
同濟大學
TONGJI UNIVERSITY

# 数据库系统原理课程设计
# “一梦江湖”数据分析器

姓　　名：张天昊

学　　号：1851850

所在院系：电子与信息工程学院

学科专业：计算机科学与技术系

指导教师：关佶红

# 目录

# 1 需求分析

## 1.1 项目概述

### 1.1.1 总述

针对游戏“一梦江湖”，将其角色数据的存储作为素材进行需求分析和关系数据库的设计，并选择仿照网易藏宝阁、游戏数据助手一类的前端进行数据库的访问、展示以及功能实现。

### 1.1.2 背景

在实际游戏过程中，由于没办法实现对资源（装备、武学等）的随意分配，因此很难对资源的分配方式进行测试。例如潜力点数分配完成之后，如果想要进行调整是需要花费游戏代币的，所以在游戏内部无法进行自由的测试。为了实现一个自由测试功能，需要将游戏内的资源以数据形式存入数据库，通过游戏助手的形式，将这些内容进行组合来尝试最终成果。本项目即基于上述过程来进行设计与实现。

### 1.1.3 功能描述

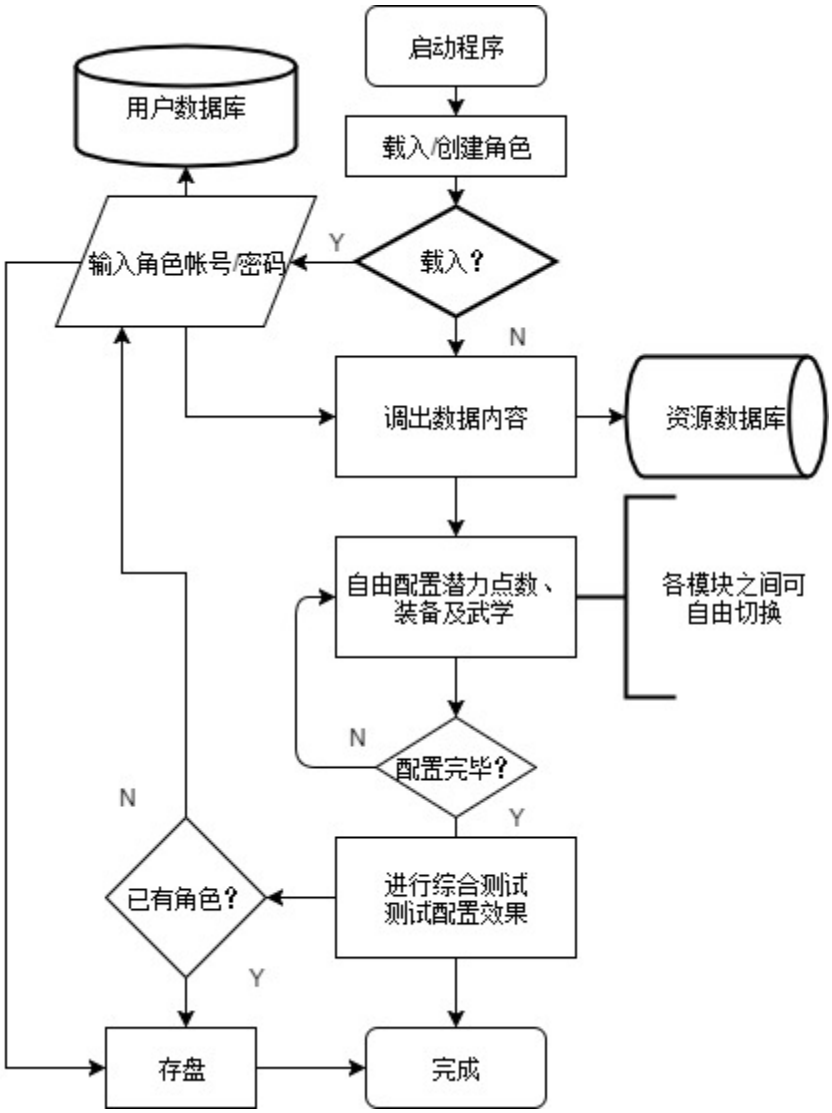
旨在实现一个游戏角色属性模拟器，用户可以通过该模拟器创建自己的虚拟游戏角色，并自由的为该角色设定等级、分配潜力点数、穿戴装备、配备武功等，最终可以看到完整的角色属性界面。

在数据的层次之上，还需要加入一些实际的测试场景，例如实战效果等等。这里拟定以文字描述或再辅以图形的方法进行实现。

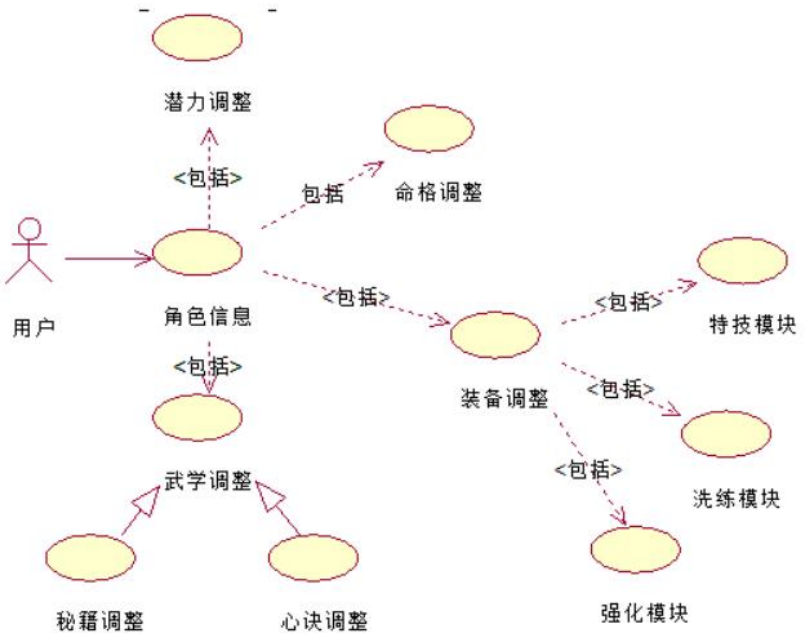
通过该模拟器，可以方便的探索合理的天赋、装备、武功搭配，因此该模拟器的功能可定义为一个简单的“游戏助手”。

## 1.2 流程概况

### 1.2.1 功能流程



### 1.2.2 数据用例




## 1.3 需求描述

阶段性批注：对于需求部分需要注意

其一，功能上的完整需求描述，任何功能的实现可能都需要提到底层数据库，与底层数据库进行交互等等，要进行详细描述。

其二，在数据库的设计阶段，对于软件标准的设计，即不会

### 1.3.1 功能需求

1、提供账户功能，主要保证用户可以对当前配置完成的状态进行存盘，因为是本机软件，不涉及联网功能，所以密码可有可无，用户可自行选择是否使

用密码功能对当前配置完成的角色状态进行二次保护。因此需要设计与其相应的用户数据库。

2、软件需要对游戏内的现有资源进行拷贝并在软件内进行自由调用与分配，因此需要多个实体模块作为资源本身的静态数据库，例如潜力影响表、命格影响表，装备列表，武学列表等。同时应实现后续添加实体模块（新增资源种类）的功能。

3、要实现各种资源数据的配置与计算，最终体现在界面上是一个完整的属性框架。这部分仅用数据库无法完全实现，应在高级程序配以相应的整合算法。但如果角色要保存当前的状态，需要有角色当前已分配的动态数据库，与用户数据库实现一个一对一关系。

4、对于部分资源模块，会有动态数据部分，例如装备模块中包含洗练属性。对于这些随机生成属性的模块，用户可以模拟整个过程，或是对该模块的属性进行自由配置。而这些内容的保存对于数据库而言是动态的，需要随用户的存盘进行频繁的修改。（所有的详细信息具体将在数据部分给出）

### 1.3.2 平台需求

- 支持 Windows10 平台

### 1.3.3 性能需求

1、实时性：对任意细节的修改影响属性界面时，要求属性能够瞬时响应（保守估计响应时间需小于 0.1s），需要保证数据库体量不能过大，在数据库设计时应纳入考虑。

2、安全性：对任意具有约束的数据项，需要保证高级程序与底层数据库的二重保障，即在高级程序中需要对错误的数据进行判断并给出提示；假设存在错误数据传递给数据库，当数据库拒绝接受时，高级程序也应捕获错误并回传提示。

### 1.3.4 界面需求

- 尽量平铺化设计，减少界面的调用层数。
- 交互以图形、图标为主，文字为辅。
- 主界面排布需清晰了然，在保证清晰的情况下尽量简洁。

### 1.3.5 维护需求

- 从数据角度，要求提供管理员帐号进行静态数据库的维护和对动态数据库的自由更改权限。
- 从软件角度，要求程序在最大限度上保留后续新功能添加的可行性。
- 从代码角度，要求可读性强，模块可分割性强，即在模块设计时遵循“高内聚低耦合”的原则。

## 1.4 数据部分

### 1.4.1 静态数据

- 用户信息表（UserInfo）
- 装备信息表
- 秘籍表
- 心诀表
- 特技表
- 洗练属性类型表

### 1.4.2 动态数据

- 角色潜力表
- 秘籍-角色关系表
- 心诀-角色关系表
- 角色-装备-(特技+强化+洗练)关系表

### 1.4.3 数据字典

用户信息表：

| 字段含义 | 字段名 | 类型 | 允许为空 |
| --- | --- | --- | --- |
| 帐号 | id | varchar | 否 |
| 密码 | pwd | varchar | 是 |
| 角色号 | Cid | int | 否 |

装备信息表：

| 字段含义 | 字段名 | 类型 | 允许为空 |
| --- | --- | --- | --- |
| 装备号 | id | int | 否 |
| 装备名 | name | varchar | 否 |
| 部位 | Part | varchar | 否 |
| 门派 | Sect | varchar | 否 |
| 属性 1 | attu_id1 | int | 是 |
| 基础值 1 | base1 | Double | 是 |
| 属性 2 | attu_id2 | int | 是 |
| 基础值 2 | base2 | Double | 是 |
| 属性 3 | attu_id3 | int | 是 |
| 基础值 3 | base3 | Double | 是 |
| 属性 4 | attu_id4 | int | 是 |
| 基础值 4 | base4 | Double | 是 |
| 属性 5 | attu_id5 | int | 是 |
| 基础值 5 | base5 | Double | 是 |
| 属性 6 | attu_id6 | int | 是 |
| 基础值 6 | base6 | Double | 是 |

秘笈表：

| 字段含义 | 字段名 | 类型 | 允许为空 |
| --- | --- | --- | --- |
| 秘笈号 | id | int | 否 |
| 秘笈名 | name | varchar | 否 |
| 领悟属性码 | qid | varchar | 否 |
| 领悟属性值 | qnum | int | 否 |
| 特效 id | sid | int | 否 |

心诀表：

| 字段含义 | 字段名 | 类型 | 允许为空 |
| --- | --- | --- | --- |
| 心诀号 | id | int | 否 |
| 心诀名 | name | varchar | 否 |
| 特效 id | sid | int | 否 |

特技表：

| 字段含义 | 字段名 | 类型 | 允许为空 |
| --- | --- | --- | --- |
| 特技号 | id | int | 否 |
| 特技名 | name | varchar | 否 |
| 特效 id | sid | int | 否 |

属性类型表：

| 字段含义 | 字段名 | 类型 | 允许为空 |
| --- | --- | --- | --- |
| 属性号 | id | int | 否 |
| 属性名 | name | varchar | 否 |

| | | | |
|---|---|---|---|
| 最大值 | max | Double | 否 |
| 基础值 | base | Double | 否 |

角色状态表：

| 字段含义 | 字段名 | 类型 | 允许为空 |
|---|---|---|---|
| 角色号 | id | int | 否 |
| 角色名 | name | varchar | 否 |
| 门派 | Sect | varchar | 否 |
| 潜力-体 | Pt_t | int | 否 |
| 潜力-劲 | Pt_j | int | 否 |
| 潜力-气 | Pt_q | int | 否 |
| 潜力-御 | Pt_y | int | 否 |
| 潜力-敏 | Pt_m | int | 否 |

秘籍-角色关系表

| 字段含义 | 字段名 | 类型 | 允许为空 |
|---|---|---|---|
| 角色号 | id | int | 否 |
| 秘笈 1 | mid_1 | int | 是 |
| 秘笈 1 等级 | mlv_1 | int | 是 |
| 秘笈 1 阶数 | ran_1 | int | 是 |
| 秘笈 2 | mid_2 | int | 是 |
| 秘笈 2 等级 | mlv_2 | int | 是 |
| 秘笈 2 阶数 | ran_2 | int | 是 |
| 秘笈 3 | mid_3 | int | 是 |

| | | | |
|---|---|---|---|
| 秘笈 3 等级 | mlv_3 | int | 是 |
| 秘笈 3 阶数 | ran_3 | int | 是 |
| 秘笈 4 | mid_4 | int | 是 |
| 秘笈 4 等级 | mlv_4 | int | 是 |
| 秘笈 4 阶数 | ran_4 | int | 是 |

心诀-角色关系表

| 字段含义 | 字段名 | 类型 | 允许为空 |
|---|---|---|---|
| 角色号 | id | int | 否 |
| 自创武学名 | name | varchar | 否 |
| 心诀号 1 | hid_1 | int | 否 |
| 心诀号 2 | hid_2 | int | 否 |

角色-装备-(特技+强化+洗练)关系表

| 字段含义 | 字段名 | 类型 | 允许为空 |
|---|---|---|---|
| 角色号 | id | int | 否 |
| 装备号 | zid | int | 否 |
| 装备等级 | zlevel | int | 否 |
| 强化等级 | qlevel | int | 否 |
| 洗练属性 1 | xlid1 | int | 是 |
| 百分比 1 | percent1 | Double | 是 |
| 洗练属性 2 | xlid2 | int | 是 |
| 百分比 2 | percent2 | Double | 是 |
| 洗练属性 3 | xlid3 | int | 是 |
| 百分比 3 | percent3 | Double | 是 |

| | | | |
|---|---|---|---|
| 洗练属性 4 | xlid4 | int | 是 |
| 百分比 4 | percent4 | Double | 是 |
| 洗练属性 5 | xlid5 | int | 是 |
| 百分比 5 | percent5 | Double | 是 |
| 特技 1 | tid1 | int | 是 |
| 特技 2 | tid2 | int | 是 |
| 特技 3 | tid3 | int | 是 |

## 1.5 实现部分

### 1.5.1 设计工具

- ✓ 使用 Word 进行文档编制
- ✓ 使用 Power Designer 进行数据库的设计
- ✓ 使用 Rational Rose 进行 UML 的绘制
- ✓ 其它还没想好

### 1.5.2 开发工具

- ✓ 没想好+10086

# 2 概念设计

## 2.1 概述

对于整个概念设计，重点完成以下几个工作：

- 将需求分析中根据经验得到的数据字典拆分整合为实体与实体间的关系，要求尽可能保证关系中尽量不出现属性，实体中的属性尽量减少冗余。
- 将数据字典改为实体表时，要注重重新修改实体表与属性的命名以符合数据库的规范化表达。
- 分析表达实体以及实体之间关系的方法，以数据字典为参照，必要时可新增或删减字段以优化设计。

## 2.2 实体描述

从概念上讲，实体即某一类事物的集合，有其客观存在性。因此联系需求分析，可以将其中所有的静态数据划为实体集。

对于动态数据而言，其重点突出的是当静态数据发生结合时，其上衍生出的一些体现结合作用的属性，对于这些属性而言，可以作为联系本身存在，也可以作为“联系的记录的实体”存在，尤其当静态实体很少修改时，将动态属性作为单独实体也是一种较优方案，因此部分动态数据也将被划为实体集。

### 2.2.1 用户信息表

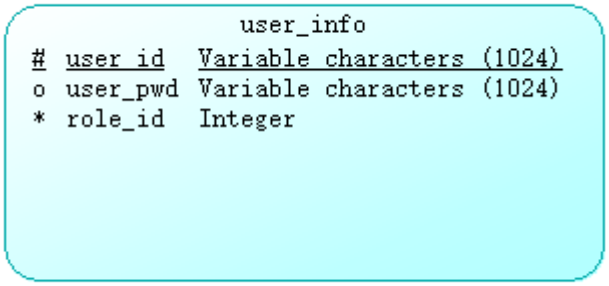
**实体说明**

用户信息表用于用户载入已配置好资源的角色，或是保存当前的配置状态。

**属性说明**

| 属性 | 字段名 | 描述 |
| --- | --- | --- |
| 帐号 | user_id | 用户用来确定角色的唯一标识 |
| 密码 | user_pwd | 用于对角色进行二次保护，可为空 |
| 角色号 | role_id | 用于指明用户与角色的对应关系 |

**实体图**




### 2.2.2 装备信息表

**实体说明**

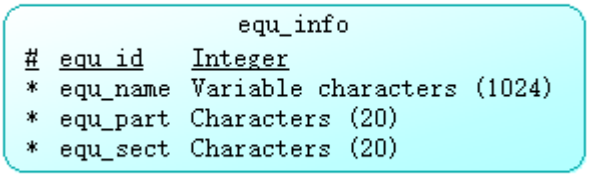
用于存储装备的基本信息，如装备名等一些不随用户不同而改变的值，即装备本身的 attribute。

**属性说明**

| 属性 | 字段名 | 描述 |
| --- | --- | --- |

| 装备号 | equ_id | 用来确定每种装备的唯一标识 |
| --- | --- | --- |
| 装备名 | equ_name | 装备本身的名字 |
| 部位 | equ_part | 装备的部位，如衣服、护腕等 |
| 门派 | equ_sect | 使用装备的角色所需的门派 |

**实体图**




### 2.2.3 秘笈表

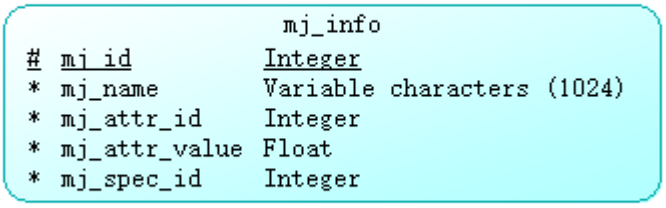
**实体说明**

用于存储游戏中的所有可选用秘笈的基本信息，是一些不随用户不同而改变的值。

**属性说明**

| 属性 | 字段名 | 描述 |
| --- | --- | --- |
| 秘笈号 | mj_id | 用来确定每种秘笈的唯一标识 |
| 秘笈名 | mj_name | 秘笈本身的名字 |
| 领悟属性码 | mj_attr_id | 秘笈作用于角色提升的属性类型 |
| 领悟属性值 | mj_attr_value | 秘笈作用于角色提升的属性数值 |
| 特效 id | mj_spec_id | 对应高级语言实现的特效算法 |

**实体图**



### 2.2.4 心诀表

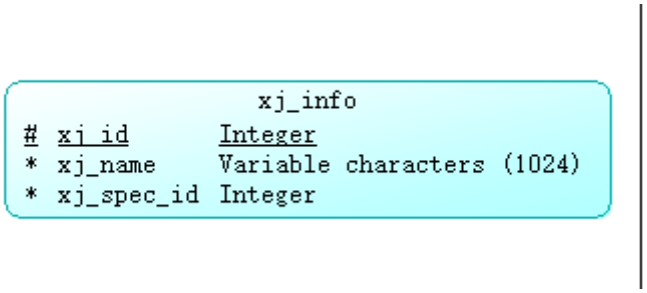
**实体说明**

用于存储游戏中的所有可选用心诀的基本信息，是一些不随用户不同而改变的值。

**属性说明**

| 属性 | 字段名 | 描述 |
| --- | --- | --- |
| 心诀号 | xj_id | 用来确定每种心诀的唯一标识 |
| 心诀名 | xj_name | 心诀本身的名字 |
| 特效 id | xj_spec_id | 对应高级语言实现的特效算法 |

**实体图**




### 2.2.5 特技表

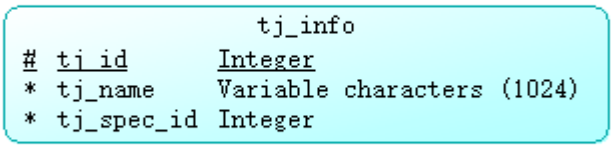
**实体说明**

用于存储游戏中的所有可选用特技的基本信息，是一些不随用户不同而改变的值。

**属性说明**

| 属性 | 字段名 | 描述 |
| --- | --- | --- |
| 特技号 | tj_id | 用来确定每种特技的唯一标识 |
| 特技名 | tj_name | 特技本身的名字 |
| 特效 id | tj_spec_id | 对应高级语言实现的特效算法 |

**实体图**




### 2.2.6 属性类型表

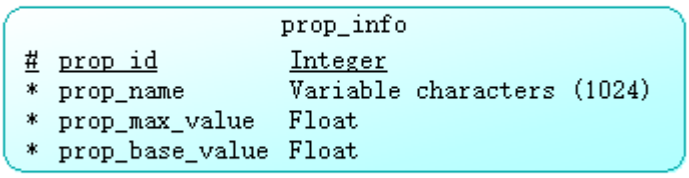
**实体说明**

用于存储游戏中的所有角色属性，每个角色都应该具有其上的所有属性的一个值，该值不存储在数据库中，会经高级程序动态计算得出，而在数据库中重点存储这些值的类型以及某些模块会应用到的数据项。

**属性说明**

| 属性 | 字段名 | 描述 |
| --- | --- | --- |
| 属性号 | prop_id | 用来确定每种属性的唯一标识 |
| 属性名 | prop_name | 属性本身的名字 |
| 最大值 | prop_max_value | 对应于洗练模块的属性可能出现的最大值，特性决定其值应该固定 |
| 基础值 | prop_base_value | 对应于装备模块应该具有的值，特性同上 |

**实体图**




### 2.2.7 角色状态表

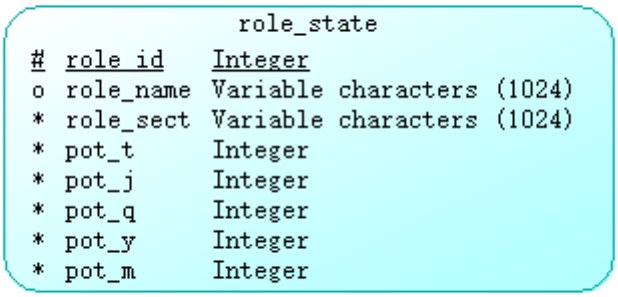
**实体说明**

用于存储一个角色最基本的有效信息，目前用到的基本信息包括自身属性和记录的潜力值，后续很有可能会对字段进行添加（应实现的需要）以扩展更复杂更贴合真实情况的功能。

**属性说明**

| 属性 | 字段名 | 描述 |
| --- | --- | --- |
| 角色号 | role_id | 用来确定角色的唯一标识 |
| 角色名 | role_name | 角色自身的名字，实际上可有可无 |
| 门派 | role_sect | 角色所在的门派，作为限制条件存在 |
| 潜力-体 | pot_t | 角色的“体”潜力值 |
| 潜力-劲 | pot_j | 角色的“劲”潜力值 |
| 潜力-气 | pot_q | 角色的“气”潜力值 |
| 潜力-御 | pot_y | 角色的“御”潜力值 |
| 潜力-敏 | pot_m | 角色的“敏”潜力值 |

**实体图**




### 2.2.8 秘笈使用表

**实体说明**

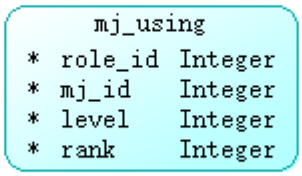
体现角色对秘笈的拥有关系，记录角色使用秘笈时存在的一些易变属性。

**属性说明**

| 属性 | 字段名 | 描述 |
| --- | --- | --- |
| 角色号 | role_id | 角色状态表的 PK |
| 秘笈号 | mj_id | 秘笈表的 PK |

| | | |
|---|---|---|
| 等级 | level | 角色当前使用的秘笈属性之一 |
| 阶数 | rank | 角色当前使用的秘笈属性之一 |

**实体图**




### 2.2.9 自创武学表

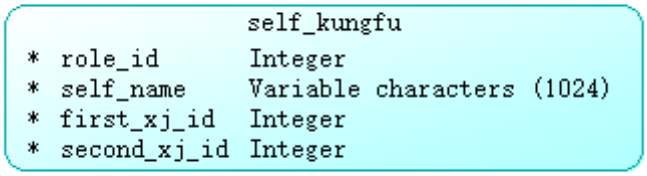
**实体说明**

对应数据字典中的心诀-角色关系表，体现角色对心诀的使用关系。

**属性说明**

| 属性 | 字段名 | 描述 |
|---|---|---|
| 角色号 | role_id | 角色状态表的 PK |
| 自创武学名 | self_name | 自创武学的名字 |
| 心诀号 1 | first_xj_id | 自创武学使用的第一个心诀 |
| 心诀号 2 | second_xj_id | 自创武学使用的第二个心诀 |

**实体图**



### 2.2.10 装备穿戴表

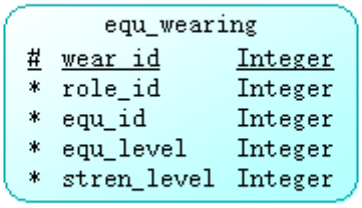
**实体说明**

角色对装备的使用关系，记录角色使用装备时的一些易变属性。

**属性说明**

| 属性 | 字段名 | 描述 |
| --- | --- | --- |
| 穿戴号 | wear_id | 描述该穿戴关系的主码 |
| 角色号 | role_id | 角色状态表的 PK |
| 装备号 | equ_id | 装备表的 PK |
| 装备等级 | equ_level | 角色当前使用的装备属性之一 |
| 强化等级 | stren_level | 角色当前使用的装备属性之一 |

**实体图**




### 2.2.11 装备洗练表

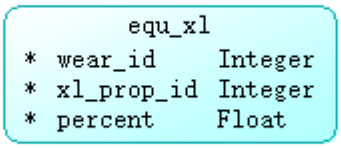
**实体说明**

对于每个已经被使用的装备（wear_id），可以对其进行洗练以获得洗练属性，因此需要洗练表存储该洗练属性值。

**属性说明**

| 属性 | 字段名 | 描述 |
| --- | --- | --- |
| 穿戴号 | wear_id | 描述该穿戴关系的 PK |
| 洗练属性号 | xl_prop_id | 对应属性表中的 PK |
| 百分比 | percent | 属性值相对于 max 的百分比 |

**实体图**

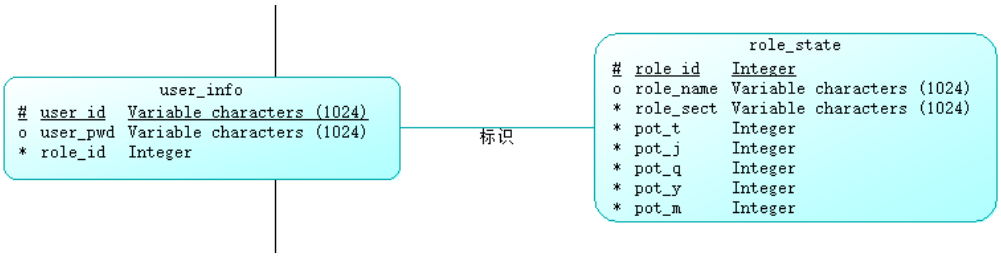



## 2.3 关系描述

在关系描述中，将从静态数据的关系出发，分析其中关系的复杂性，再提出以“记录”作为解决办法的“实体-关系-实体-关系-实体”的一种关系描述。给出各实体集间的局部关系。

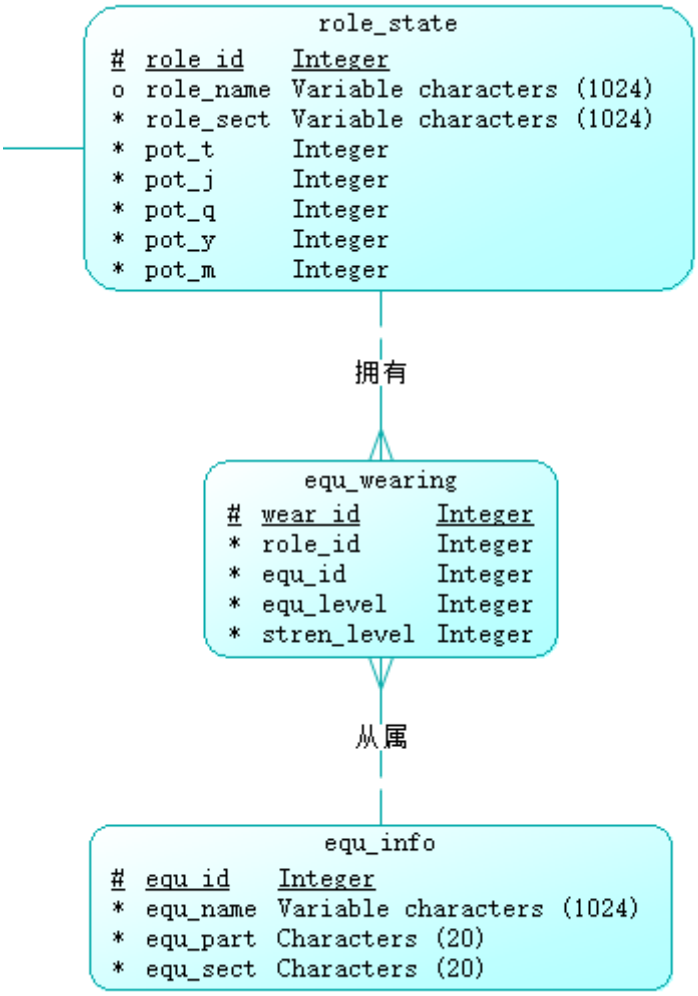
### 2.3.1 用户-角色关系描述

用户与角色具有一个 1 对 1 关系，具体描述为一个用户帐号对应一个已配置资源的角色，一个角色也只能由唯一一个账户进行标识。



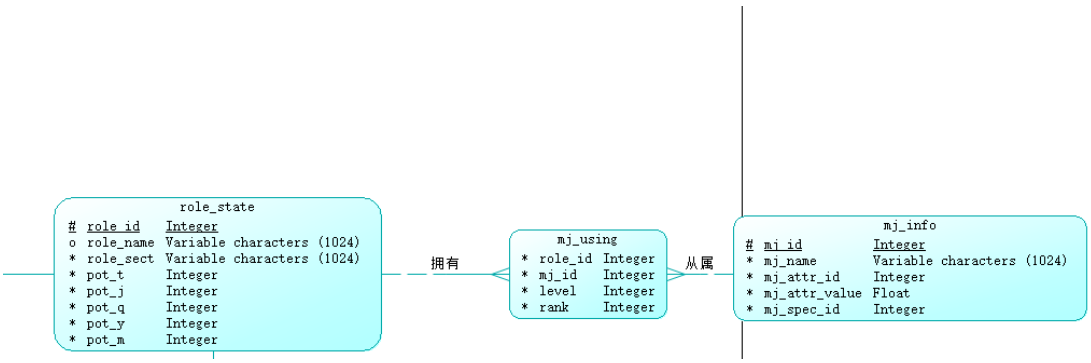

### 2.3.2 角色-装备关系描述

角色表与装备表本身应该是一个多对多关系，即一个角色可以对应多件装备，同一种装备也可能被多个角色使用，因此需要加入一个角色使用装备的记录表，该表用来将多对多关系拆分为两个一对多关系。



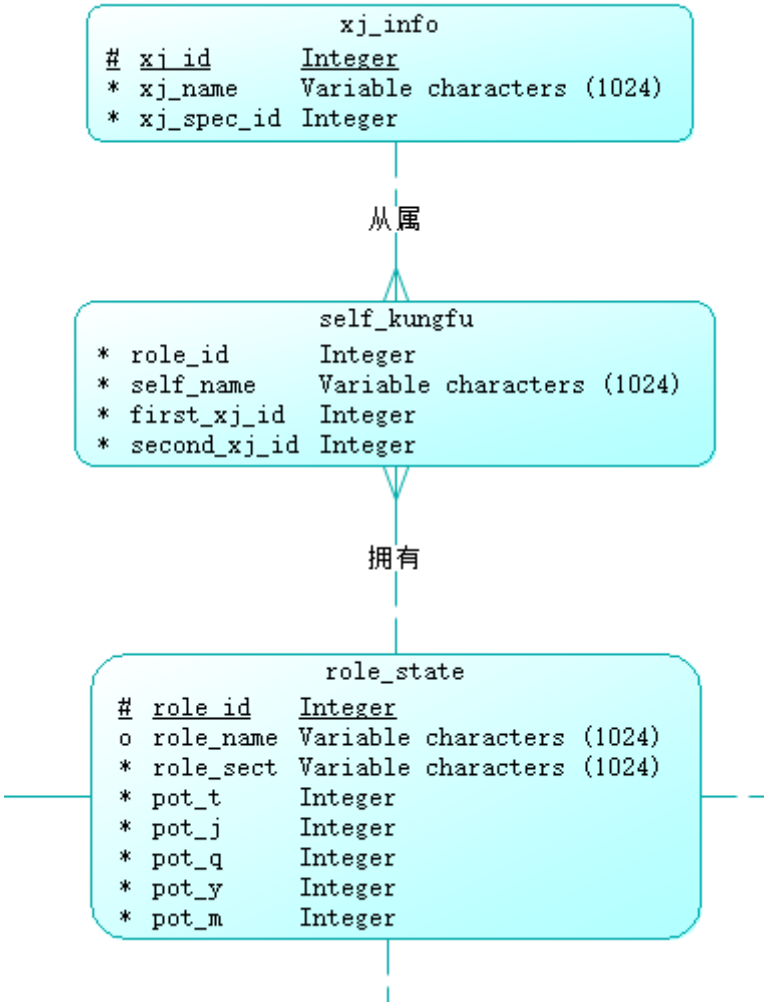

### 2.3.3 角色-秘笈关系描述

角色与秘笈本身也是多对多关系，因此加入秘笈使用记录进行拆分，结果如下：



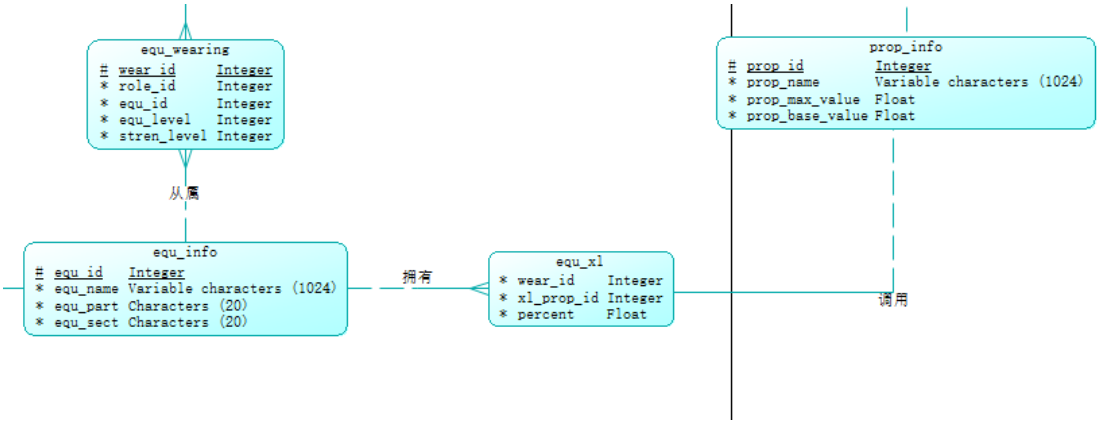
### 2.3.4 角色-心诀关系描述

角色与心诀的多对多关系使用自创武学表进行拆分，结果如下。注意自创武学表的每条记录应对应心诀表的两条记录，但由于是固定死的整数 2，所以实际上体现的依然是一对多关系，这里用一对多的表现形式进行体现。



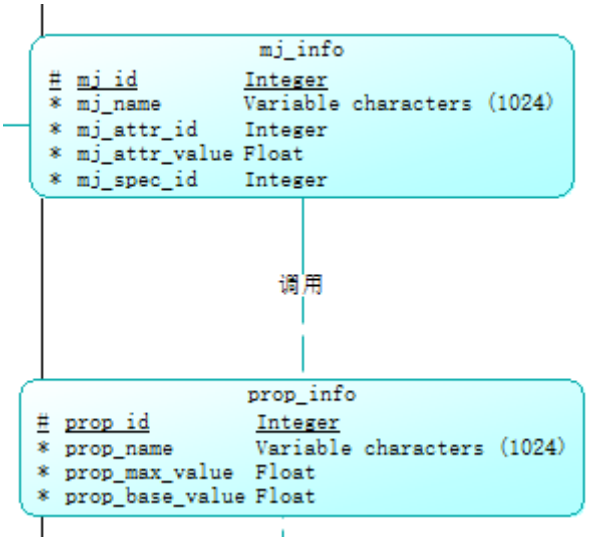

### 2.3.5 装备-属性关系描述

装备对属性的使用包含基础属性和洗练属性，都是仅对属性号进行使用，这里借用洗练属性表标记装备的基础属性，若 percent 为空则代表基础属性。这里需要注意的是，对于装备洗练表中出现的记录，即便洗练记录对属性表是一对一关系，但洗练记录若存在，则必然需要连接属性的一条 tuple，因此洗练记录一边要用实线。



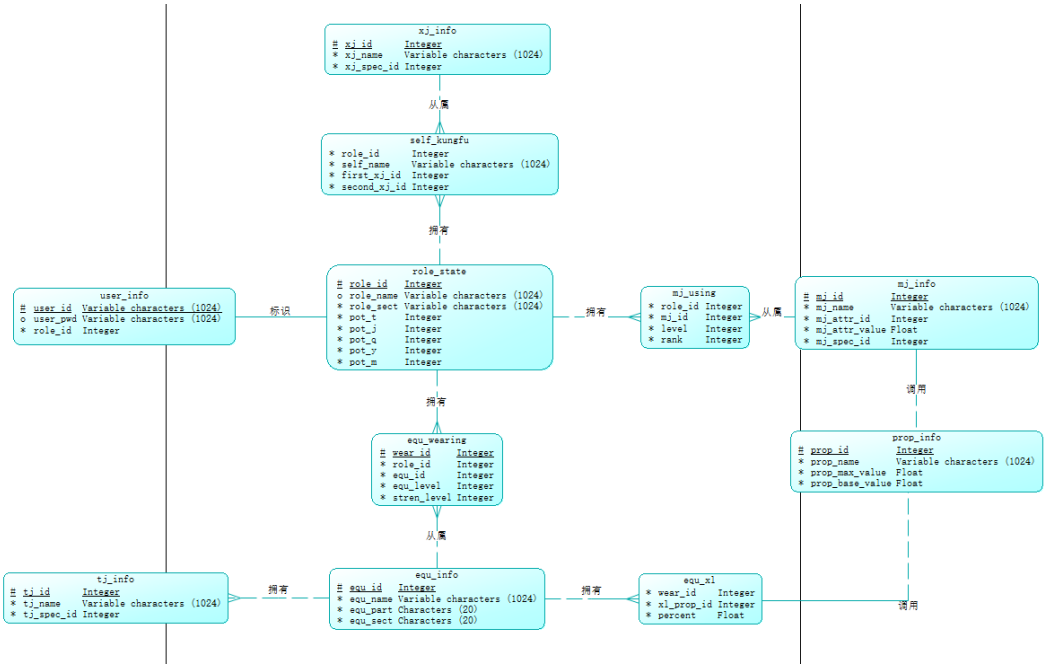

### 2.3.6 秘笈-属性关系描述

每个秘笈中需要引用属性的一条记录，实现一个一对一关系，这个逻辑比较简单，结果如下：




## 2.4 全局 E-R 图

综上，完整的全局 E-R 图显示如下：




最后对整体 E-R 图的符号进行解释。

实体字段前端的#代表非空主键，*代表非空，O代表可空。

关系中的实线代表强制关系，虚线代表无约束，单线代表 1，多线代表多。

# 3 逻辑设计

## 3.1 概述

在概念设计中，分析了与游戏助手有关的存储部分内容，并使用 E-R 图对要存储的实体以及实体之间的关系进行了描述。在逻辑设计中，需要对概念设计的 E-R 图转化为关系模式并对模型进行审视，具体将完成以下几个工作：

- 根据 E-R 图中的数据表给出关系模式的属性集，依赖关系集
- 对额外添加的概念设计中的实体表主键进行审视，若多余可进行剔除
- 对概念设计时的关系表进行判断，如果仍是弱实体集则添加新的属性，如果已有的属性已能保证关系表的属性原子性（即不具有多值），则去除概念设计中添加的主码
- 完成上述处理后已确保为 1NF，接下来若各关系模型不满足 3NF，将其改写为 3NF 形式，并给出各关系模型的函数依赖（作为后续设计时描述详细 Check 的判断条件的一部分）
- 对概念设计中表的结构的错误进行修改

- 事实上在概念设计时，已经尽可能保证表的结构尽量满足 3NF 要求，因此绝大多数表不需要进行改动。

## 3.2 关系模式描述

### 3.2.1 概述

在本节中将对概念设计中提出的所有表进行进一步审视，对每一个表都将完成上述的 5 个工作，因此概念设计中的一个表有可能会被拆分为多个表，最后给出所有表的关系模式总览。

### 3.2.2 用户信息表

**依赖关系描述**

在确定账号的情况下角色被唯一确定，因此 帐号→角色号

密码为帐号所用，被帐号决定，因此 帐号→密码

**属性审视**

对于上述依赖关系，目前存在的 3 个属性都是必须的，且无需增加新的属性。

由依赖关系可以确定 CK 为帐号，该表设计满足 3NF，无需拆分。

**关系模式**

$$U = \{user\_id, user\_pwd, role\_id\}$$

$$F = \{user\_id \rightarrow role\_id, user\_id \rightarrow user\_pwd\}$$

### 3.2.3 装备信息表

**依赖关系描述**

装备号作为一种装备的唯一标识，其内包含有关装备的所有信息，因此

装备号→装备名

装备号→部位

装备号→门派

**属性审视**

对于装备信息表，实际上有两种设计方式。一种是以装备名为决定因素，由装备名确定装备的部位和门派；但这样会存在一个问题，即需要有一个完整的装备命名规则保证装备名可以体现上述信息，否则无法建立检查关系。

因此，加入装备号，使装备名变成一个普通的属性，不与部位、门派相关。采用第 2 种设计思路，则表已经为 3NF，无需改动。

**关系模式**

$$U = \{equ\_id, equ\_name, equ\_part, equ\_sec\ t\}$$

$$F = \{equ\_id \rightarrow equ\_name, equ\_id \rightarrow equ\_part, equ\_id \rightarrow equ\_sec\ t\}$$

### 3.2.4 秘笈表

**依赖关系描述**

同装备信息的设计，引入秘笈号同样可以解决秘笈名命名问题。因此秘笈号可以决定秘笈名与特效 id，即：

秘笈号→秘笈名

秘笈号→特效 id

对于领悟属性码而言，每种秘笈会有一个特定的领悟属性，因此秘笈号可以决定秘笈属性码，但秘笈属性值会受到秘笈号本身（不同秘笈会因稀有度不同而属性值产生差异）和领悟属性码（不同属性码对应的属性类型和取值范围不同）的共同约束，即：

秘笈号→领悟属性码

(秘笈号，领悟属性码)→领悟属性值)

**属性审视**

对于前半部分的约束，可以保证设计没有问题，而对于后半部分的约束需要检测其 CK。

根据算法确定 CK 为秘笈号，因此（秘笈号，领悟属性码）为 SK，其满足 3NF 的定义，因此也不需要修改。

**关系模式**

$$U = \{mj\_id, mj\_name, mj\_attr\_id, mj\_attr\_value, mj\_spec\_id\}$$

$$F = \{mj\_id \rightarrow mj\_name, mj\_id \rightarrow mj\_attr\_id,$$

$$(mj\_id, mj\_attr\_id) \rightarrow mj\_attr\_value, mj\_id \rightarrow mj\_spec\_id\}$$

### 3.2.5 心诀表

**依赖关系描述**

同理，引入心诀号。

心诀号→心诀名

心诀号→特效 id

**属性审视**

CK = 心诀号

无属性缺失与冗余，满足 3NF。

**关系模式**

$$U = \{xj\_id, xj\_name, xj\_spec\_id\}$$

$$F = \{xj\_id \rightarrow xj\_name, xj\_id \rightarrow xj\_spec\_id\}$$

### 3.2.6 特技表

**依赖关系描述**

同理，引入特技号。

特技号→特技名

特技号→特效 id

**属性审视**

CK = 特技号

无属性缺失与冗余，满足 3NF。

**关系模式**

$$U = \{tj\_id, tj\_name, tj\_spec\_id\}$$

$$F = \{tj\_id \rightarrow tj\_name, tj\_id \rightarrow tj\_spec\_id\}$$

### 3.2.7 属性类型表

#### 依赖关系描述

一梦江湖中对属性的访问存在多个模块，每个模块访问属性时，其可见的部分都不同，这里我暂时没有找到按模块访问表的一种较好的设计（不打算将功能模块类型作为表存入数据库），因此目前的设计是，将不同模块访问表时所需要的所有字段都加入属性类型表，某个模块访问时自取所需，因此属性类型表中出现的一些内容可能会比较奇怪。

具体的依赖关系有：

属性号→属性名

属性号→最大值

属性号→基础值

#### 属性审视

引入属性号的道理同上，但该表在后续有可能会进行字段的添加已满足其它新补充的功能模块的需要（只是添加不会有删除，对已有模块的需访问部分不会有影响）

因此由于上述原因，类似最大值、基础值这类实际上是属于某个功能模块的值也被放入了属性类型表当中，这样做的目的是为了保证涉及到属性时，对属性的访问方式唯一。

无属性缺失与冗余，满足 3NF。

#### 关系模式

$$U = \{prop\_id, prop\_name, prop\_max\_value, prop\_base\_value\}$$

$$F = \{prop\_id \rightarrow prop\_name, prop\_id \rightarrow prop\_\max\ value,$$

$$prop\_id \rightarrow prop\_base\_value\}$$

### 3.2.8 角色状态表

#### 依赖关系描述

在角色状态中，角色号可以决定其它内容的唯一性。因此：

角色号→角色名

角色号→门派

角色号→五种潜力值

**属性审视**

CK = 角色号

在属性设计时，曾考虑将五种潜力值作为潜力类型表的实例并单独列出去，但是考虑后续实现中，没有添加其它潜力类型的需要，如果将五种潜力独立出去的话，需要额外添加潜力类型表与角色-潜力类型关系表，这样的设计是复杂且扩展意义不大的。

因此还是选择了列举五种潜力类型的设计方式，采用这种方式，除角色号外其它内容互不影响，五种潜力值的和满足约束即可（后续设计给出详细的值约束），满足 3NF。

**关系模式**

$$U = \{role\_id, role\_name, role\_sec\,t, pot\_t, pot\_j, pot\_q, pot\_y, pot\_m\}$$

$$F = \{role\_id \rightarrow role\_name, role\_id \rightarrow role\_sec\,t, role\_id \rightarrow pot\_t,$$

$$role\_id \rightarrow pot\_j, role\_id \rightarrow pot\_q, role\_id \rightarrow pot\_y, role\_id \rightarrow pot\_m\}$$

### 3.2.9 秘笈使用表

**依赖关系描述**

秘笈使用表描述角色与秘笈的联系，角色为主体，同一个角色可以拥有多个秘笈，同一种秘笈也可以被多个角色拥有，因此角色与秘笈不具有决定关系。

(角色号,秘笈号)→秘笈等级

(角色号,秘笈号)→秘笈阶数

**属性审视**

角色号和秘笈号可以作为外键与主键，等级和阶数之间有相互依赖关系，但由于 29 级为 3 阶，30 级可能为 3 阶可能为 4 阶，因此如果将等级和阶数进行依赖可能会导致多值，所以分开进行记录

**关系模式**

$$U = \{role\_id, mj\_id, level, rank\}$$

$$F = \{(role\_id, mj\_id) \rightarrow level, (role\_id, mj\_id) \rightarrow rank\}$$

### 3.2.10 自创武学表

#### 依赖关系描述

由于一个角色只能使用一个自创武学，因此可以不使用角色号，而是直接认为自创武学为角色的一部分属性，相当于角色状态表的一部分，但由于并不经常访问，所以独立出来。

角色号→自创武学名

角色号→心诀号 1

角色号→心诀号 2

#### 属性审视

心诀号 1 和心诀号 2 按理说可以拆分成角色-心诀的 1 对 1 关系表，但是这里固定为一个角色使用两个心诀，后续可以增添比较方便的连携作用，因此设定为 2 个心诀分开，也是合理的设计方式。

CK = 角色号，满足 3NF

#### 关系模式

$$U = \{role\_id, self\_name, first\_xj\_id, second\_xj\_id\}$$

$$F = \{role\_id \rightarrow self\_name, role\_id \rightarrow first\_xj\_id, role\_id \rightarrow second\_xj\_id\}$$

### 3.2.11 装备穿戴表

#### 依赖关系描述

与游戏相比，该游戏助手对同一种装备只能拥有一件（即穿在身上，不实现背包功能）

穿戴号→装备等级

穿戴号→强化等级

#### 属性审视

与概念设计相比，由于不实现背包功能，角色对于同一个装备号只能拥有一件，因此可以去掉主键穿戴号，用角色号和装备号共同拼成 CK，但是穿戴号可以帮助维护与其它表之间的关系，因此这里进行保留

CK：穿戴号，满足 3NF

#### 关系模式

$$U = \{wear\_id, \text{role\_id,equ\_id,equ\_level,stren\_level}\}$$

$$F = \{wear\_id \rightarrow equ\_level, wear\_id \rightarrow stren\_level\}$$

### 3.2.12 装备洗练表

**依赖关系描述**

由洗练序号决定穿戴号与洗练属性号，百分比为存值，不受依赖关系影响。

洗练序号→穿戴号

洗练序号→洗练属性号

**属性审视**

CK = 洗练序号，百分比

与概念设计相比，这里需要新增洗练序号，原因是一个装备可以对应多个洗练，每个洗练的洗练属性号和百分比都有可能相同，因此无法正确标识，需要新加属性值进行标识。

**关系模式**

$$U = \{xl\_id, wear_{id}, xl_{prop_{id}}, percent\}$$

$$F = \{xl\_id \rightarrow wear\_id, xl\_id \rightarrow xl\_prop\_id, percent\}$$

## 3.3 表结构设计

### 3.3.1 用户信息表

| 属性 | 字段名 | 类型 | 约束 |
|---|---|---|---|
| 帐号 | user_id | Varchar2 | PK / NOT NULL |
| 密码 | user_pwd | Varchar2 | 无 |
| 角色号 | role_id | Integer | FK / NOT NULL |

### 3.3.2 装备信息表

| 属性 | 字段名 | 类型 | 约束 |
|---|---|---|---|
| 装备号 | equ_id | Integer | PK / NOT NULL |
| 装备名 | equ_name | Varchar2 | NOT NULL |
| 部位 | equ_part | Char(20) | NOT NULL |

| 门派 | equ_sect | Char(20) | 无 |
|---|---|---|---|

### 3.3.3 秘笈表

| 属性 | 字段名 | 类型 | 约束 |
|---|---|---|---|
| 秘笈号 | mj_id | Integer | PK / NOT NULL |
| 秘笈名 | mj_name | Varchar2 | NOT NULL |
| 领悟属性码 | mj_attr_id | Integer | FK / NOT NULL |
| 领悟属性值 | mj_attr_value | Float | NOT NULL |
| 特效 id | mj_spec_id | Integer | NOT NULL |

### 3.3.4 心诀表

| 属性 | 字段名 | 类型 | 约束 |
|---|---|---|---|
| 心诀号 | xj_id | Integer | PK / NOT NULL |
| 心诀名 | xj_name | Varchar2 | NOT NULL |
| 特效 id | xj_spec_id | Integer | NOT NULL |

### 3.3.5 特技表

| 属性 | 字段名 | 类型 | 约束 |
|---|---|---|---|
| 特技号 | tj_id | Integer | PK / NOT NULL |
| 特技名 | tj_name | Varchar2 | NOT NULL |
| 特效 id | tj_spec_id | Integer | NOT NULL |

### 3.3.6 属性类型表

| 属性 | 字段名 | 类型 | 约束 |
|---|---|---|---|
| 属性号 | prop_id | Integer | PK / NOT NULL |
| 属性名 | prop_name | Varchar2 | NOT NULL |
| 最大值 | prop_max_value | Float | 无 |
| 基础值 | prop_base_value | Float | 无 |

### 3.3.7 角色状态表

| 属性 | 字段名 | 类型 | 约束 |
|---|---|---|---|
| 角色号 | role_id | Integer | PK / NOT NULL |

| 角色名 | role_name | Varchar2 | 无 |
| --- | --- | --- | --- |
| 门派 | role_sect | Varchar2 | NOT NULL |
| 潜力-体 | pot_t | Integer | NOT NULL |
| 潜力-劲 | pot_j | Integer | NOT NULL |
| 潜力-气 | pot_q | Integer | NOT NULL |
| 潜力-御 | pot_y | Integer | NOT NULL |
| 潜力-敏 | pot_m | Integer | NOT NULL |

### 3.3.8 秘笈使用表

| 属性 | 字段名 | 类型 | 约束 |
| --- | --- | --- | --- |
| 角色号 | role_id | Integer | PK/FK/NOT NULL |
| 秘笈号 | mj_id | Integer | PK/FK/NOT NULL |
| 等级 | level | Integer | NOT NULL |
| 阶数 | rank | Integer | NOT NULL |

### 3.3.9 自创武学表

| 属性 | 字段名 | 类型 | 约束 |
| --- | --- | --- | --- |
| 角色号 | role_id | Integer | PK/FK/NOT NULL |
| 自创武学名 | self_name | Varchar2 | |
| 心诀号 1 | first_xj_id | Integer | FK |
| 心诀号 2 | second_xj_id | Integer | FK |

### 3.3.10 装备穿戴表

| 属性 | 字段名 | 类型 | 约束 |
| --- | --- | --- | --- |
| 穿戴号 | wear_id | Integer | PK / NOT NULL |
| 角色号 | role_id | Integer | FK / NOT NULL |
| 装备号 | equ_id | Integer | FK / NOT NULL |
| 装备等级 | equ_level | Integer | |
| 强化等级 | stren_level | Integer | |

#### 3.3.11 装备洗练表

| 属性 | 字段名 | 类型 | 约束 |
| --- | --- | --- | --- |
| 洗练序号 | xl_id | Integer | PK / NOT NULL |
| 穿戴号 | wear_id | Integer | FK / NOT NULL |
| 洗练属性号 | xl_prop_id | Integer | FK / NOT NULL |
| 百分比 | percent | Float | NOT NULL |

# 4 物理设计

## 4.1 概述

在逻辑设计中，已对项目的后台存储部分进行了规范化的数据库设计。在逻辑设计的基础上，需要根据项目的功能，确定数据库中某些字段由于功能而需要的约束。同时，在实现前端功能与后端数据库的交互时，有一些数据会经常被访问，因此需要在这些数据建立索引。

由于在对操作的约束方面，前端的检查比后端更方便实现、维护和记录，因此这里的物理设计暂时不考虑一些复杂约束和触发器、函数等一系列过程类功能的实现，只对索引部分进行设计。

在项目初期，功能部分还需要进一步细化精化，所以这里只给出索引部分的设计框架和几个简单功能的索引设计，后续有可能再进行补充。

由于索引在设计时需要考虑多方面问题，例如对于某些字段是否建立索引、是否是唯一性索引、应该如何尽量设计较合适的索引类型等等。

## 4.2 功能描述

### 4.2.1 角色登录

应用中包含保存当前设置数据，并后续通过角色以及密码（可选）进行访问的操作，因此，需要有角色登录功能。在进行角色登录时，需要按照帐号来进行匹配，如果当前角色设置了密码，还需要将输入的密码与当初建立角色时的密码进行比较。

由于在设置用户帐号时不允许重复，因此对帐号的索引可以选择唯一性索引。根据约束，账号的长度是有限制的，因此不必使用前缀。综上考虑，可以使用按字符排序的方式建立唯一性索引进行查找。

### 4.2.2 静态信息查询

根据功能需求，用户在完成角色登录之后，需要调整自己的角色信息。调整信息时需要按不同字段来搜索秘笈表、心诀表、特技表的内容。这些表的访问是非常频繁的。

但是这样的表实际上数据量很小，因为它是“逻辑上”可数的。说得形象一点，就好比数据库版本种类。虽然它理论上也是可以无限制的，但是事实上这类表“数据量少”的特征非常明显。由于在各种字段上建立索引需要占用额外的空间，因此并不是索引越多越好。同时，数据量少的情况下，遍历索引依然效率不低。

因此对于这样的功能，采取的策略就是不建索引。

### 4.2.3 状态查询

这样的查询是紧接用户登录之后的，需要将与用户相关的一些内容从后端提取到前端，且是自动完成的。

由于在状态查询时，可以根据角色信息表中的角色号（对用户不可见）来联系角色，即使用 Int 值代替字符串，所以效率更高且能够更自由的选择索引方式。

这里我们假定用户很多，那么显然状态查询时数据量也是很大的，因此需要建立索引；且角色号唯一，可以选择唯一性索引方式。使用整数进行查询时，为了效率的提高可以使用哈希形式，它能够更快的索引想要查询的位置。

### 4.2.4 角色筛选

角色筛选和上述三种查询都不同，上述三种查询都是 equal query，而在进行角色筛选时，需要的是 range query，并且角色筛选可以使用多个字段进行，且这些字段可能甚至不在同一个表中，因此在执行这部分功能时，需要仔细设计索引形式。

角色筛选和状态查询都是以用户数量来评估数据规模，但显然角色筛选比用户登录连接的状态查询频次要更多。在对 range 进行查询时，首先哈希形式是不适合的。同时，由于需要多字段进行查询，但对数据字段进行排序时选用角色号作为主索引，所以，这些字段只能以 secondary 形式出现。

## 4.3 索引设计

在功能描述部分，已经对各种功能可能用到的索引进行了分析，因此这里只给出最后要建立索引的结果。

注意由于只是初步设计报告，所以这里的索引设计可能后续还需要经过大幅度的修改。这里将列出需要建立索引的表、建立索引的表的字段以及它们的查询类型和索引类型。当然在具体实现时，也许索引类型是由软件自行分配，但这里还是要考虑得更周到一些。如果需要优化则按照自己的设计进行优化。

| 表名 | 字段名 | 查询类型 | 索引类型 |
| --- | --- | --- | --- |
| 用户信息表 | 帐号 | equal | 排序-二分-唯一 |
| 角色状态表 | 角色号 | equal | 主索引 |
|  | 潜力-体 | range | Secondary |
|  | 潜力-劲 | range | Secondary |
|  | 潜力-气 | range | Secondary |
|  | 潜力-御 | range | Secondary |
|  | 潜力-敏 | range | Secondary |