WEBGAME 和平世界

设计文档精要

负责人： 梁爽
联系方式： LiangShuangDe@163.com
http://www.wizicer.com
QQ: 9721316

目录

和平世界游戏系统

1. 系统定义

1.1. 网游市场

当前的游戏行业充斥着各种网络游戏，单机游戏的市场已经不大，并不是因为大家都不愿意玩单机的了，而是盗版太厉害，我国的生活水平尚未达到让大多数人能从容的购买单机游戏的地步，所以走网络游戏路线既是必然，也是不可改变的未来趋势。

当前网络游戏横行，出品的游戏可以说是层出不穷，数不胜数，同样，每天面临无法继续经营而必须关门的游戏公司也是相当多的，在网络游戏市场，现在呈现出一种十分混乱的局面，也是因此在这个行业里面充满了机遇和挑战。

从九十年代中期由一部分程序纯属爱好而做出来的可以在浏览器上玩的游戏，不过可惜当时的浏览器技术根本不足以方便快捷的进行游戏，而且那时候人们对网络的认识也不够深刻，制作出来的游戏一般都不能让太多的人一同进行，很多时候看上去就是一个大家都可以在网上玩的单机游戏。

随着时代的进步，浏览器及其表现技术的发展，即 Ajax、Flash 等技术的发展，现在在浏览器上能表现出更加丰富的视觉效果和用户体验，并且服务器性能的大幅度提高，使这种类型的游戏有了发展的空间。

1.1.1. Webgame 市场

今年开始，不少游戏公司的目光都不约而同的朝向了 Webgame，发现这个在网络游戏时代留下来的大蛋糕，于是开始争先恐后的上前瓜分，不过因为时间还很短，现在出品的 Webgame 还并不是很多，大约在今年的六月份左右，将会有相当大一批 Webgame 出炉来分割这块蛋糕，相比传统的网络游戏来说，这个蛋糕要小得多，更容易就被市场满足了，所以现在必须抓紧这个时机迅速出击才能获得胜利。

对广大的游戏制作公司来说，Webgame 的制作成本是在游戏界里面最入门级的，所以大公司只需要分出一部分的人力和财力就可以轻松胜任，小公司也可以借此作为公司的发展契机，所以现在市场的现实状况也是很紧迫的，必须“快、狠、准”的抓住市场、抓牢市场。

1.1.2. 盈利模式

由于本游戏是网页游戏，所以在游戏的开发过程中所用到的成本相对来说是比较低的。不过既然开发团队有成本支出，考虑盈利也就是理所当然的了。本游戏的盈利方式主要途经有三种：（以下在游戏中进行交易的“金币”是玩家用人民币向系统通过一定的途经购买获得的，后期游戏还会开通其他途径来让玩家轻松获得金币。这也是我们盈利的基础。）

当然，玩家在不用金币的情况下也能很好的进行游戏，与其他玩家进行互动，但是有些功能将被制约，这既体现了免费的网络游戏肯接受低端玩家，同时也能给高端玩家带去更加不一样的享受。

2. 需求分析

2.1. 可行性分析

2.1.1. 技术可行性

随着时代的进步，浏览器及其表现技术的发展，即 Ajax、Flash 等技术的发展，现在在浏览器上能表现出更加丰富的视觉效果和用户体验，并且服务器性能的大幅度提高，使这种类型的游戏有了发展的空间。所以本系统在技术上是可行的。

2.1.2. 经济可行性

由于本游戏是网页游戏，所以在游戏的开发过程中所用到的成本相对来说是比较低的。本游戏的盈利方式主要途经有两种：（以下在游戏中进行交易的“金币”是玩家用人民币向系统通过一定的途经购买获得的，后期游戏还会开通其他途径来让玩家轻松获得金币。这也是我们盈利的基础。）1、卖道具提高用户体验。2、提供用户便捷的游戏方式。当然，玩家在不用金币的情况下也能很好的进行游戏，与其他玩家进行互动，但是有些功能将被制约，这既体现了免费的网络游戏肯接受低端玩家，同时也能给高端玩家带去更加不一样的享受。所以本系统在经济上是可行的。

2.1.3. 操作可行性

本系统支持多种操作系统、网络环境和硬件平台，完全实现其开放性。服务器端支持主流的操作系统 Windows；客户端运行在浏览器中，所以只要有连上网的浏览器，就能连上系统进行操作。所以本系统在操作上是可行的。

2.1.4. 法律可行性

本系统属于 WEBGAME，按照现在的管理规范，本系统属于网站系统，系统的开发过程完全按照相关互联网法律要求进行。所以本系统在法律上是可行的。

2.2. 系统总体分析

由于考虑到本系统的用户量，本系统使用三层分布式构架，数据库和逻辑服务器使用 SQL Server 的管道连接，逻辑服务器和表现层服务器使用微软分布式构架 Remoting 构建，使用 tcp 通道连接。

三个服务器均可使用 Windows+.Net2.0 或是 Linux+Mono 方式组建。

2.3. 系统功能

2.3.1. 城堡建设

玩家可以根据城堡中的现有情况以及用户已经学习的技术情况，显示城堡中可以建设/升级的建筑物，列出城堡中已经拥有的建筑物，选择可以建设/升级的建筑物进行建设/升级操作。

2.3.2. 技术研究

玩家可以根据用户的学习状况以及城堡中的建筑情况，进行技术的查看以及技术学习/升级操作。

2.3.3. 资源采集

在界面上显示当前操作城堡周围的可采集资源，并可设置各个资源的采集效率。

2.3.4. 效率设置

显示当前城堡中的所有建筑物及其当前生产效率，并可以进行设置和修改。

2.3.5. 贸易

显示当前城堡周围一定范围内的贸易中心，及其里面的交易项目，用户可以提交自己的贸易要求。另外，用户可以与系统进行直接交易。

2.3.6. 联盟

显示当前用户已经加入联盟的信息，有四个页面，分别是基本信息、成员管理、联盟消息、联盟管理，基本信息中显示当前联盟的基本信息及统计信息；根据权限在成员管理中可以进行成员的增减；在联盟消息中可以发送和查看联盟消息；在联盟管理可以进行联盟称号、

联盟外交的管理。

2.3.7. 英雄

显示当前城堡周围在野的英雄，用户可以选择雇佣。显示当前城堡内的英雄，并可以选择解雇。英雄可以进行升级、学习新技能操作。

2.3.8. 地图

显示大地图，其中按当前城堡为中心显示地图。可以进行移动，可以输入具体坐标，进行快速移动。

2.3.9. 配方生产

可以研制配方，选择一些材料进行配方测试，成功的话则将该配方存入用户的配方列表。在该页面用户可以选择自己已经研制出来的配方进行产品生产。

2.3.10. 军队

可以进行军队的集结与解散。并可以进行军队的组织和修改。军队可以含有多个队伍，每个队伍含有一个兵种。

2.3.11. 用户管理

进行用户的基本信息管理，可以修改用户的密码，可以修改城堡的名称。

2.3.12. 站内信息

可以查看当前用户的站内信息，并可以向其他用户发送站内信息。

2.4. 游戏特色

2.4.1. 全新的游戏角色

本游戏打破常规，并不沿袭前人用“人”作为游戏的角色，而是选用一种动物进行拟人化后来作为本游戏的角色。本游戏之所以要选择“蚂蚁”作为游戏的角色，是应为蚂蚁有一些优良的品质：他们十分勤劳，每天都为自己的群落辛勤劳动；他们分工明细，都兢兢业业干好自己份内的工作；他们忠诚，从出生到死，他们只会忠于自己的群落，绝对不会背叛；他们勇敢，面对来犯之敌，他们奋起反抗，毫不吝啬自己的生命，因而战斗力超强；他们团结互助，面对比自己庞大许多的猎物的时候，他们终能靠团队战胜。这些优良的品质可以说基本都是现代社会的人所缺少的，所以本游戏才选择用蚂蚁作为游戏的角色。相信当它们粉墨登场的时候，必会给玩家带来一种全新的感觉。

2.4.2. 英雄等级与技能

英雄从招募到他开始就具有一定的初始等级，其初始等级的高低由招募到此英雄时神殿的等级高低所决定：神殿等级越高，招募到英雄的等级也就越高。在此基础上，通过英雄带兵出去征战获得经验（不管胜利还是失败都会获得经验，只是胜利时获得的经验比失败时获得的经验要多得多）。经验累加都一定的数量就会提升英雄的等级，然后又为下一等级的提升积累经验。

作为游戏一大特色，和其他的网页游戏相比，该游戏的英雄可以通过各种途经习得能左右战争胜负的技能，可以派遣到友盟的学院中进行进修，可以向其他武将进行学习，甚至可以派遣至敌方阵营进行偷偷学习（在下面的“英雄的相关设定”里面有详细的说明）。具体技能参考实时更新的英雄技能表。

2.4.3. 传承战略游戏的精华

传统战略游戏中的精华被完完整整的传承下来，资源采集、城堡建设、战争等等在战略游戏中耳熟能详的词语将会贯穿整个游戏，让你进行游戏的同时也让自己的头脑进行了一次思考的风暴，更让你充满战略的头脑有了用武之地，驰骋沙场，获得属于自己的更高的荣誉。

加入网络过后，联盟、外交等功能也会使整个游戏更加有乐趣，玩家的互动共同创造一部属于玩家自己编写的异世界历史。

2.4.4. 丰富的资源类型

游戏中设计了几十种基本资源类型，并由这些资源通过组合合成上百种产品，丰富的资源类型大大的丰富了游戏的内容。

每位玩家所拥有的城堡所在的地域能够生产的资源类型可能只有少少的几种，为了能制造出高级的产品，或者自己去占领更多的土地，或者自己去进行交易。这样的设定能大大的增强玩家之间的互动。

随着时间的推进，游戏里面还会根据需求或者“时代”的要求，添加新的资源产品。

2.4.5. 突破传统的自定义兵种

游戏中的兵种不再有限制，玩家可以自定义的为自己的士兵穿戴上不同的武器装备，当然不同的武器装备配置方法必然会得到很多意想不到的效果。使游戏的趣味性大大的提高。

玩家在游戏过程中可以不断的实验武器装备的配置方法，得出来的结果或者保密，或者传授与他人，更可以当作交易的商品与他人交易。

2.4.6. 全新的贸易设定

只有比较有实力的玩家能够建造具有竞争力的市场，所有玩家要进行贸易，除了可以自己联系其他玩家之外，一般就只能通过附近的市场进行贸易了，市场只提供信息，当玩家确认交易后，需要自己派出相应的队伍进行贸易或是保护贸易，市场的建造者将会在这场贸易中收取其自己制定的贸易费用。

当需要运送比较高级的货物并且对自己的部队不是很报信心的时候，可以向市场建造者申请“保镖”运送。

2.4.7. 创新的时代和国家设定

由最高级的玩家引领，通过大量的研究，当完成了这个时代所拥有的所有的技术，整个时代将会向前推进，进入到下一个时代，所有低级用户将会自动获得这个时代的“常识”技术，前一时代的技术的研究速度将会大幅度提高。

时代的前进，即代表着研究技术的提高，虽然没有进入下一个时代也能研究下一个时代的技术，但是研究成果相对来说也会比较收效甚微。

几十个甚至上百个玩家可以组成一个联盟，几百个甚至几千个联盟就能组成一个国家，一个国家里面的“时代”是共享的。

就如同一个联盟能修建联盟城堡一样，一个国家也能修建一些特殊建筑物，如伟人雕像、国家宫殿等等。当然就会有很多普通城堡没有的功能。提高了游戏的趣味性。

2.4.8. 全新的战场模式

战场的设定使同时参与战斗的玩家可以增加到无限，战斗也不会是瞬间完成，会是一个过程，在这个过程中，你也可以监视你部队的战斗情况，并做出适当的命令。

战场上有追击设定，当大部队逃走时，如果脚力没有追逐的部队强，很有可能就会被追上并遭到围剿。

当有部队进入自己城堡的视线距离时，会立即对玩家进行提示，以做出相应的应对措施。

2.4.9. 多途径的游戏方式

我们除了提供传统的网页游戏方式之外，我们也提供 PC 客户端和手机 WAP 方式，使玩家能够广泛的加入到游戏中来，并且使有条件使用 PC 客户端的用户有了更加方便的游戏方式。

我们的 PC 客户端会比较注重开发玩家们之间的互动，所以可以更好的是玩家融入到游戏中去，游戏的同时也不忽略与周围玩家的交流。

2.4.10. 可自定义的地域命名

每位玩家都可以为大地图中的任意一个地方命名，命名后的一大特色就是在记录历史的时候将会写清楚具体的战斗地点名称，并可方便玩家进行称呼。

当多个玩家对同一块地方进行命名时，将会激活投票系统，由大家的投票结果决定这个地方的名称，毕竟一个地方只能有一个名称。游戏也会限制在一定范围内是不允许重名的。

2.4.11. 能够记录的历史设定

所有地图块及每个用户自己都会有历史记录设定，每当做了比较重大的事情时就会记录下他自己的历史，当玩家偶尔看起来的时候，便能找到自己在这个游戏中的轨迹，回味游戏中的点点滴滴，有自己光辉的一面：某某时间，和某某玩家进行大战于某某地方，损失少量士兵的情况下几乎全歼对方，获得不少的经验，意外的是还获得一件宝物，堪称大获全胜。也有自己惨淡的一面：游戏初期，什么都比不上别人，自己就是一只任人宰割的小羊，经常被人捋夺，让自己十分的无奈。这些都可以根据玩家自己的设定记录下来。

2.5. 服务器构架业务流程图

由于考虑到本系统的用户量，本系统使用三层分布式构架，数据库和逻辑服务器使用 SQLServer 的管道连接，逻辑服务器和表现层服务器使用微软分布式构架 Remoting 构建，使用 tcp 通道连接。

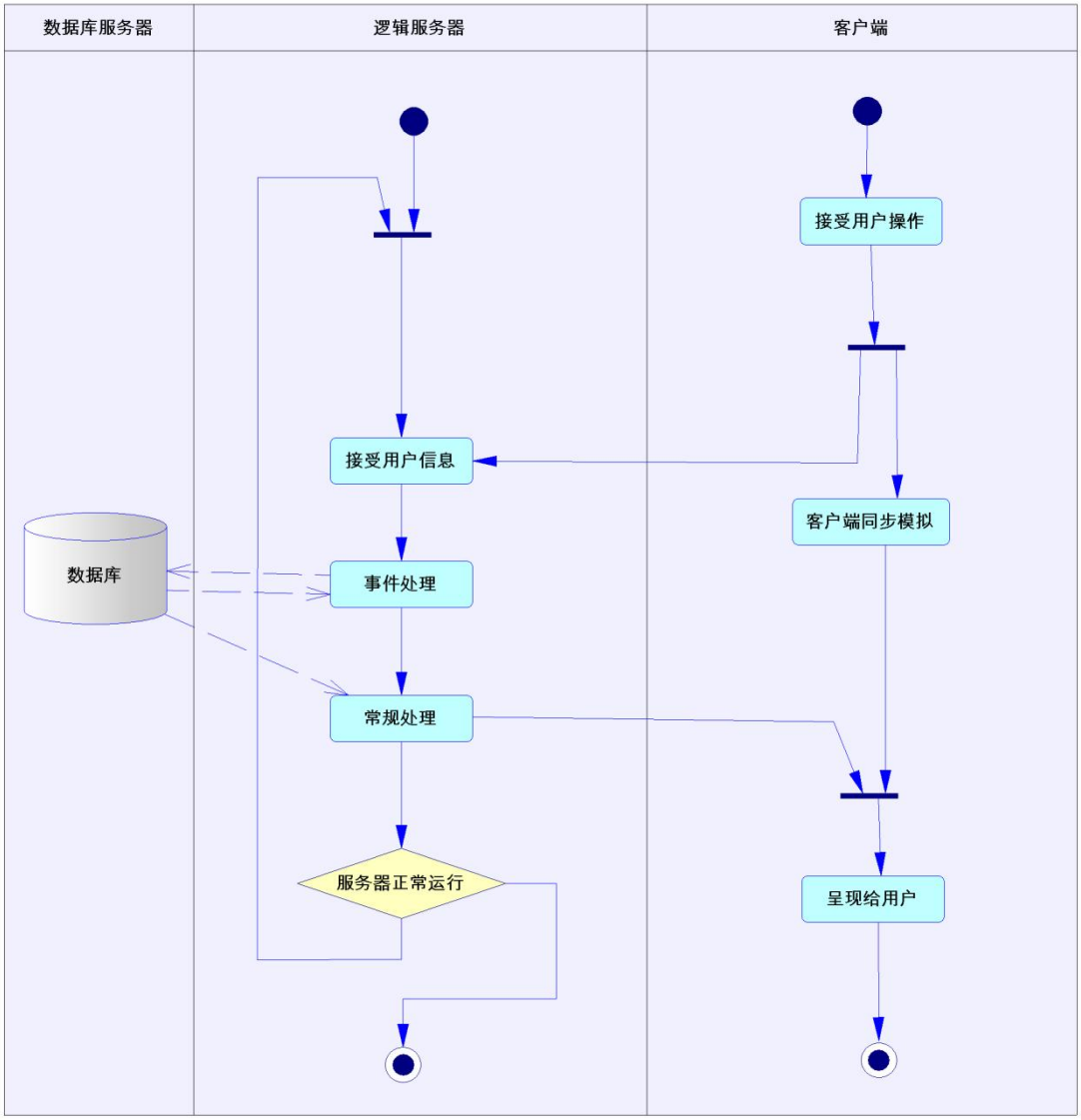


图 2.1 系统业务流程图

3. 系统设计

3.1.数据库设计

系统整体概念模型见附录一，物理模型见附录二。

3.1.1. 设计介绍

为了数据库设计的合理化、规范化，也为了更好的服务前台程序的控制，我们选择了 PowerDesigner 12.5 数据库设计工具。PowerDesigner 是 Sybase 推出的数据库设计工具。PowerDesigner 致力于采用基于 Entity-Relation 的数据模型，分别从概念数据模型

(Conceptual Data Model)和物理数据模型(Physical Data Model)两个层次对数据库进行设计。概念数据模型描述的是独立于数据库管理系统(DBMS)的实体定义和实体关系定义。物理数据模型是在概念数据模型的基础上针对目标数据库管理系统的具体化。

在数据库设计上，Third Normal Form（3NF）通常被认为在性能、扩展性和数据完整性方面达到了最好平衡。概括的说，遵守三范式（3NF）标准的数据库的表设计原则是：“One Fact in One Place”即某个表只包括其本身基本的属性，当不是它们本身所具有的属性时需进行分解，表之间的关系通过外键相连接。关于范式定义于下：

第一范式(first normal form，简称 1st NF)就是指在同一表中没有重复项出现，如果有则应将重复项去掉。

第二范式(second normal form，简称 2nd NF)是指每个表必须有一个(而且仅一个)数据元素为主关键字(primary key)，其它数据元素与主关键字一一对应。通常我们称这种关系为函数依赖(functional dependence)关系。即表中其它数据元素都依赖于主关键字，或称该数据元素唯一地被主关键字所标识。

第三范式(third normal form，简称 3rd NF)就是指表中的所有数据元素不但要能够唯一地被主关键字所标识，而且它们之间还必须相互独立，不存在其它的函数关系。也就是说对于一个满足了 2nd NF 的数据结构来说，表中有可能存在某些数据元素依赖于其它非关键字数据元素的现象，必须加以消除。

为防止数据库出现更新异常、插入异常、删除异常、数据冗余太大等现象，关系型数据库要尽量按关系规范化要求进行数据库设计。

在本系统的数据库设计过程中没有一味的坚持使用第三范式，为了加速运算和方便编程，我们将用户排名字段放在了用户表中，每过一段时间进行一次计算，减少数据库的不必要的负担，同时也使程序更加清晰，更容易实现和维护。

3.1.2. 联盟相关数据库设计

在联盟相关的数据表中实现了联盟内部消息发送和接收，联盟成员的分等级权限管理，联盟外交管理以及联盟中成员的管理。

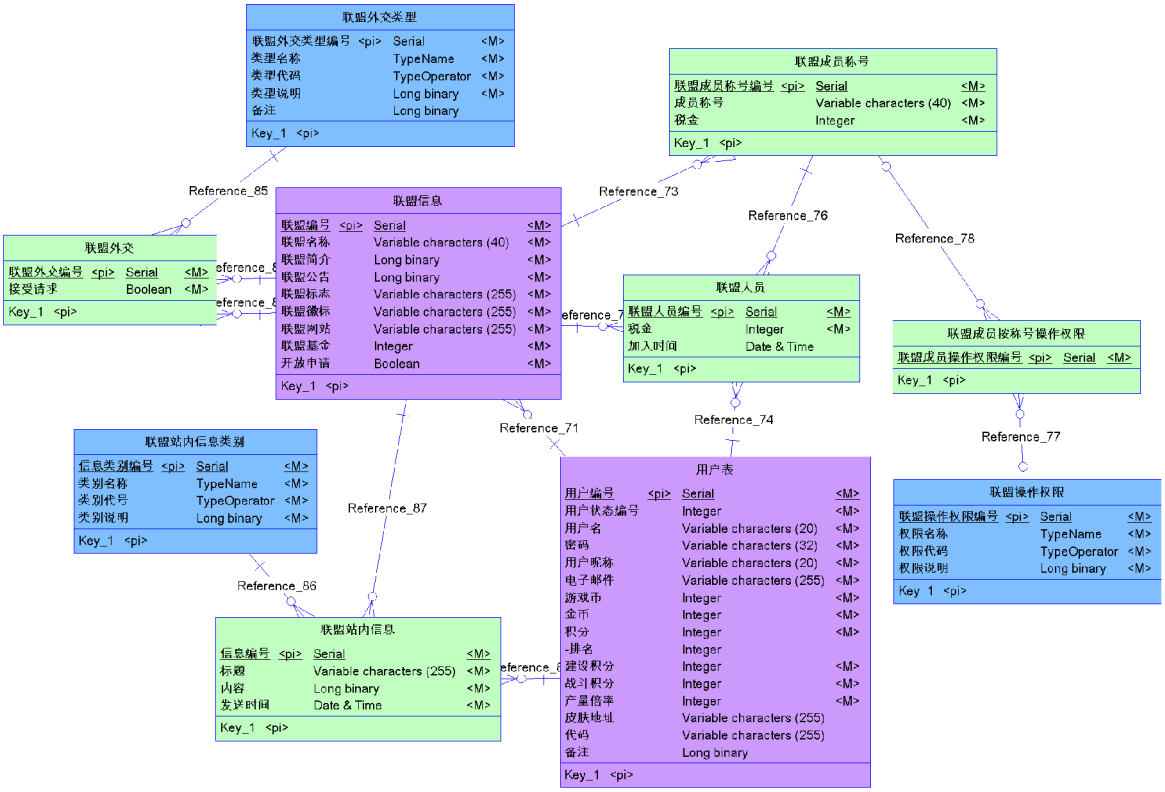


图 3.1 联盟相关概念模型

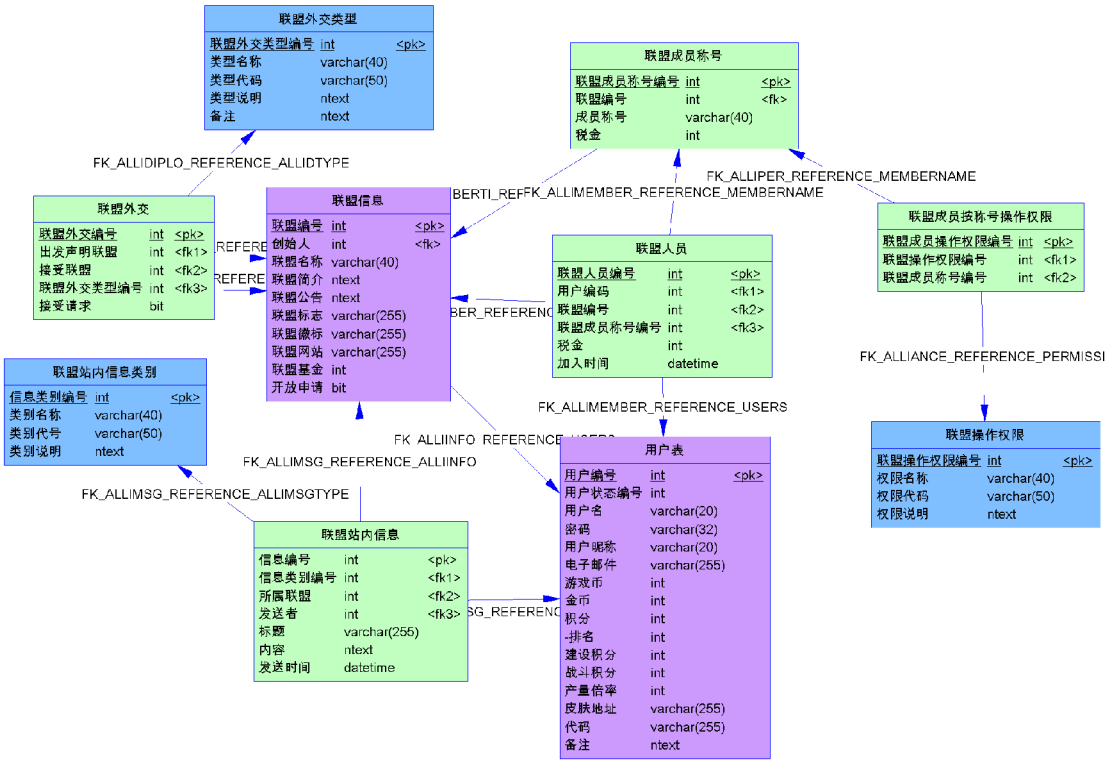


图 3.2 联盟相关物理模型

3.1.3. 战场相关数据库设计

战场相关的数据表实现了多个用户的军队在地图上某块区域上的战场打仗的过程，战场中支持多个用户的多支军队，并同时支持用户的英雄及技能，对战斗过程会进行记录以便将来用户可以查看战斗的整个过程，当战斗完毕时，会对相应表进行数据清理，在战场部队信息表和战场英雄信息表含有以下触发器：

```
CREATE TRIGGER tr_DeleteTroopInBattlefield
ON [TroopInBattlefield]
INSTEAD OF DELETE
AS
BEGIN
    UPDATE [UnitInBattlefield] SET [TroopInBattlefieldID] = NULL WHERE [TroopInBattlefieldID] IN
    (SELECT [TroopInBattlefieldID] FROM deleted)
    DELETE [TroopInBattlefield] WHERE [TroopInBattlefieldID] IN
    (SELECT [TroopInBattlefieldID] FROM deleted)
END

CREATE TRIGGER tr_DeleteHeroInBattlefield
ON [HeroInBattlefield]
INSTEAD OF DELETE
AS
BEGIN
    UPDATE [UnitInBattlefield] SET [HeroInBattlefieldID] = NULL WHERE [HeroInBattlefieldID] IN
    (SELECT [HeroInBattlefieldID] FROM deleted)
    DELETE [HeroInBattlefield] WHERE [HeroInBattlefieldID] IN
    (SELECT [HeroInBattlefieldID] FROM deleted)
END
```

使用以上两个触发器在清理战场中部队和英雄的时候，能够自动将对应用于记录战斗过程的表中信息脱开关联，这样使数据库更加健壮，并易于操作和编程。

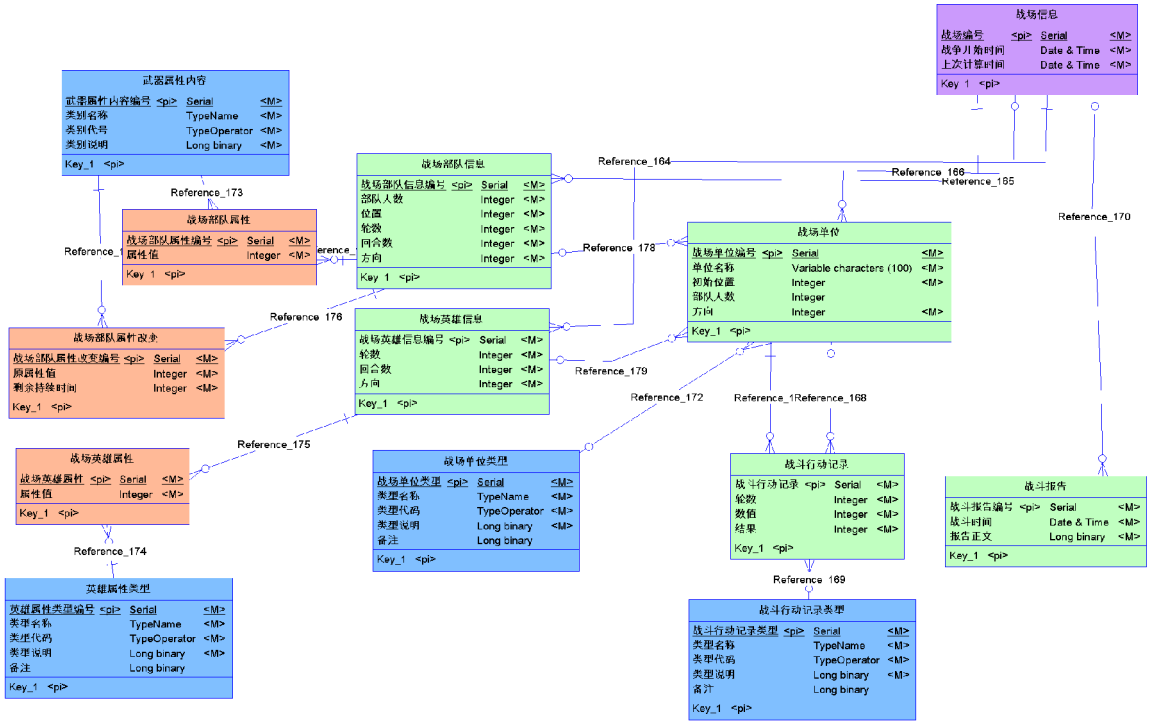


图 3.3 战场相关概念模型

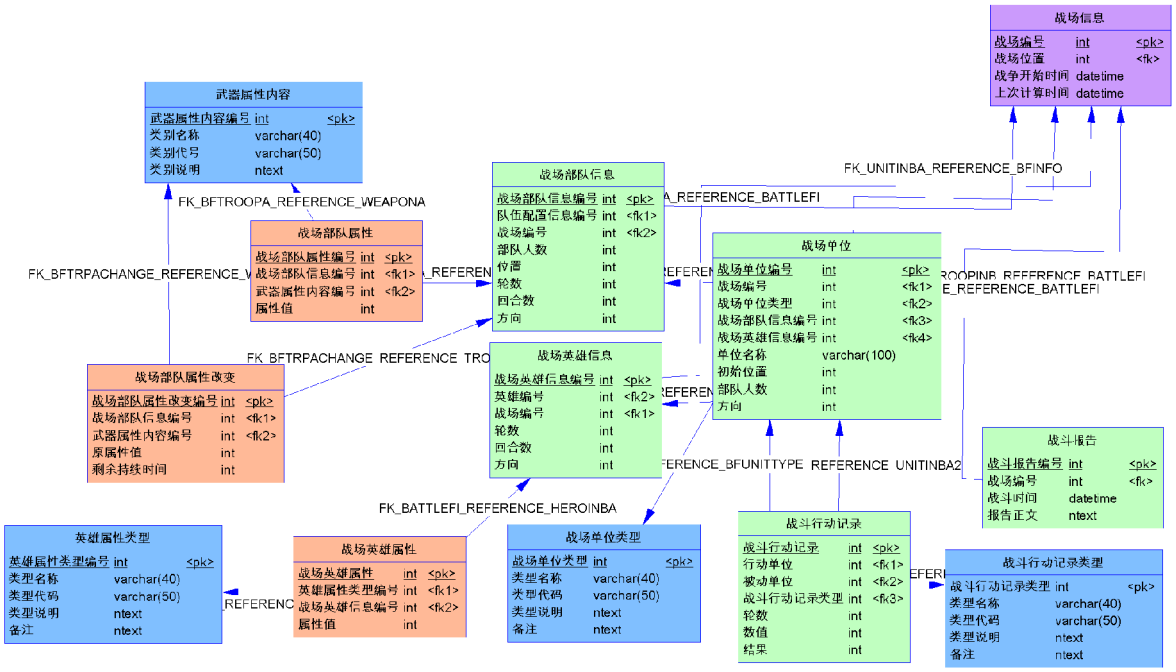


图 3.4 战场相关物理模型

3.2. 系统功能模块图

3.2.1. 城堡相关用例图

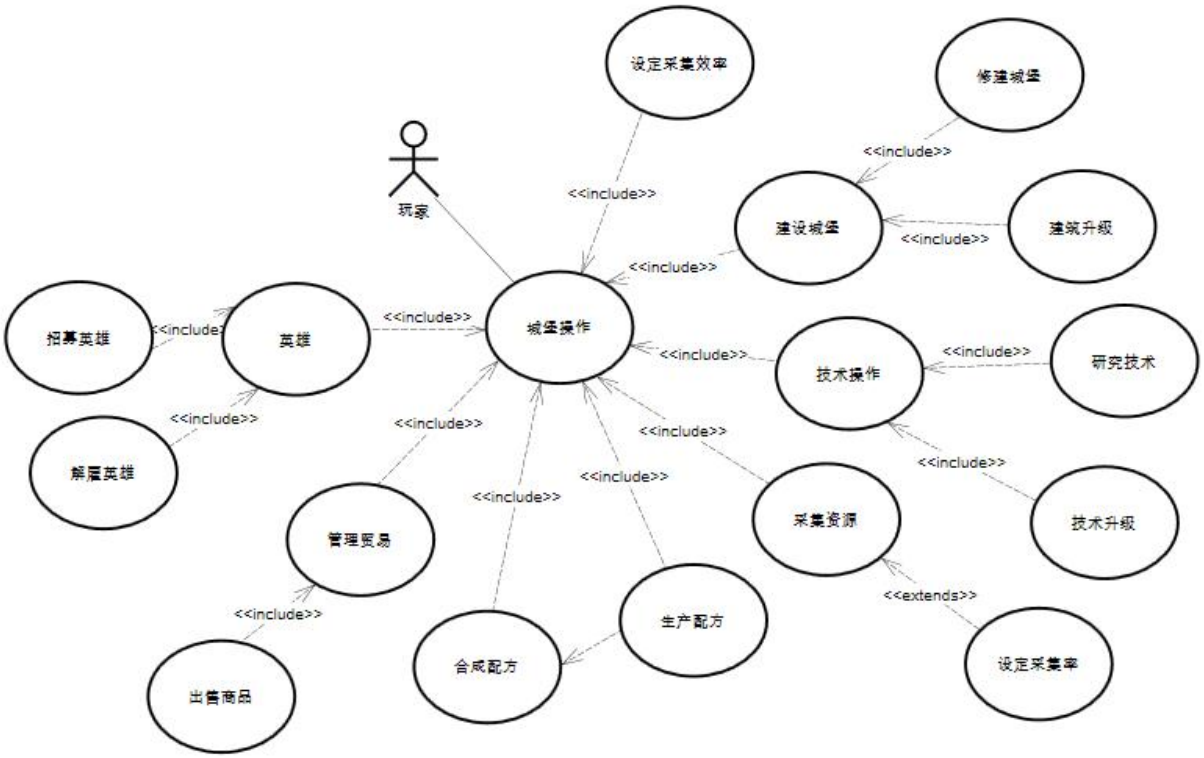


图 3.5 城堡相关用例图

用例名称：城堡操作

参与者：玩家

简要说明：

玩家可以建设城堡，升级建筑，资源采集，技术研究，技术升级，合成配方，生产配方，招募英雄，解雇英雄，出售物品。

基本事件流:

1．玩家鼠标点击"城堡菜单"。

2．系统出现一个页面，显示城堡的基本信息，资源信息，城堡建设信息。

3．玩家可以点击修建建筑物，升级建筑物，设定采集率，设定生产效率，升级技术。

3.2.2. 联盟相关用例图

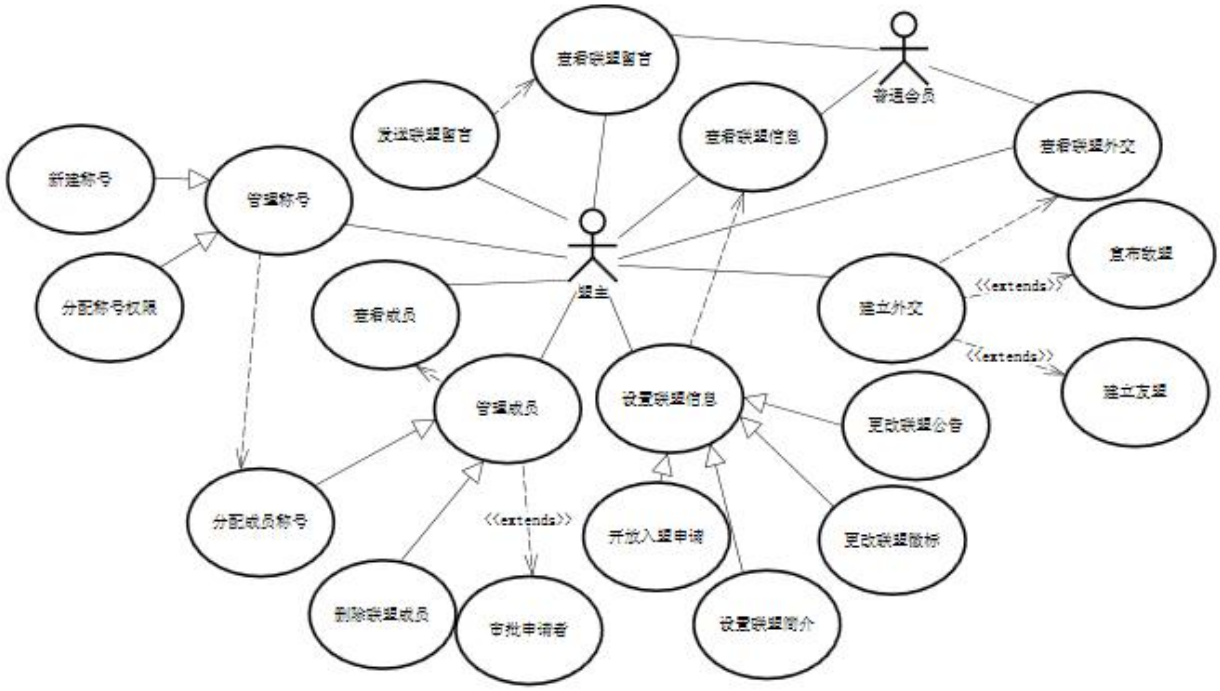


图 3.6 联盟相关用例图

用例名称：设置联盟信息

参与者：盟主

简要说明：

盟主负责联盟的基本信息维护,可以修改联盟简介、公告、徽标、开放申请

基本事件流:

1．盟主鼠标点击"修改联盟信息"

2．系统出现一个窗口，显示着原来的联盟简介、公告、徽标

3．负责人可以在窗口中修改公告、简介、徽标，也可以完全删除，重新写新的公告

4．负责人编辑完内容，按“修改”按钮，联盟基本信息就被修改

5．用例终止

3.2.3. 战场相关用例图

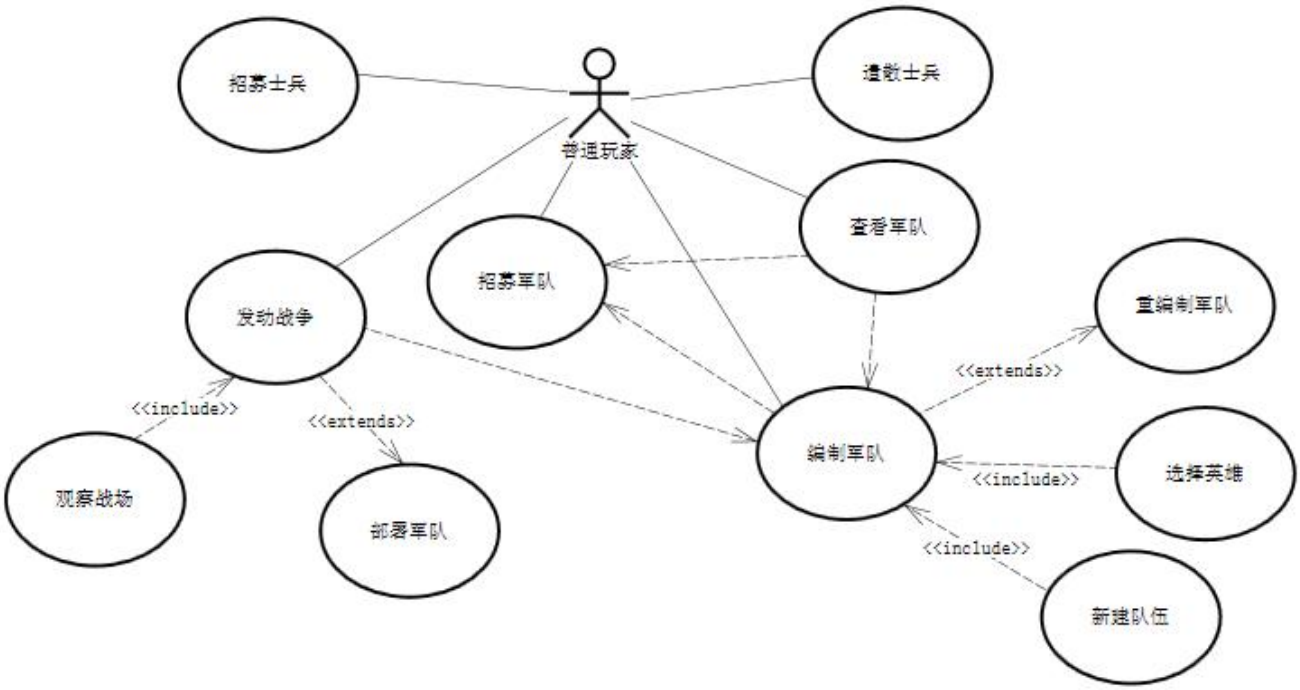


图 3.7 战场相关用例图

用例名称：招募军队

参与者：玩家

简要说明：

玩家可以招募士兵，编制军队，发动战争，遣散士兵

基本事件流:

1．玩家鼠标点击"战争"

2．系统出现一个页面，显示军队，部署，战争

3．玩家可以点击征兵，编制军队

4．给军队设置英雄，进行战争

4. 详细设计

4.1. 建设城堡

建设城堡是本系统中较为复杂的逻辑部分，同时为了层次分明使用了多个存储过程，以下为这其中最为核心的部分，用于返回用户该城堡可以建设/升级的建筑物项目，以及其相关基本信息。

一个建筑物可以被建筑的要求是，建设该建筑物的前提建筑物已经建立，而且当前建筑物并非为已经在建设中的，该建筑物的建设上限尚未达到，建设该建筑物的的技术已经学会。

```
CREATE FUNCTION castle_BuildableTable (@UID INT, @CID INT)
RETURNS TABLE
AS
RETURN   (
SELECT cs.ConstructionID AS ID, cs.[Name], cs.[Description],
     [Acreage], [People], [Money], cs.[Level], [Limit]
     ,ISNULL(outcic.ConstructionInCastleID,0) AS CCID
     ,CASE cs.[Level]
        WHEN 1 THEN 'BUILD'
        ELSE 'UPGRADE'
      END AS TYPE
     ,ISNULL(outcic.[XPosition],-1) AS [XPosition],ISNULL(outcic.[YPosition],-1) AS [YPosition]
    FROM Construction cs
    FULL OUTER JOIN [ConstructionInCastle] outcic ON cs.[UpgradedFrom] = outcic.[ConstructionID]
        AND outcic.[CastleID] = @CID
    WHERE
    -- 限制条件：前提建筑已建立
     (SELECT COUNT(*) FROM [ConstructionRequirement] cr
        WHERE cr.[ConstructionID]=cs.[ConstructionID]
         AND EXISTS(SELECT * FROM [ConstructionInCastle] cic
                    WHERE cic.[CastleID]=@CID
                     AND (SELECT [Level] FROM [Construction]
                            WHERE [Construction].[ConstructionID]=cr.[RequiredConstructionID])
                     <=(SELECT ISNULL(MAX(cicin.[Level]),0) FROM [ConstructionInCastle] cicin
                        JOIN [Construction] csin ON cicin.[ConstructionID] = csin.[ConstructionID]
                        WHERE cicin.[CastleID] = @CID
                            AND cicin.[Name]=(SELECT [Name] FROM [Construction]
                            WHERE [ConstructionID]=cr.[RequiredConstructionID]))
                    )
        )=(SELECT COUNT(*) FROM [ConstructionRequirement]
        WHERE [ConstructionRequirement].[ConstructionID]=cs.[ConstructionID])
    -- 限制条件：当前建筑物并非在建设中
    AND (outcic.[ConstructionStateID]=1 OR outcic.[ConstructionStateID] IS NULL)
    AND (NOT(outcic.[ConstructionInCastleID] IS NULL AND cs.[Level]>1) OR cs.[Level]=1)
    -- 限制条件：建设上限尚未达到
    AND cs.[Limit]>(SELECT COUNT(*) FROM [ConstructionInCastle] cic
        WHERE cic.[Name]=cs.[Name] AND [CastleID]=@CID
         AND cic.[Level]+(SELECT COUNT(*) FROM [ConstructionQueue] cqin , [ConstructionInCastle] cic
                    WHERE cqin.[ConstructionInCastleID]=cic.[ConstructionInCastleID]
                        AND cs.[Name]=cic.[Name])>=cs.[Level])
    -- 限制条件：该建筑所需要的技术已达到
    AND ((SELECT COUNT(*) FROM UserTechnology,RequirementBetweenTechnologyAndConstruction rbtac
    WHERE rbtac.ConstructionID=cs.ConstructionID AND rbtac.isTechnologyForConstruction=1) =
     (SELECT COUNT(DISTINCT ut.[TechnologyID])
      FROM UserTechnology ut,RequirementBetweenTechnologyAndConstruction rbtac
      WHERE rbtac.ConstructionID=cs.ConstructionID AND rbtac.isTechnologyForConstruction=1
       AND rbtac.TechnologyID=ut.TechnologyID
       AND ut.UserID=@UID)))
```

4.2. 更新状态

城堡的发展与其中的建筑物有密切的关系，这个存储过程便是用于计算城堡中建筑物的状态改变量，用于计算城堡发展情况。

一般来说，尽量避免使用游标，因为游标的效率不高，通常情况使用游标能解决的问题都能用查询语句解决，不过在本存储过程中，综合考虑编写难度、阅读难度、扩展难度、维护难度以及效率，最后我们决定使用游标来实现。事实上也表明这样做是很明智的选择。

在进行完所有的数据初始化之后，定义了一个游标，用于遍历该用户当前选定城堡的所有建筑物状态改变情况，并依次对其进行处理，会根据各种特殊情况选择性的进入相应的子过程，在其中增加自然资源产量子过程中嵌套了一个内层游标，用于遍历及处理所有自然资源产量。在本存储过程的末尾调用了两个存储过程，用于数据的进一步处理，具体内容在此不再赘述。

```
CREATE PROCEDURE castle_CalcStateChange
    @CID INT
AS
-- 初始化数据
UPDATE Castles SET [MaxPeople] = 0, [MaxStore] = 0 WHERE CastleID = @CID
-- 初始化所有资源产量，除了工作量
UPDATE ResourceInCastle SET NumberPerHour = 0 WHERE CastleID = @CID
 AND [ResourceID] <> (SELECT [ResourceID] FROM [Resources] WHERE [Operator] = 'WORKLOAD')
-- 声明用于外层循环的变量
DECLARE @OPERATOR VARCHAR(50), @TARGET INT, @VALUE INT, @WORKER INT, @TOTAL INT
-- 内层循环所需变量
DECLARE @RID INT, @PERCENT DECIMAL(18,2)
-- 取出所有的状态数据
DECLARE CSCURSOR CURSOR FOR
SELECT stct.Operator, Target, Value, cic.[Worker],cs.[People] FROM ConstructionStateToChange cstc
 JOIN StateToChangeType stct ON stct.StateToChangeTypeID = cstc.StateToChangeTypeID
 JOIN ConstructionInCastle cic ON cstc.ConstructionID = cic.ConstructionID
 JOIN [Construction] cs ON cic.[ConstructionID] = cs.[ConstructionID]
WHERE cic.CastleID = @CID
OPEN CSCURSOR
FETCH NEXT FROM CSCURSOR INTO @OPERATOR, @TARGET, @VALUE, @WORKER, @TOTAL
-- 主循环依次进行状态改变
WHILE @@FETCH_STATUS = 0
BEGIN
    SET @VALUE = @VALUE * @WORKER / @TOTAL
    ELSE IF @OPERATOR = 'INCREASERESOURCEOUTPUT' AND @TARGET <> 0 -- 增加资源产量
    BEGIN
        EXECUTE [resource_AddResourceOutput] @CID, @TARGET, @VALUE
    END
    ELSE IF @OPERATOR = 'INCREASENATURALRESOURCE' -- 增加自然资源产量
    BEGIN
        DECLARE INNERCURSOR CURSOR FOR
        SELECT [ResourceID], SUM([Percent]) FROM [ResourceCollection]
        WHERE [CastleID] = @CID GROUP BY [ResourceID]
        OPEN INNERCURSOR
        FETCH NEXT FROM INNERCURSOR INTO @RID, @PERCENT
        -- 内循环依次进行状态改变
        WHILE @@FETCH_STATUS = 0
        BEGIN
            DECLARE @PVALUE INT
            SET @PVALUE = @VALUE * @PERCENT
            EXECUTE [resource_AddResourceOutput] @CID, @RID, @PVALUE
            FETCH NEXT FROM INNERCURSOR INTO @RID, @PERCENT
```

```
        END
        CLOSE INNERCURSOR
        DEALLOCATE INNERCURSOR
    END
  -- 更多状态变化程序由于篇幅省略
    FETCH NEXT FROM CSCURSOR INTO @OPERATOR, @TARGET, @VALUE, @WORKER, @TOTAL
END
CLOSE CSCURSOR
DEALLOCATE CSCURSOR
-- 调用其他存储过程执行进一步更新状态
EXECUTE [resource_ApplyTimesSpeed] @CID
EXECUTE [resource_ClearResourceInCastle] @CID
```

4.3. 创建联盟

该存储过程实现功能：创建一个新联盟。要创建联盟首先第一步必须在[Alliances]添加联盟的基本信息，联盟不可能单独存在其必须拥有一定的成员,其联盟成员信息均存储在[AllianceMember]中，通过[AllianceID]外键进行区分，其创建者默认为联盟的盟主。另外我们系统中联盟模块的称号管理是有严格要求的，每个称号都有其相应的权限，所有的称号均存储在[MemberTitleInAlliance] 中，并通过 AllianceID 区分不同的联盟称号,称号相应的权限则存储在[AllianceMemberPermissionByTitle]中。对于一个联盟我们默认其两个称号分别是盟主和新人，这样设定的目的是为了联盟的某些管理的实现（如刚加入联盟的会员）。

```
CREATE      PROCEDURE alliance_CreateNewAlliance
    @UserID INT,
    @AllianceName VARCHAR(40),
    @AllianceIntro NTEXT,
    @Notice NTEXT,
    @Logo VARCHAR(255),
    @Icon VARCHAR(255),
    @Website VARCHAR(255),
    @Money INT,
    @IsOpenToAll BIT
AS
-- 创建一个新联盟
BEGIN
    INSERT INTO [Alliances] ( [UserID], [Name], [Intro], [Notice], [Logo], [Icon], [Website], [Money]
[IsOpenToAll])
    VALUES ( @UserID, @AllianceName, @AllianceIntro, @Notice, @Logo, @Icon, @Website, @MONEY, @IsOpenToAll)
    -- 新建联盟编号
    DECLARE @AllianceID INT
    SELECT  @AllianceID=[AllianceID] FROM [Alliances] WHERE [UserID]=@UserID
    -- 新建联盟创始人默认称号
    DECLARE @Title VARCHAR(40)
    SET @Title='盟主'
    DECLARE @NewMemberTitle VARCHAR(40)
    SET @NewMemberTitle='新人'
    -- 盟主
    INSERT INTO [MemberTitleInAlliance] ( [AllianceID], [Title], [Money])
    VALUES (  @AllianceID, @Title, @Money)
    -- 新人
    INSERT INTO [MemberTitleInAlliance] ( [AllianceID], [Title], [Money])
    VALUES ( @AllianceID, @NewMemberTitle, @Money )
    -- 联盟成员权限编号
    DECLARE @MemberTitleInAllianceID INT
    SELECT @MemberTitleInAllianceID=[MemberTitleInAllianceID] FROM [MemberTitleInAlliance] WHERE
[AllianceID]=@AllianceID AND [Title]=@Title
    -- 当前时间
    DECLARE @CreateAllianceTime DATETIME
```

```
    SET @CreateAllianceTime=GETDATE()
    -- 创建联盟新成员 盟主
    INSERT INTO [AllianceMember] ( [UserID], [AllianceID], [MemberTitleInAllianceID], [Money], [JoinDate])
    VALUES (  @UserID, @AllianceID, @MemberTitleInAllianceID, @Money, @CreateAllianceTime )
    -- 设定盟主权限
    DECLARE @NumRoles INT
    SELECT @NumRoles=COUNT([PermissionInAllianceID]) FROM [PermissionInAlliance]
    DECLARE @i INT
    SET @i=1
    WHILE @i<=@NumRoles
     BEGIN
         INSERT INTO [AllianceMemberPermissionByTitle] (
              [PermissionInAllianceID],
              [MemberTitleInAllianceID]
          ) VALUES (  @i, @MemberTitleInAllianceID )
          SET @i=@i+1
     END
END
```

4.4. 修改联盟权限

该存储过程实现功能：联盟称号权限修改。要实现称号权限的修改就必须对[AllianceMemberPermissionByTitle]表进行修改更新。因为一个称号的权限不止只有一个而是可能有多个但 T-Sql 不支持数组 ，若要修改权限就必须多次调用存储过程，所以在这里我们设想就用一个 VARCHAR 类型字符串模拟一个数组出来 ，字符串中每个元素以一个自定义的分隔符号（如 “ ; ”）。然后把传进来的字符串按特定分隔符号进行提取即可得到需要的数据。这样就可以一次调用存储过程并完成相应数据的传入和修改。又因为修改权限就要删除全来存在但现在不需要了的权限，为了保证修改的正确性，我们使用了简单的事务处理 这样就保证了修改更新权限和删除权限必须同时实现。

```
CREATE  PROCEDURE alliance_ModifyTitleInAlliancePermission
(
    @TID INT,
    @Permission VARCHAR(50)
)
AS
BEGIN
   BEGIN TRAN T1
      DELETE FROM [AllianceMemberPermissionByTitle]
      WHERE [MemberTitleInAllianceID]=@TID
    DECLARE @str VARCHAR(50)
    SET @str=@Permission
    DECLARE @value VARCHAR(50)
    DECLARE @i INT
    WHILE LEN(@str)>0
     BEGIN
         SELECT @i = Charindex(';',@str)
         SET @value = SUBSTRING ( @str , 1 , @i-1 )
         -- 用于处理所分离出来的数据
         INSERT INTO [AllianceMemberPermissionByTitle] (
                  [PermissionInAllianceID],
                  [MemberTitleInAllianceID]
              ) VALUES ( @value, @TID)
         SET @str = SUBSTRING ( @str , @i+1 , len(@str)-@i  )
     END
    COMMIT TRAN T1
    END
```

5. 系统实现与测试

5.1. 开发平台和工具选择

5.1.1. 系统建模工具：Power Designer

选择理由： Power Designer 系列产品提供了一个完整的建模解决方案，业务或系统分析人员，设计人员，数据库管理员 DBA 和开发人员可以对其裁剪以满足他们的特定的需要。Power Designer 灵活的分析和设计特性允许使用一种结构化的方法有效地创建数据库或数据仓库，而不要求严格遵循一个特定的方法学。Power Designer 提供了直观的符号表示使数据库的创建更加容易，并使项目组内的交流和通讯标准化，同时能更加简单地向非技术人员展示数据库和应用的设计。 Power Designer 不仅加速了开发的过程，也向最终用户提供了管理和访问项目的信息的一个有效的结构。它允许设计人员不仅创建和管理数据的结构，而且开发和利用数据的结构针对领先的开发工具环境快速地生成应用对象和数据敏感的组件。开发人员可以使用同样的物理数据模型查看数据库的结构和整理文档，以及生成应用对象和在开发过程中使用的组件。应用对象生成有助于在整个开发生命周期提供更多的控制和更高的生产率。Power Designer 是一个功能强大而使用简单工具集，提供了一个复杂的交互环境，支持开发生命周期的所有阶段，从处理流程建模到对象和组件的生成。Power Designer 产生的模型和应用可以不断地增长，适应并随着你的组织的变化而变化。

开发过程中的作用：进行系统建模，绘制设计相关的各类图形（如用例图、序列图、活动图、概念模型图、物理模型图等），并组织需求分析文档。

5.1.2. 系统开发平台：VS.Net2008(C#)

选择理由：VS.Net2008 是微软首推的开发平台，也是其花大量精力打造的华丽的开发平台，可以使开发人员的工作大量的减轻，并同时增加程序的可靠性，能够方便快捷的管理大型工程，而其中其首推的 C#语言更具有简单、现代、面向对象、类型安全、版本控制、兼容等优点，是 C++语言的扩充和发展。

其附带的工具更是丰富多彩，例如分析工具、源代码管理工具、单元测试工具等，这些都能大幅度的提高开发效率，使用 msdn 更是能够快速的定位程序帮助，使问题能够得到快速的解答。

开发过程中的作用：作为本系统的基本开发平台，承担了所有的程序编写工作，并使用其中的测试分析工具提高了程序的健壮性、稳定性和执行效率。

5.1.3. 数据库系统：SQL Server

选择理由：我们在进行数据选择的时候使用了一下数据库系统的选择原则：

1) 数据库系统采用易于集成的，开放的技术。

2) 产品质量优异，可靠性高，适于长期运行，能支持关键应用。

3) 数据安全，保安型高。

4) 能提供分布式数据库功能。

5) 支持多种开发环境，软件开发容易。

6) 扩充性和升级能力强。

而 SQL Server 作为微软推荐的配套数据库软件，特别适合于网络基础搭构，提供从很小到巨大数据库的可扩缩性存储。使用其中的存储过程能大幅度提高程序的运行效率，利用其中的索引优化引擎更能够大幅度提高数据库运行中的查询效率。满足了我们的要求，所以被我们选择用于我们的系统开发。

开发过程中的作用：作为本系统的基本数据库系统，用于存放数据，并进行存储过程的开发。

5.1.4. 其他工具

Screw turn wiki：用于文档协作，快速传达开发内容信息，明确各阶段工作目标等。

Microsoft Project：用于项目进度管理，是项目时间和质量的保证。

X2Blog：用于团队内部交流，以及向外部传达项目的开发情况，并在内测公测期间用于获得外部意见和建议。

Mind Manager：思维导图绘制软件，可以用于头脑风暴的快速记录，在团队合作中可以方便团队成员们的思想交流。

5.2.应用新技术

5.2.1. .NET Remoting

什么是 Remoting，简而言之，我们可以将其看作是一种分布式处理方式。从微软的产品角度来看，可以说 Remoting 就是 DCOM 的一种升 级，它改善了很多功能，并极好的融合到.Net 平台下。Microsoft® .NET Remoting 提供了一种允许对象通过应用程序域与另一对象进行交互的框架。这也正是我们使用 Remoting 的原因。为什么呢？在 Windows 操作系统中，是将应用 程序分离为单独的进程。这个进程形成了应用程序代码和数据周围的一道边界。如果不采用进程间通信（RPC）机制，则在一个进程中执行的代码就不能访问另一 进程。这是一种操

作系统对应用程序的保护机制。然而在某些情况下，我们需要跨过应用程序域，与另外的应用程序域进行通信，即穿越边界。

Remoting 的通道主要有两种：Tcp 和 Http。Tcp 通道提供了基于 Socket 的传输工具，使用 Tcp 协议来跨越 Remoting 边界传输序列化的消息流。TcpChannel 类型默认使用二进制格式序列化消息对象，因此它具有更高的传输性能。它提供了一种使用 Http 协议，使其能在 Internet 上穿越防火墙传输序列化消息流。默认情况下，HttpChannel 类型使用 Soap 格式序列化消息对象，因此它具有更好的互操作性。通常在局域网内，我们更多地使用 TcpChannel；如果要穿越防火墙，则使用 HttpChannel。

5.2.2. AJAX

术语 AJAX 用来描述一组技术，它使浏览器可以为用户提供更为自然的浏览体验。在 AJAX 之前，Web 站点强制用户进入提交/等待/重新显示范例，用户的动作总是与服务器的“思考时间”同步。AJAX 提供与服务器异步通信的能力，从而使用户从请求/响应的循环中解脱出来。借助于 AJAX，可以在用户单击按钮时，使用 JavaScript 和 DHTML 立即更新 UI，并向服务器发出异步请求，以执行更新或查询数据库。当请求返回时，就可以使用 JavaScript 和 CSS 来相应地更新 UI，而不是刷新整个页面。最重要的是，用户甚至不知道浏览器正在与服务器通信：Web 站点看起来是即时响应的。虽然 Ajax 所需的基础架构已经出现了一段时间，但直到最近异步请求的真正威力才得到利用。能够拥有一个响应极其灵敏的 Web 站点确实激动人心，因为它最终允许开发人员和设计人员使用标准的 HTML/CSS/JavaScript 堆栈创建“桌面风格的（desktop-like）”可用性。目前，AJAX 应用最普遍的领域使 GIS-Map，GIS 应用交互非常频繁，它强调快速响应，AJAX 的特点正好可以满足这类需要。

简单的说，AJAX 就局部刷新网页的一种机制。本系统在使用 AJAX 的使用是基于微软推出代号为 Atlas 的 AJAX 开发包。其实 Atlas 的功能将超越 AJAX 本身，包括整个 Visual Studio 的调试功能(客户端 JavaScript 教本的调试一直是困扰开发人员的难题)。另外，新的 ASP.NET 空间将客户端控件和服务器端代码的捆绑更为简便。Atlas 客户端教本框架（Atlas Client Script Framework）也使网页及相关项目的交互更为便利。

5.2.3. 可远程自动更新的服务器端

我们的服务器端程序使用 Windows Service 技术，能够稳定的运行在服务器上。我们使用可远程自动更新的技术，能够便捷的进行服务器升级及维护。

图 5.1 服务器端运行状态图

5.2.4. URL 重写

我们利用 URL 重写技术提高我们系统的用户体验，在地址栏显示如图 5.2 URL 重写后的地址栏所示，而实际地址为 http://localhost:3506/WebServer/Pages/Castle/Technology.aspx，这样使用户看不到他不需要看到的东西，能够提高用户体验，使用户只看见他所关心的信息。

图 5.2 URL 重写后的地址栏

5.3.系统测试

软件测试是又人工或计算机来执行或评价系统或系统部件的过程，以验证它是否满足规定的需求或识别期望的结果和实际结果之间的差别。如果认为测试是为了表明程序是正确的，从主观上不是为了查找错误而进行测试，测试者没有发现错误的愿望，这样的测试是不大会成功的。在本系统的测试过程中，我们小组抱着去发现错误并改正错误的态度去进行测试，发现了系统中存在的一些问题，在老师的帮助下改正了错误，完善了系统。

5.3.1. 测试方法介绍

5.3.1.1. 黑盒测试

也称功能测试或数据驱动测试，它是在已知产品所应具有的功能，通过测试来检测每个功能是否都能正常使用，在测试时，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，测试者在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息，并

且保持外部信息（如数据库或文件）的完整性。

黑盒测试方法主要有等价类划分、边值分析、因—果图、错误推测等，主要用于软件确认测试。“黑盒”法着眼于程序外部结构、不考虑内部逻辑结构、针对软件界面和软件功能进行测试。“黑盒”法是穷举输入测试，只有把所有可能的输入都作为测试情况使用，才能以这种方法查出程序中所有的错误。实际上测试情况有无穷多个，人们不仅要测试所有合法的输入，而且还要对那些不合法但是可能的输入进行测试。

5.3.1.2. 白盒测试

白盒测试也称结构测试或逻辑驱动测试，它是知道产品内部工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都有能按预定要求正确工作，而不顾它的功能，白盒测试的主要方法有逻辑驱动、基路测试等，主要用于软件验证“白盒”法全面了解程序内部逻辑结构、对所有逻辑路径进行测试。“白盒”法是穷举路径测试。

在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。贯穿程序的独立路径数是天文数字。但即使每条路径都测试了仍然可能有错误。第一，穷举路径测试决不能查出程序违反了设计规范，即程序本身是个错误的程序。第二，穷举路径测试不可能查出程序中因遗漏路径而出错。第三，穷举路径测试可能发现不了一些与数据相关的错误。

5.3.1.3. 灰盒测试

灰盒测试确实是介于二者之间的，可以这样理解，灰盒测试关注输出对于输入的正确性，同时也关注内部表现，但这种关注不象白盒那样详细、完整，只是通过一些表征性的现象、事件、标志来判断内部的运行状态，有时候输出是正确的，但内部其实已经错误了，这种情况非常多，如果每次都通过白盒测试来操作，效率会很低，因此需要采取这样的一种灰盒的方法。 灰盒测试结合了白盒测试盒黑盒测试的要素. 它考虑了用户端、特定的系统知识和操作环境。它在系统组件的协同性环境中评价应用软件的设计。灰盒测试由方法和工具组成，这些方法和工具取材于应用程序的内部知识盒与之交互的环境，能够用于黑盒测试以增强测试效率、错误发现和错误分析的效率。 灰盒测试涉及输入和输出，但使用关于代码和程序操作等通常在测试人员视野之外的信息设计测试。

5.3.2. 测试方案与测试策略

在测试方案的选择上我们选择了逻辑覆盖法，对系统进行了各种过程的测试。在单元测试时的策略是交换测试（既测试模块的人员不是编写该模块的人员）。采用白盒测试对模块进行静态和动态的测试。由于所有经过单元测试的模块，在组装中会出现新的问题，因此需要集成测试。集成测试采用的策略是自顶向下测试方式。在集成测试后进行确认测试，它是由独立的测试小组完成的。最后是系统测试，它是由用户单位组织实施的。在整个测试过程中，白盒测试达到完全覆盖程度，即语句覆盖率和分支覆盖率分别达到 100%。

5.3.3. 测试用例和结果

5.3.3.1. 对注册页面的测试

当不输入注册信息时，直接点注册按钮提交信息，系统会提示信息不能为空；当使用已经注册过的用户名进行注册时，系统会提示用户名已注册；当密码少于 6 位或者大于 12 位时，系统会提示格式不正确；当输入邮箱地址时，必须符合邮箱的格式，要不然系统会提示请输入正确格式；验证码是系统随机生成的，必须立即注册，超时会提示验证码错误；当输入了正确的信息，点击注册按钮提交，系统验证通过，系统会直接跳转到用户登陆页面。这些都经过测试，验证通过。

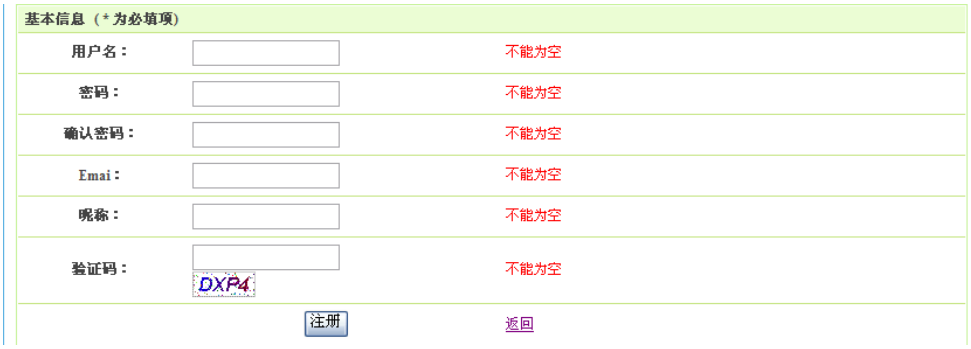


图 5.3 错误提示

5.3.3.2. 英雄招募与解雇的测试

当用户登陆成功，进入系统英雄操作模块，玩家可以在英雄列表选择自己喜欢的英雄，对其进行招募和解雇。当招募成功后，所在城堡会增加刚招募的英雄，而招募处则相应减少一名英雄。玩家可以根据需要，让英雄学习各种技能、对英雄的各种属性进行需要的升级、也可以对其装配和脱去宝物，按照当初设计要求，测试都达到要求。

图 5.4 英雄雇佣图

图 5.5 英雄属性操作和解雇

5.3.3.3. 联盟管理

当一个玩家建立联盟后，系统就设置该玩家为盟主。盟主可以对该盟进行外交管理，称号管理，资产管理等操作。盟主可以创建称号，并且可以给改称号设置不同的权限，也可以修改该称号的权限。其他玩家可以通过申请加入该联盟，通过盟主的审批后，就会被添加到该联盟中，盟主可以给所在联盟的成员设置称号，各称号可以设置不同的权限。这部分主要是测试盟主给成员修改称号，各称号设置不同权限是否成功，成员是否能通过所赋称号进行相关的权限操作。

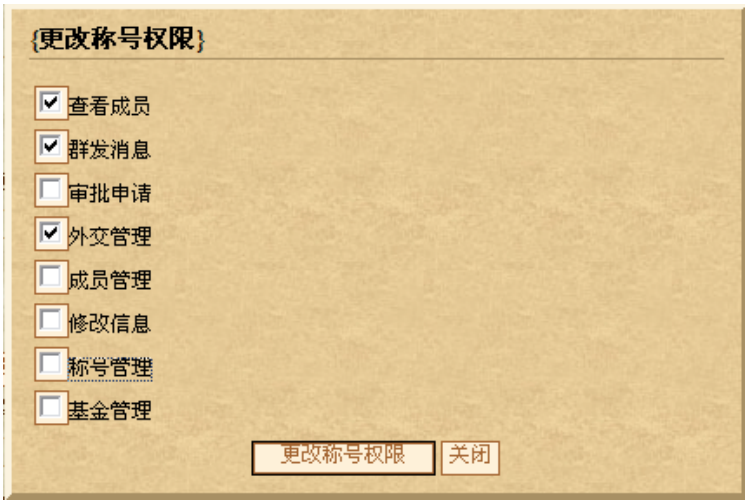


图 5.6 更改称号“外交官”权限

图 5.7 称号“外交官”权限设置成功

图 5.8 盟主审批

图 5.9 盟主给成员修改称号

刚开始默认是新人，也只有查看成员，群发消息两个权限。测试时把他修改为外交官，检查其是否有与称号匹配权限。通过测试该部分设置成功。

图 5.10 成员称号修改为“外交官”

登陆成员，看是否有称号——外交官的权限。通过测试该部分权限设置成功。

5.3.4. 测试环境与测试辅助工具

测试环境	VS2008
测试辅助工具	UnitTest

5.3.5. 测试完成准则

对于非严格系统可以采用“基于测试用例”的准则：

（1）功能性测试用例通过率达到 100%；

（2）非功能性测试用例通过率达到 95%时。

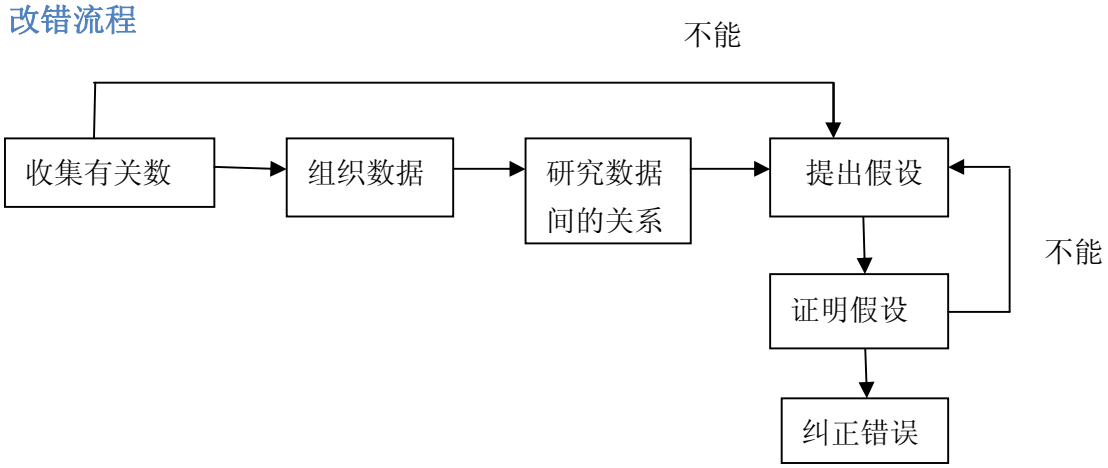
对于严格系统，应当补充“基于缺陷密度”的规则：

（3）相邻 n 个 CPU 小时内“测试期缺陷密度”全部低于某个值 m。例如 n 大于 10，m 小于等于 1。

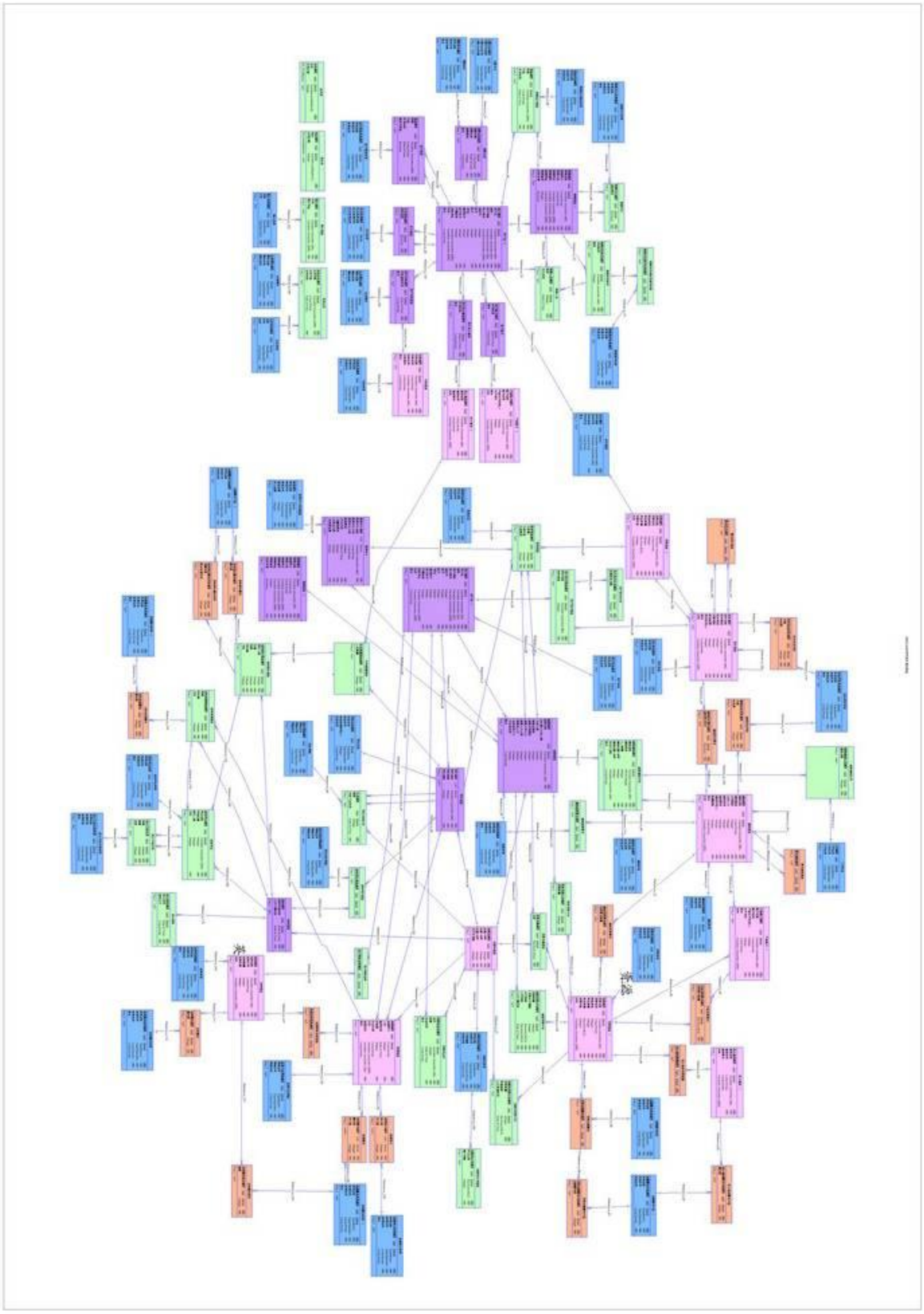
5.3.6. 人员与任务表

人员	角色	职责、任务
梁爽	系统设计	系统框架设计及系统运行
梁四六	软件设计	数据方面设计
姚淞文	软件设计	程序方面设计

5.3.7. 改错流程



附录一



附录二