Toc452718108)

[1.6用户资产表 user\_reset 4](#_Toc452718109)

[1.7数据表关系图 5](#_Toc452718110)

[2.消息和会话管理模块 5](#_Toc452718111)

[2.1会话表session 5](#_Toc452718112)

[2.2会话详细内容表sesion\_detail\_xxx 6](#_Toc452718113)

[2.3数据表关系图 6](#_Toc452718114)

[3.问答功能模块 6](#_Toc452718115)

[3.1问题汇总列表question\_count 6](#_Toc452718116)

[3.2问题回答表question\_reply 7](#_Toc452718117)

[3.3问题评论表question\_comment 7](#_Toc452718118)

[3.4问题被关注表question\_follow 7](#_Toc452718119)

[3.5数据表关系图 8](#_Toc452718120)

[4文章阅读模块 8](#_Toc452718121)

[4.1文章汇总列表article\_count 8](#_Toc452718122)

[4.2文章评论表article\_comment 9](#_Toc452718123)

[4.3文章被点赞表article\_star 9](#_Toc452718124)

[4.4问题被举报表question\_follow 9](#_Toc452718125)

[4.5数据表关系图 9](#_Toc452718126)

[5. 订单管理模块 10](#_Toc452718127)

[5.1充值订单管理recharge \_order 10](#_Toc452718128)

[5.2消费订单管理consume\_order 10](#_Toc452718129)

[5.3物品信息表item\_detail 11](#_Toc452718130)

[5.4数据表关系图 11](#_Toc452718131)

[三、关系型数据库对未来发展的限制 11](#_Toc452718132)

[1.关系型数据库的局限 11](#_Toc452718133)

[2.新型数据库的应用 12](#_Toc452718134)

# 一、文档编写说明

编写本说明书是为了重点说明本站数据库的结构、建设及实施方案等内容。本说明书先介绍了在关系型数据模型下的基本数据表及各表间的关系，再讨论了在大规模建站情况下关系型数据库的劣势，最后讨论了向非关系型数据库的迁移问题。

本说明书将作为系统维护的重要参考资料，也是后续更改数据库架构工作的指南。

# 二、关系型数据库建设

根据对系统所做的需求分析、系统设计、我方在关系型数据库中针对各个功能模块规划出相关数据表。本套设计方案撰写了非常多的表格，目的是防止信息负载过大的情况下遍历一张表带来的性能损耗。例如，每个用户的资产、每篇文章的评论内容都各自建立表格，而不是建在一个巨大的表中。

作为数据库的设计者，我深知SQL在多表链接的时候会长生巨大的性能损耗，所以我在本说明书的后部分做了关于非关系型数据库设计的畅想。我认为本网站（以及其他类似本网站的大型社交型网站）是不适合RDBM的。

本节讨论了在关系型数据库下的表设计RDBM。

## 用户信息管理模块

本模块管理用户的各类信息，包括有关用户安全的信息、有关用户生日和地理信息的信息，有关用户资产和经验的信息等等。其中用户ID是最终要的主键。

### 1.1注册信息表  uers\_registry

本表在用户注册本站时就建立，其中照片和简介非必须。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | **默认值** | 允许非空 | 备注 |
| user\_id | varchar(16) |  | NOT NULL | 主键/用户ID |
| user\_name | varchar(30) |  | NOT NULL | 昵称 |
| user\_gender | enum |  | NOT NULL | 性别 |
| user\_pic | Blob |  | YES | 头像 |
| user\_intro | mediumtext | No | YES | 用户简介 |

**备注：**

1. 用户的ID是由注册时的手机号或者Email生成的随机十六进制值（手机号、邮箱和注册时间拼接而成的文本通过SHA-1算法生成16位的用户ID）
2. 用户的名字可以包括中英文和数字（2-8字符），但不能包括特殊字符，本条通过正则表达式检验；
3. 用户头像为小于2MB的png、jpeg、jpg格式图片。用户的头像如果为空，系统会自动指派一个图片；
4. 用户性别可以在男、女和保密中选择。

### 1.2安全信息表 user\_security

本表记录和安全登录相关的全部信息，权限非常之高，注意保护。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| user\_id | varchar(16) |  | NOT NULL | 主键/用户ID |
| user\_tel | int unsigned |  |  | 注册电话(13位) |
| user\_email | varchar |  |  | 注册邮箱 |
| user\_pwd | varchar(32) |  | NOT NULL | 密码 |
| user\_pwd\_q | varchar(32) | No | NOT NULL | 密码保护问题 |
| user\_pwd\_a | varchar(32) |  | NOT NULL | 密码保护问题答案 |

**备注**

1. 电话号码必须是13位或者11位的数字(xxxxxxxxxxx或者xx-xxxxxxxxxxx), 本条通过正则表达式检验;
2. 用户邮箱必须是 xx@xx.com的格式，本条通过正则表达式检验；
3. 用户的密码必须大于6位而且小于20位，包括大写英文、小写英文、特殊字符和数字中两者以上，本条通过正则表达式检验；
4. 用户的密码保护问题和密码保护问题的答案必须小于15个中文字符或者30个英文字符，本条通过正则表达式检验；
5. 注册手机号和注册邮箱只要一个就可以，但是鼓励用户全部使用，如果两者都没有，就发出警报。

### 1.3用户生日表 user\_birth

本表记录用户的生日，一般意义上不允许修改。在不给予write权限的情况下，可以加index而不用担心read的效率。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| user\_id | varchar(16) |  | NOT NULL | 主键/用户ID |
| user\_birth\_year | int unsigned |  | YES | 用户出生年份 |
| user\_birth\_month | enum |  |  | 用户出生月份 |
| user\_birth\_day | enum |  |  | 用户出生日期 |

**备注**：

1. 用户出生年份从1900年开始，直到系统计算的YEAR()值；
2. 用户出生月份只能从1~12这些值中选择；
3. 用户出生日期只能从1~31这些值中选择，不同的月份不同的计算方法。

### 1.4用户地域分布表 user\_location

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| user\_id | varchar(16) |  | NOT NULL | 主键/用户ID |
| user\_nation | enum |  | YES | 国家 |
| user\_province | enum |  | YES | 省/州 |
| user\_city | enum |  | YES | 市 |
| user\_district | enum |  | YES | 区/县/郡 |

**备注**：

1. 用户的地理信息只能在enumeration值中选择；
2. 可以赋予用户修改权限；
3. 允许不填写本表；

### 1.5用户经验表 user\_exp

用户通过签到、评论、发表文章等等积极行为获得的经验将被记录在本表，同时记录用户通过充值获得的金币数量。这个表对于用户的意义仅次于安全表（甚至和安全信息表同样重要）。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| user\_id | varchar(16) |  | NOT NULL | 主键/用户ID |
| user\_exp | int |  | NOT NULL | 用户经验值 |
| user\_rank | enum |  | NOT NULL | 用户等级 |
| user\_money | int |  | NOT NULL | 用户金钱数 |
| user\_membership | boolean |  | NOT NULL | 用户会员状态 |

备注：

1. 用户等级是enum，它只能在“normal”“fan”“superfan”中选择；
2. 用户会员状态是有或者无；

### 1.6用户资产表 user\_reset

本表记录用户购买的资产，包括鲜花和会员使用权。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| user\_item\_id | varchar(16) |  | NOT NULL | 用户拥有的物品ID |
| user\_item\_num | int |  | NOT NULL | 用户拥有的物品数 |

### 1.7数据表关系图

下图为用户信息管理模块各数据表关系图，其中除了用户资产表为每个用户一张外，其他的表格为全系统共用一张。



## 2.消息和会话管理模块

两个用户之间的全部信息交流组成一个会话(Session),其中每一个信息为一条消息(Message),每个消息和会话都有唯一标识的ID。

### 2.1会话表session

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| session\_id | varchar(16) | NOT | NOT NULL | 会话ID |
| talker\_id\_1 | varchar(16) | NOT | NOT NULL | 会话参与者ID |
| talker\_id\_2 | varchar(16) | NOT | NOT NULL | 会话参与者ID |

**备注**：

1. 在会话中首先发起会话的参与者成为talker1，另者为talker2；
2. 会话ID由两个对话者的用户ID生成；（ “talker\_id\_1+talker\_id\_1”的SHA-1值）

### 2.2会话详细内容表sesion\_detail\_xxx

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| message\_id | varchar(16) |  | NOT NULL | 主键/会话ID |
| sender\_id | varchar(16) |  | NOT NULL | 发送者ID |
| receiver\_id | varchar(16) |  | NOT NULL | 接受者ID |
| message\_time | TIME |  | NOT NULL | 信息发送时间 |
| message\_text | mediumtext |  | NOT NULL | 信息内容 |

**备注**：

1. 消息ID是“message\_text+message\_time+sender\_id+receiver\_id”的SHA-1值；
2. 消息发送时间是yyyy-mm-dd-hh-ii-ss的时间格式；
3. 发送时间以信息发送时间为准；

### 2.3数据表关系图

下图为信息和会话管理模块各数据表关系图，其中会话详细信息表是每个用户一张，会话表为全系统共用一张。



## 3.问答功能模块

用户可以提出合乎社区规范的问题，其他用户可以回答该问题，或者在该问题下发表评论。有兴趣的用户可以关注问题，以便收到该问题以后的全部新回答。用户可以把心仪的答案加入到自己的收藏夹。对于不合乎社区规范或者法律法规的问题、评论或者答案，用户可以举报。

### 3.1问题汇总列表question\_count

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| question\_id | varchar(16) |  | NOT NULL | 主键/问题ID |
| quizzer\_id | varchar(16) |  | NOT NULL | 提问者ID |
| question\_time | TIME |  | NOT NULL | 提问时间 |
| question\_text | mediumtext |  | NOT NULL | 提问内容 |
| isAnswered | boolean | No | NOT NULL | 问题是否被回答 |
| question\_follow | int | 0 | NOT NULL | 被关注的数量 |
| question\_report | int | 0 | NOT NULL | 被举报的数量 |

**备注**：

1. question\_id是“question\_text+question\_time+quizzer\_id”的SHA-1；
2. 问题时间是yyyy-mm-dd-hh-ii-ss的时间格式；
3. 用户每天提问的上限是5个问题；
4. question\_text的内容要经过审核；（知乎提问条例？？）
5. 不符合相关条例的问题可以被举报，在question\_report\_num下设置trigger，当举报数达到一定数量以后让管理员仲裁是否关闭问题；

### 3.2问题回答表question\_reply

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| reply\_id | varchar(16) |  | NOT NULL | 主键/答案ID |
| replier\_id | varchar(16) |  | NOT NULL | 回答者ID |
| reply\_time | TIME |  | NOT NULL | 回答时间 |
| reply\_content | mediumtext |  | NOT NULL | 回答内容 |
| reply\_report | int |  | NOT NULL | 回答举报数 |
| reply\_star | int |  | NOT NULL | 回答点赞数 |
| reply\_favorite | int |  | NOT NULL | 回答收藏数 |

**备注**：

1. 每个问题都有属于自身的question\_reply表，表名是question\_reply+question\_id，故此表有多个；
2. 不符合相关条例的问题可以被举报，在question\_report下设置trigger，当举报数达到一定数量以后让管理员仲裁是否关闭问题；

### 3.3问题评论表question\_comment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| comment\_id | varchar(16) |  | NOT NULL | 主键/评论ID |
| commentater\_id | varchar(16) |  | NOT NULL | 评论者ID |
| comment\_time | TIME |  | NOT NULL | 评论时间 |
| comment\_content | mediumtext |  | NOT NULL | 评论内容 |
| comment\_report | int |  | NOT NULL | 评论举报数 |
| comment\_star | int |  | NOT NULL | 评论点赞数 |

**备注**：

1. 每个问题都有属于自身的question\_comment表，表名是question\_comment+question\_id，此表有多个；
2. 不符合相关条例的问题可以被举报，在comment\_repor下设置trigger，当举报数达到一定数量以后让管理员仲裁是否关闭问题。

### 3.4问题被关注表question\_follow

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| follower\_id | varchar(16) |  | NOT NULL | 主键/关注者ID |
| follow\_time | TIME |  | NOT NULL | 关注时间 |

### 3.5数据表关系图

下图展示了问题回答模块中各个数据表的关系，除了问题汇总表仅有一张外，其他的表都为各个问题所拥有。



## 4文章阅读模块

客观深刻的文章是本站发展的核心之一。每个人都可以发表文章，其他用户可以评论、收藏或者赞赏该文章。如果该文章不符合社区规范或者法律法规，用户可以举报该文章。

### 4.1文章汇总列表article\_count

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| article\_id | varchar(16) |  | NOT NULL | 主键/文章ID |
| author\_id | varchar(16) |  | NOT NULL | 作者ID |
| publish\_time | TIME |  | NOT NULL | 发表时间 |
| article\_text | mediumtext |  | NOT NULL | 文章内容 |
| article\_star | int | 0 | NOT NULL | 被关注的数量 |
| article\_report | int | 0 | NOT NULL | 被举报的数量 |

**备注**：

1. article\_id是“article\_text+ publish\_time+author\_id”的SHA-1；
2. 发布时间是yyyy-mm-dd-hh-ii-ss的时间格式；
3. article\_text的内容要经过审核；（知乎提问条例？？）
4. 不符合相关条例的问题可以被举报，在article\_report下设置trigger，当举报数达到一定数量以后让管理员仲裁是否关闭问题；

### 4.2文章评论表article\_comment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| comment\_id | varchar(16) |  | NOT NULL | 主键/评论ID |
| commentater\_id | varchar(16) |  | NOT NULL | 评论者ID |
| comment\_time | TIME |  | NOT NULL | 评论时间 |
| comment\_content | mediumtext |  | NOT NULL | 评论内容 |
| comment\_report | int |  | NOT NULL | 评论举报数 |
| comment\_star | int |  | NOT NULL | 评论点赞数 |

**备注**：

1. 每个文章都有属于自身的article\_comment表，表名是article\_comment+article\_id，此表有多个；
2. 不符合相关条例的问题可以被举报，在comment\_report下设置trigger，当举报数达到一定数量以后让管理员仲裁是否关闭问题；

### 4.3文章被点赞表article\_star

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| landator\_id | varchar(16) |  | NOT NULL | 主键/点赞者ID |
| landator\_time | TIME |  | NOT NULL | 点赞时间 |

### 4.4问题被举报表question\_follow

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| reporter\_id | varchar(16) |  | NOT NULL | 主键/举报者ID |
| report\_time | TIME |  | NOT NULL | 举报时间 |

### 4.5数据表关系图

下图展示了文章阅读模块中各个数据表的关系，除了文章汇总表仅有一张外，其他的表都为各个文章所拥有。



## 订单管理模块

### 5.1充值订单管理recharge \_order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| recharge\_id | varchar(16) |  | NOT NULL | 主键/充值订单ID |
| recharger\_id | varchar(16) |  | NOT NULL | 充值者ID |
| recharge\_time | TIME |  | NOT NULL | 充值时间 |
| recharge\_sum | int |  | NOT NULL | 充值金额 |
| recharge\_mode | enum |  | NOT NULL | 充值方式 |

备注:

1. 单次充值金额小于10,000而且单日充值金额小于100,000；
2. 充值方式现在仅在支付宝、微信支付、网银和话费支付中选择，如果未来有新增渠道，将做变更；

### 5.2消费订单管理consume\_order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| consume\_id | varchar(16) |  | NOT NULL | 主键/消费订单ID |
| consumer\_id | varchar(16) |  | NOT NULL | 消费者ID |
| consume\_time | TIME |  | NOT NULL | 消费时间 |
| consume\_item\_id | enum |  | NOT NULL | 消费物品ID |
| consume\_num | int |  | NOT NULL | 消费物品个数 |

备注：

1. 为了减少数据的冗余性，本表不记载消费物品的单价和本次消费的总价。消费物品的单价可以在物品单价表中用SQL语句联合查询；本次消费的总金额可以通过单价和购买个数的乘积计算得，故略去。
2. 消费物品ID是enum类型是因为消费物品只能是物品信息表中含有的物品。

### 5.3物品信息表item\_detail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **数据类型** | **默认值** | **允许非空** | **备注** |
| item\_id | varchar(16) |  | NOT NULL | 主键/物品ID |
| item\_name | varchar(16) |  | NOT NULL | 物品名称 |
| item\_unit\_price | TIME |  | NOT NULL | 物品单价 |

### 5.4数据表关系图

下图展示了订单管理模块中各个数据表的关系：



## 三、关系型数据库对未来发展的限制

本站的目标是通过专家和读者的有效互动形成一流的行情社交平台，所以本站的目标是持续吸收巨大的专家和股民群体，持续鼓励用户之间畅所欲言、相互增进价值。上述的目标也对本网站的数据库设计提出了两个挑战：

1. 数据库要支持巨大的用户数；
2. 数据库必须支持任何两个数据库用户的数据关联操作；（任何两个用户都可能交流）

## 1.关系型数据库的局限

关系型数据库（例如本网站现阶段所用的MySQL以及本网站发展中期准备使用的Oracle）的查询语言SQL大量使用多表连接操作,但随着越来越多大规模工作负荷应用的发行,对可伸缩性的需求可能会变得无比庞大。

关系数据库的确能伸缩自如,但通常只能在单台服务器节点上进行。例如采用表分区技术,一个表格可以由多个物理文件组成,虽然表格的容量增大了,但该表格仍然只能由一数据库引擎管理;另外增加一物理文件时,表格Schema得做改动,也就是还不能支持动态扩容。一旦单节点的能力抵达上限,就得通过多服务器节点来扩展来分发负载。

这时关系数据库的复杂性就开始影响其潜在的扩展规模了。RDBMS支持分区视图(Partition View) 技术,也就是支持联合数据库(Federated Database)。一个分区视图可以由多个分布在不同数据库节点服务器上的表格组合而成,数据库用户看到的只是该视图,不必关心物理表格。通过数据水平分割技术,分区视图把负载分担到多个数据库节点服务器上。扩容时,该方法除了需改动视图定义外,分区视图成为分布式数据库系统的中心,存在单点故障问题。另外,跨数据库节点之间多表格间连接操作的支持出现极大困难。

当试图扩展数据库系统到成百上千个节点时,将导致不堪复杂性之重负,这一特点使得RDBMS在大型分布式系统平台市场里的生存能力被大幅削减。

## 2.新型数据库的应用

为了能向客户提供一个伸缩自如的空间存放应用数据,本站只能追求一种新型的专注于可扩展性的数据库系统,而牺牲掉关系数据库所带来的其他好处。NoSQL是非关系型数据存储的广义定义,它打破了长久以来关系型数据库与ACID理论大一统的局面。NOSQL数据存储不需要固定的表结构,通常也不存在连接操作,在超大型数据存取上具备关系型数据库无法比拟的性能优势。