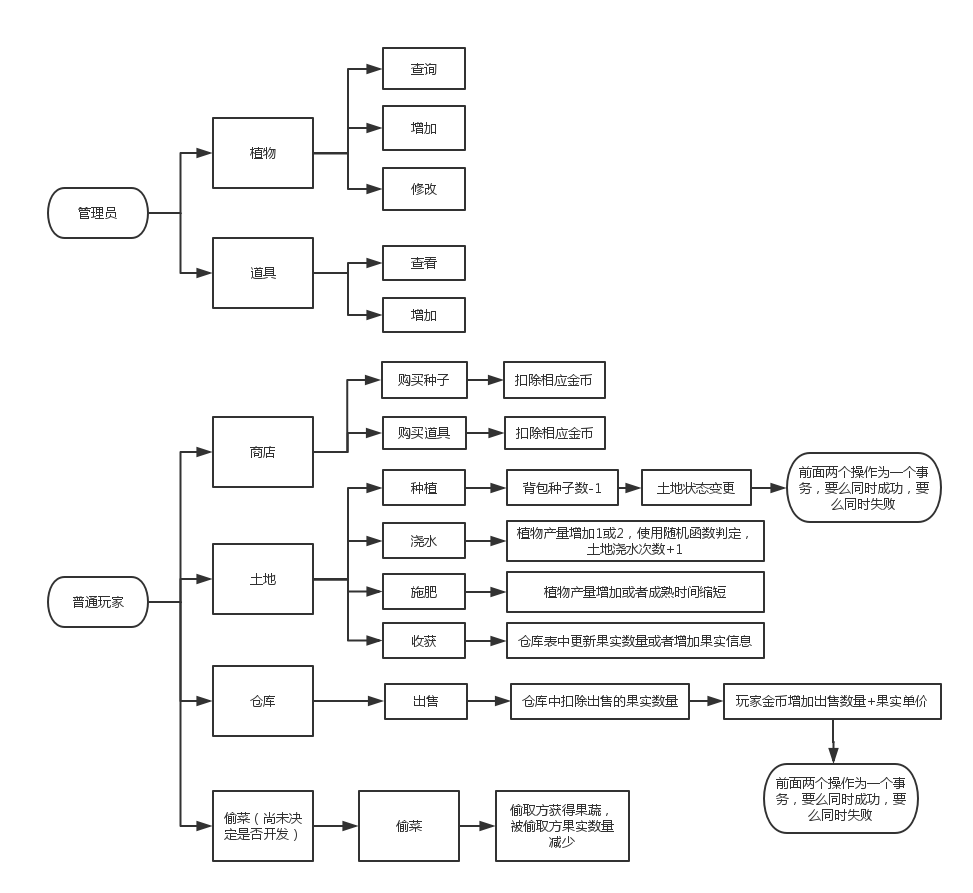
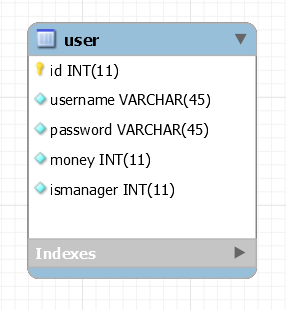
开心农场设计报告

——王恺鹏，张志通，吴家昊

1. 首先我们明确了一下至少要完成的大致功能，画出了流程图，后面有具体的操作分析和数据库分析。
2. 开发环境

小组讨论决定使用Java语言进行开发，使用NetBeans进行图形化桌面应用程序的开发。为了程序和项目的结构性和开发效率，决定引入hibernate框架，并对常用的操作进行封装，如加密操作，数据库CURD操作，字符编码操作。

1. 然后我们明确了一下数据库的设计，一共7个表。具体介绍如下，里面包括了一些实现的逻辑和具体的操作如何执行。



1. user表（用户表）：

字段有id、username、password、money、ismanager

数据类型为int、string、string、int、int

具体含义为唯一标识自动增量、用户名、用户密码、玩家的金币数、是否为管理员

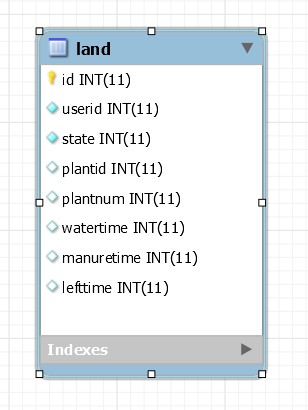
相关的业务逻辑为：

(1) 用户注册成功后将用户密码进行hash(如md5)后存入数据库

(2) 用户登录时验证获取登录表单中的用户名和密码和身份选择(管理员和普通玩家)，将密码进行MD5加密，然后通过数据库查询语句查询是否存在对应的用户名和密码，如果存在则登录成功。

(注：普通玩家的ismanager值为1，管理员的ismanager值为0)

(3) 用户购买商品时获取玩家金币数，如果数量足够则进行购买并扣取相应金币。



1. land表（土地表）:

字段有id、userid、state、plantid、fruitnum、watertime、manuretime、leftttime

数据类型为int、int、int、int、int、int、int、int

后面6个值默认为0

具体含义为唯一标识自动增量、用户id、土地状态(空间、植物正常、植物干涸、植物成熟)、植物id、当前情况果实数量、浇水次数(浇水次数上限为3次，土地收获后置0，浇水)、施肥次数(施肥次数上限为3次，土地收获后置0)、收获剩余时间。

相关的业务逻辑为：

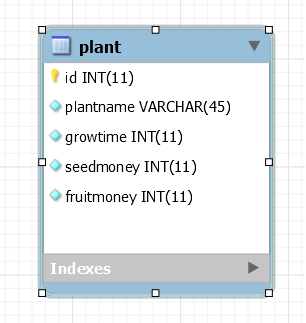
这个表的作用和功能最复杂

1. 当一个用户注册成功后，在这个表插入三条数据

为id、userid、0、0、0、0、0、0

标识初始状态下默认给玩家三块可用土地，state为0标识土地为空闲状态。

1. 当用户开垦土地时，若金币充足，则扣除相应金币，在land表中加入一条新的数据id、userid、0、0、0、0、0，即和用户注册时默认开垦土地的形式相同。
2. 当用户种植植物时，首先判断此土地的state，如果为0才能进行种植。种植成功后相应条目更改为id、userid、1(植物正常) 、植物全程正常状态下的收入、0(初始浇水次数为0)、0(初始施肥次数为0)、植物收获所需时间(因为刚开始还没有生长).
3. 因为服务器没有长期开启的原因，植物的生长我们设置为断点续传的方式，即玩家在线时植物才能生长，玩家下线时数据库记录植物生长剩余时间，再次上线后继续计时。
4. 当进行浇水时，首先判断watertime的值，如果<3就可以进行浇水，浇水成功后值+1，fruitnum增加相应的收益。如果=3，则不能进行浇水。浇水可以增加植物的产量，或者移除干涸debuff。
5. 当进行施肥时，首先判断maturetime的值，如果<3就可以进行施肥。然后判断使用的是何种肥料，如果是降低成熟时间的肥料，成功后maturetime值+1，lefttime扣取肥料降低的成熟时间，如果剩余时间<缩短的时间，则植物直接变成成熟状态。如果是增大植物产量的肥料，成功后maturetime值+1，fruitnum增加相应的收益。如果=3，则不能进行施肥。施肥可以增加植物的产量，或者是缩短植物成熟时间。
6. 用户收获植物时，土地还原为默认初始状态。



3. plant表（植物表）：

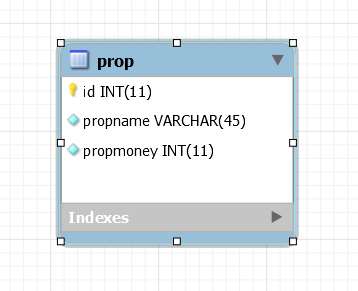
字段有id、plantname、growtime、seedmoney、fruitmoney

数据类型为int、string、int、int、int

具体含义为唯一标识自动增量、植物名称、植物生长所需时间、植物种子价格、植物果实价格。

相应的业务逻辑为：

1. 管理员可以在后台进行植物信息的管理，包括删除植物，修改植物信息和新增植物，为了不影响用户体验，决定去掉删除植物的功能。
2. 商店中显示所有的植物信息，玩家可以在商店中进行种子的购买，购买种子后可以进行种植
3. 种植后土地表的行数据进行相应的更改，plantid设定为这个植物的id，fruitnum设定为初始固定值10，但是后续会根据道具的使用情况进行变更，state设定为1，lefttime设定为这个植物的growtime。



4. prop表（道具表）：

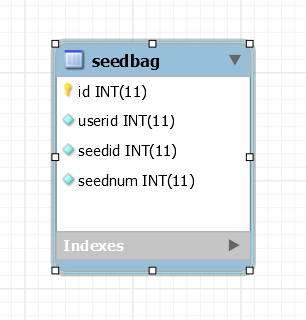
字段有id、propname、propmoney

数据类型威威int、string、int

具体含义为唯一标识自动增量、道具名称、道具价格。

相应的业务逻辑为：

1. 管理员可以在后台进行道具信息的管理，为了不影响用户体验，决定只制作增加道具这一功能。
2. 商店中显示所有的道具信息，玩家可以在商店中进行道具的购买，购买道具后可以进行使用。
3. 在使用道具的时候，程序通过分析道具名称的字符串来唯一确定道具的使用用途和作用。程序通过关键字的算法分析来定位每一种具体道具的作用，确保其功能正确的体现。



5. seedbag表（种子背包表）：

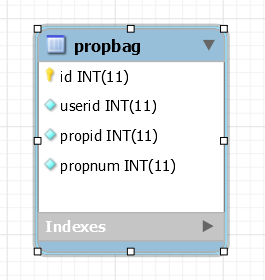
字段有id、userid、seedid、seednum

数据类型为int、int、int、int

具体含义为唯一标识自动增量、用户id、种子id、种子数量。

相应的业务逻辑为

1. 当用户从商店购买种子成功时，我们先对这个seedbag表进行查询(通过userid、seedid属性)如果有相应的数据，则我们执行更新操作，首先提取seednum的值，然后加上用户的购买量，再把得到的总数值存入数据库中完成更新操作。如果没有相应的数据，则我们执行增加操作，将userid、seedid、用户的购买量存入数据库，完成购买操作。
2. 当用户使用种子进行种植时，我们查询获取相应的数据，先查找数据库中是否存在对应此userid和seedid的值，如果不存在，提示用户并不存在想要种植的种子，是否跳转至商店进行种子的购买。如果存在，先判断seednum的数值是否为1，如果为1，将此数据行删除，执行删除操作，如果并不为1，将seednum的数据进行-1操作，执行更新操作。



6. propbag表（道具背包表）：

这个表和seedbag表类似

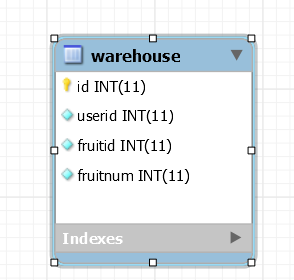
字段有id、userid、propid、propnum

数据类型为int、int、int、int

具体含义为唯一标识自动增量、用户id、道具id、道具数量。

相应的业务逻辑为

1. 当用户从商店购买道具成功时，我们先对这个propbag表进行查询(通过userid、propid属性)如果有相应的数据，则我们执行更新操作，首先提取propnum的值，然后加上用户的购买量，再把得到的总数值存入数据库中完成更新操作。如果没有相应的数据，则我们执行增加操作，将userid、propid、用户的购买量存入数据库，完成购买操作。
2. 当用户使用道具时，我们查询获取相应的数据，先查找数据库中是否存在对应此userid和propid的值，如果不存在，提示用户并不存在想要使用的道具，是否跳转至商店进行种子的购买。如果存在，先判断propnum的数值是否为1，如果为1，将此数据行删除，执行删除操作，如果并不为1，将propnum的数据进行-1操作，执行更新操作。



7. warehouse表（仓库表）：

字段有id、userid、fruitid、fruitnum

数据类型为int、int、int、int

具体含义为唯一标识自动增量、用户id、果实id（植物id）、果实数量

相应的业务逻辑为

1. 当用户收获时，首先根据userid和fruitid到表中查询是否有相应的信息，如果有，则执行更新操作，把fruitnum加上land表中对应的fruitnum的数值。如果没有相应的信息，则执行增加操作，正在表中加入一条相应的数据。
2. 当用户出售果实时，先判断用户出售的数量是否等于数据库中的现有值，如果是，则开启事务管理，执行删除操作删除这条数据，同时用户金币更新为当前量+果实单价\*果实数量。如果用户出售的数量不等于数据中的现有植，则开启事务管理，执行更新操作，fruitnum减去用户出售的数量，同时用户金币更新为当前量+果实单价\*果实数量。

注意这里务必要使用事务的机制，即果实数量减少的操作和用户金币增加的操作要么同时成功，要么同时失败。

这里说明一下没有使用外键的原因，在关系型数据库中，一般表之间会使用外键关联的形式实现表与表之间的关联来进行级联删除和级联更新。比如删除用户后，用户的所有相应信息都进行删除。但是我们没有设置删除用户的功能。同时为了用户体验也没有设置删除植物种类和道具的操作（删除很方便实现，比更新容易的多）。所以我们小组认为没有必要设置外键。

后续根据开发难度的简单和容易，我们会决定是否开发偷菜功能，进行好友之间的互动。