**数据库系统原理与应用课程设计报告**

项目名称：出租车管理系统

组长：方志晗 1152703

组员：路 捷 1152726

程 冉 1152692

洪海杰 1152747

盛 雯 1152729

宁 婧 1152706

赵 青 1152683

王 成 1152813

2013年8月1日星期四

1. **引言**

本设计为一个基本的出租车信息管理系统，整个系统分为前端的门户网站和后台的信息管理系统，对出租车公司中常见的新闻发布、车辆交易、客户投诉处理。。。等活动以及相关的车主、司机、车辆、客户等基本信息进行管理。整个系统从符合操作简便、界面友好、灵活、实用、安全的要求出发，完成。。。。的全过程，包括新闻公告的发布与删除，司机的基本信息查看、增加和删除，车辆交易、车辆的增加和删除，车主的增加和删除，客户基本信息查看、客户个人积分统计，完整备份等。该出租车管理系统功能强健，完全可以满足出租车公司业务及其管理需求。

通过对系统的初步调查，明确了系统开发的目标为：按照管理信息系统的原理和方法，采用成熟、先进的信息技术和手段，支持出租车公司员工管理、车辆交易、客户信息管理以及客户反馈、数据库备份的全过程，加强出租车公司管理活动中各种信息资源的管理和利用，实现出租车公司信息管理现代化，及时、准确地提供各种信息。

1. **系统分析**

**2.1 现行系统的总目标**

通过对出租车公司管理机制的研究以及向有关人员的咨询，对出租车公司的各个方面的工作流程有了具体的了解，经过小组成员的多次协商，最终确定了在出租车信息管理系统中需要实现的主要功能。

**2.2 系统实现的主要功能**

管理员（经理）：

1. 新闻公告：发布、预览、删除
2. 司机：新建、查看基本信息、删除
3. 车辆：交易、增加、删除
4. 车主：车辆交易、车主信息查看、修改、车主删除
5. 客户：查看基本信息、查看客户积分情况
6. 信息：数据库备份

客户：

1. 新闻公告：查看
2. 客户：以客户ID或者邮箱登录查看积分情况，也可根据未被登记的发票换取积分，系统会根据积分情况赋予客户一个特定的等级，客户可以用积分换取乘车优惠
3. **系统概要设计**

**3.1 数据库设计**

数据库设计是开发数据库及其应用系统的技术，也是信息系统开发和建设的重要组成部分。具体的说，数据库设计是要在一个给定的应用环境中，通过合理的逻辑设计和有效的物理设计，构造较优的数据库模式，建立数据库及其应用系统，满足用户各种信息需求。

**3.2 系统总ER图**

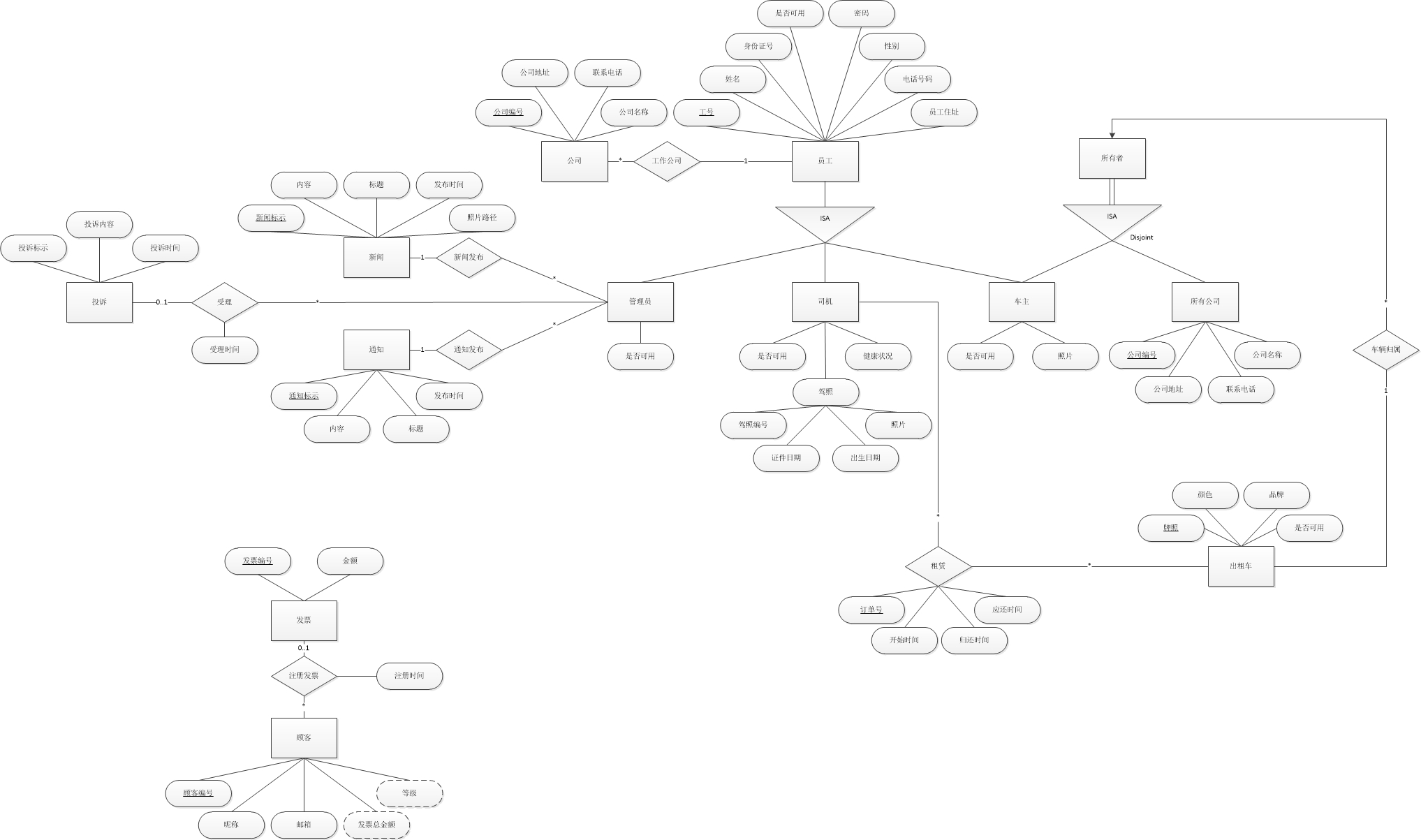


图3.2-1

**3.3 实体描述**

本系统中所涉及的实体一共有13个，其结构如下：

**表3.3-1 实体集说明表**

|  |  |
| --- | --- |
| 实体集名称 | 存储信息 |
| 投诉 | 投诉编号，投诉内容，投诉时间 |
| 新闻 | 新闻编号，新闻标题，新闻内容，新闻图片，发布时间 |
| 通知 | 通知编号，通知标题，通知内容，发布时间 |
| 员工 | 员工编号，姓名，身份证号，登录密码，性别，电话号码，地址 |
| 经理 | 是否可用（是否可用表示删除标志位，0表示不可用，即已删除，1表示可用。） |
| 司机 | 是否可用，健康状况，驾照（驾照编号，证件日期，出生日期，照片） |
| 车主 | 是否可用，照片 |
| 所有者 | - |
| 所有者所在公司 | 公司编号，公司地址，联系电话，公司名称 |
| 出租车 | 车牌号，颜色，品牌，是否可用 |
| 订单 | 订单号，开始时间，应该归还时间，实际归还时间 |
| 发票 | 发票编号，金额 |
| 顾客 | 顾客编号，昵称，邮箱，发票总金额，等级 |

**3.4 联系集描述**

本系统中所涉及的联系集一共有7个，其关系如下：

**表3.4-1 联系集说明表**

|  |  |  |  |
| --- | --- | --- | --- |
| 联系集名称 | 联系集特有属性 | 参与联系集的实体集 | 联系集描述 |
| 投诉受理 | 受理时间 | 投诉、管理员 | 顾客在前台匿名投诉，管理员需要在后端受理投诉，自动生成受理时间 |
| 新闻发布 |  | 新闻、管理员 | 管理员要定时发布新闻，提供给前台的访问者查看 |
| 通知发布 |  | 通知、管理员 | 管理员负责发布通知，主要面向的是公司内部的员工，但前台的访问者也可查看 |
| 工作公司 |  | 公司、员工 | 每个员工都要率属于一个特定的公司 |
| 车辆归属 |  | 所有者、出租车 | 每辆出租车都有真正的主人，这个所有者拥有出租车的所有权 |
| （车辆归属）车辆-车主 |  | 出租车、订单 | 每辆出租车都要对应一笔订单，记录该出租车由司机支配 |
| （租赁）司机-出租车 |  | 司机、订单 | 司机与出租车之间的关系凭证，司机要根据订单对应一辆特定的出租车 |
| 注册发票 | 注册时间 | 发票、顾客 | 顾客每次乘车都会得到一张发票，上面有具体日期 |

对应关系说明：

1. 投诉受理关系：投诉与管理员的关系是0，1：\*，因为一个投诉可能被受理，也可能未被受理，最多被一个管理员受理，但一个管理员可以受理多个客户投诉。
2. 新闻发布关系：新闻与管理员的关系是1：\*，因为每个新闻都要由一个管理员来发布，但同一个管理员可以发布多条新闻。
3. 公告发布关系：公告与管理员的关系是1：\*，因为每个公告都要由一个管理员来发布，但同一个管理员可以发布多条公告。

注意：新闻和公告的区别：新闻主要面向所有人，无论是公司内部雇员，还是访问门户网站的客户，而公告主要是面对公司内部雇员。

1. 工作公司：公司与员工的关系是\*：1，因为一个公司可以容纳多个员工，而任意一个员工都只能率属于一家公司。
2. 车辆归属关系：所有者和出租车的关系是\*：1，任意一个所有者可以拥有多辆出租车，而任意一辆出租车只能有一个主人。
3. 租赁关系：司机与出租车的关系是\*：\*，因为任意一个司机至多拥有一辆出租车，任意一辆出租车也至多由一个司机驾驶，但针对系统中所有租赁订单关系，司机与出租车的关系是\*：\*。
4. 注册发票：发票与顾客的关系是0，1：\*，因为发票是统一录入的，那时发票还不对应特定的顾客，待发票到顾客手里之后，每张发票只能对应一个顾客，而顾客只要多次乘车，就能获得多张发票。
5. 系统详细设计

**4.1 数据库的逻辑设计**

数据库的逻辑设计的任务就是把概念结构设计阶段的基本E－R图转化为与选用具体机器上的DBMS

产品所支持的数据模型相符合的逻辑结构，首先要实现的是E－R图关系模型的转化。而为此要解决的问题是如何将实体和实体之间的联系转化为关系模式，

如何确定这些关系模式的属性和码。对于实体，将每个实体转换为一个关系，实体的属性即为关系的属性，实体的码即为关系的码。

对于实体间的联系，可以分成三种情况：

1. 若实体间的联系是1：1，可以在两个实体转换成的两个关系中任意一个关系的属性中加入另一个关系的码。
2. 若实体间的联系是1：n，则在n端实体转换成的关系中加入1端实体转换成的关系码。
3. 若实体间的联系是n：m，则将联系转换为关系，关系的属性为诸个实体的码加上联系具有的属性，而关系的码则为诸实体的码的组合。

**4.2 数据库的实现**

### 4.2-1 employee表

表格 4.2‑1 empolyee表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| empolyee\_id | int | 默认 | 员工工号 | PK。 |
| password | varchar | 50 | 密码 | 储存密码，非空。 |
| firm\_id | int | 默认 | 所属公司编号 | FK，参照了firm表的firm\_id属性，非空。 |
| name | nvarchar | 50 | 员工姓名 | 非空。 |
| id\_card | char | 20 | 身份证号 | 唯一、非空。 |
| birthday | date | 默认 | 出生日期 | 非空。 |
| gender | bit | 默认 | 员工性别 | 1为男性，0为女性。 |
| telephone | varchar | 20 | 联系电话 | 非空。 |
| employee\_address | nvarchar | 50 | 员工住址 | 非空。 |
| is\_available | bit | 默认 | 标记是否可用，是否被系统删除 | 非空，1为可用，0为不可用。默认值为1。 |

### driver表

表格 4.2-‑2 driver表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| employee\_id | int | 默认 | 员工工号 | PK；FK，参照empolyee表中的empolyee\_id属性。 |
| health | int | 默认 | 司机健康状况 | 可为空。 |
| license\_id | varchar | 50 | 司机驾照号 | FK，参照了license表的license\_id属性。 |
| is\_available | bit | 默认 | 是否可用 | 非空，1为可用，0为不可用。默认值为1。 |

### host表

表格 4.2‑3 host 表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| employee\_id | int | 默认 | 员工工号 | PK；FK，参照empolyee表中的empolyee\_id属性。 |
| photo | nvarchar | 200 | 车主照片路径 | 可为空。 |
| is\_available | bit | 默认 | 是否可用 | 非空，1为可用，0为不可用。默认值为1。 |

### manager表

表格 4.2‑4 manager表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| employee\_id | int | 默认 | 员工工号 | PK；FK，参照empolyee表中的empolyee\_id属性。 |
| is\_available | bit | 默认 | 是否可用 | 非空，1为可用，0为不可用。默认值为1。 |

### firm表

表格 **4.2**‑5 firm表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| firm\_id | int | 默认 | 公司编号 | PK。 |
| firm\_name | nvarchar | 20 | 公司名称 | 非空。 |
| firm\_address | nvarchar | 50 | 公司地址 | 非空。 |
| phone\_number | varchar | 20 | 联系号码 | 非空。 |

### license表

表格 4.2‑6 license表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| license\_id | varchar | 50 | 驾照编号 | PK。 |
| license\_time | date | 默认 | 驾照颁发日期 | 非空。 |
| birthday | date | 默认 | 驾驶员出生日期 | 非空。 |
| photo\_path | nvarchar | 256 | 驾照复印件照片 | 非空，储存实际照片的路径。 |

### taxi表

表格 4.2‑7 taxi表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| plate\_number | nvarchar | 50 | 牌照 | PK。 |
| taxi\_color | nvarchar | 20 | 车辆颜色 | 非空。 |
| taxi\_brand | nvarchar | 20 | 车辆品牌 | 非空。 |
| is\_available | bit | 默认 | 是否可用 | 非空，1为可用，0为不可用。默认值为1。 |
| owner\_firm\_id | int | 默认 | 车辆归属公司公司编号 | 与owner\_empolyee\_id有且仅有一个属性为空值。 |
| owner\_empolyee\_id | int | 默认 | 所有者工号 | 与owner\_firm\_id有且仅有一个属性为空值。 |

### order表

表格 4.2‑8 order表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| order\_id | int | 默认 | 订单标识 | PK，自增长。 |
| plate\_number | nvarchar | 50 | 出租车牌照号 | FK，参照了taxi表的plate\_number属性；非空。 |
| employee\_id | int | 默认 | 司机工号 | FK，参照了driver表的empolyee\_id属性。 |
| rent\_begin\_time | date | 默认 | 订单开始日期 | 非空。 |
| rent\_due\_return\_time | date | 默认 | 订单应结束日期 | 非空。 |
| rent\_return\_time | date | 默认 | 车辆实际归还日期 | 当且仅当车辆未归还时为空值，并且通过判断该属性是否为空判断是否是当前的订单。 |

### owner\_firm表

表格 4.2‑9 owner\_firm表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| firm\_id | int | 默认 | 车辆归属公司编号 | PK。 |
| firm\_name | nvarchar | 50 | 公司名称 | 非空。 |
| firm\_address | nvarchar | 50 | 公司地址 | 非空。 |
| phone\_number | varchar | 20 | 联系电话 | 非空。 |

### mews表

表格 4.2‑10 news表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| news\_id | int | 默认 | 新闻编号 | PK，自增长。 |
| title | nvarchar | 50 | 标题 | 非空。 |
| employee\_id | int | 默认 | 发布者工号 | FK，参照manager表的empolyee\_id属性。 |
| publish\_time | smalldatetime | 默认 | 发布时间 | 可精确到分钟，默认值为getdate()，非空。 |
| picture\_path | nvarchar | 256 | 配图地址 | 可为空。 |
| news\_content | ntext | 默认 | 新闻内容 | 非空 |

### notice表

表格 4.2‑11 notice表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| notice\_id | int | 默认 | 通知编号 | PK；自增长。 |
| title | nvarchar | 50 | 通知标题 | 非空。 |
| employee\_id | int | 默认 | 发布者工号 | FK，参照manager表的empolyee\_id属性。 |
| publish\_time | smalldatetime | 默认 | 发布时间 | 可精确到分钟，默认值为getdate()，非空。 |
| notice\_content | ntext | 默认 | 通知内容 | 非空 |

### complaint表

表格 4.2‑12 complaint表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| complaint\_id | int | 默认 | 投诉编号 | PK，自增长。 |
| customer\_name | nvarchar | 20 | 顾客姓名 | 非空。 |
| customer\_email | varchar | 50 | 顾客邮箱 | 非空。 |
| complaint\_content | text | 默认 | 投诉内容 | 非空。 |
| complaint\_time | smalldatetime | 默认 | 投诉事件 | 可精确到分钟，默认值为getdate()， 非空。 |
| employee\_id | int | 默认 | 受理管理员工号 | FK，参照manager表的empolyee\_id属性。当且仅当投诉还未受理时，empolyee\_id属性为空。 |

### customer表

表格 4.2‑13 customer表

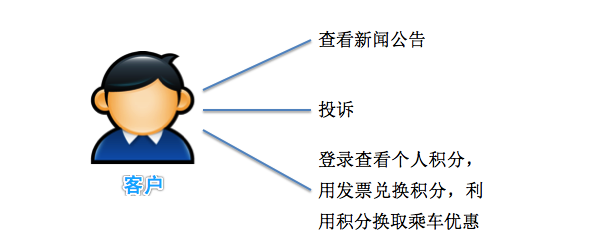
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| customer\_id | int | 默认 | 顾客编号 | PK，自增长。 |
| password | varchar | 50 | 登陆密码 | 非空。 |
| nick\_name | nvarchar | 50 | 昵称 | 非空。 |
| email | varchar | 50 | 邮箱 | 唯一且非空，登陆和找回密码的凭证 |

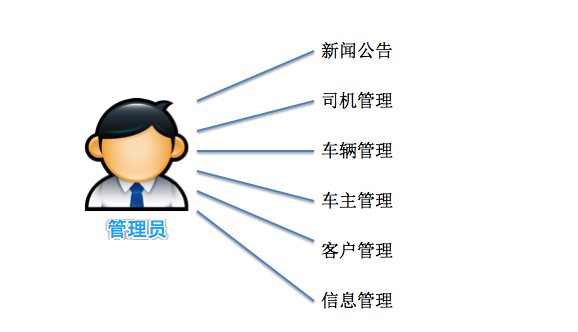
### invoice表

表格 4.2-‑14 invoice表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| invoice\_id | int | 默认 | 发票编号 | PK，实际生活中中会有固定值，再次简单使用自增长。 |
| create\_time | date | 默认 | 发票日期 | 非空，默认值为getdate()。 |
| amount | money | 默认 | 发票金额 | 非空 |
| customer\_id | int | 默认 | 注册顾客号 | FK，参照customer表的customer\_id属性，可空。 |
| regist\_time | smalldatetime | 默认 | 发票注册日期 | 可空。 |

* 1. **系统角色活动设计**

****

****

* 1. **功能逻辑设计**
  2. **关于系统角色主要活动的数据库操作说明**
  3. **人机交互界面设计**

用户界面是指软件系统和用户交互的接口，良好的用户界面往往可以提高用户使用系统的热情，并刺激他们的积极性。

1. 程序说明

## 两个主要控制器

### 5.1.1类名：FrontPageController

作用：负责前段页面的事务处理，主要处理前端事务，包括首页，新闻，公告，投诉，报销，注册，客户登录注销和改密码等操作的事务处理。其中主要包括重定位动作和视图动作。具体动作对应如下。

|  |  |  |
| --- | --- | --- |
| 动作名 | 属性 | 作用 |
| CustomerLoginHandle | 视图动作 | 验证客户用户名密码 |
| ChangePasswordHandle | 视图动作 | 修改客户密码 |
| RegisterHandle | 视图动作 | 注册客户 |
| GetCredit | 视图动作 | 获取客户积分 |
| News | 视图动作 | 新闻列表 |
| Notification | 视图动作 | 公告列表 |
| Galley | 视图动作 | 画廊 |
| Elements | 视图动作 | 发票积分 |
| Login | 视图动作 | 客户登录 |
| Complain | 视图动作 | 客户投诉 |
| NewsContent | 试图动作 | 新闻内容 |
| NotificationContent | 试图动作 | 公告内容 |

### HomeController类

作用：负责后台页面的事务处理，主要处理管理事务，包括角色管理，工号管理，账号管理，组织管理，新闻公告，车辆管理，车主管理，司机管理，信息管理，客户管理十大模块。具体动作对应如下。

|  |  |  |
| --- | --- | --- |
| 动作名 | 属性 | 作用 |
| BackHandle | 视图动作 | 返回主页面 |
| SetNewBackup | 视图动作 | 创建新备份文件 |
| ChangePasswordHandle | 重定向动作 | 更改经理密码 |
| Login | 视图动作 | 登录 |
| AddNotice | 视图动作 | 添加公告 |
| SaveNewNews | 普通函数 | 保存新闻 |
| SaveNewNotice | 普通函数 | 保存公告 |
| SaveChangeNotice | 普通函数 | 保存修改的公告 |
| ChangeNews | 视图动作 | 修改新闻 |
| ChangeNotice | 视图动作 | 修改公告 |
| deleteNews | 普通函数 | 删除新闻 |
| deleteNotice | 普通函数 | 删除公告 |
| Index | 视图动作 | 欢迎界面 |
| Customer | 视图动作 | 客户列表 |
| TaxiList | 视图动作 | 车辆列表 |
| NewsList | 视图动作 | 新闻列表 |
| InformationList | 视图动作 | 公告列表 |
| NewsContent | 视图动作 | 新闻内容 |
| NotificationContent | 视图动作 | 公告内容 |
| InvoiceList | 视图动作 | 发票列表 |
| EmployeeInfo | 视图动作 | 雇员信息 |
| ComplainList | 视图动作 | 投诉列表 |
| AcceptedComplaintSearchList | 视图动作 | 受理投诉列表 |
| NotAcceptedComplaintSearchList | 视图动作 | 未受理投诉列表 |
| ComplainAccept | 视图动作 | 投诉受理 |
| ComplainDelete | 视图动作 | 投诉删除 |
| ComplainListAccepted | 视图动作 | 投诉受理列表 |
| DriverList() | 视图动作 | 司机列表 |
| AddDriver() | 视图动作 | 添加司机 |
| DriverInfo | 视图动作 | 司机信息 |
| DriverSelect | 视图动作 | 司机选择 |
| HostList | 视图动作 | 车主信息 |
| HostBuyTaxi | 视图动作 | 车辆交易 |
| HostBuyTaxi | 视图动作 |  |
| getTaxiInfoByID | 视图动作 |  |
| hostTaxiDetail | 视图动作 | 车辆交易 |
|  |  |  |
| HostTransaction | 视图动作 | 车辆交易 |
| AddHost | 视图动作 | 添加车主 |
| EditHost | 视图动作 | 修改车主 |
| DeleteHost | 视图动作 | 删除车主 |
| AddEmployee | 视图动作 | 添加工号 |
| AddEmployeeHandle | 重定向动作 | 添加工号 |
| HostInfo | 视图动作 | 车主信息 |
| HostTaxiInfo | 视图动作 | 车主车辆信息表 |
| HostSearchResult | 视图动作 | 车主搜索列表 |
| AddNews | 视图动作 | 添加新闻 |
| BackupList | 视图动作 | 备份列表 |
| DeleteEmployee | 视图动作 | 删除工号 |
| ManagerList | 视图动作 | 经理列表 |
| AddManager | 视图动作 | 添加经理 |
| ModifyEmployee | 视图动作 | 修改工号 |
| ModifyManager | 视图动作 | 修改经理 |
| ManagerInfo | 视图动作 | 经理信息 |
| ManagerSelfInfo | 视图动作 | 登陆经理信息 |
| GetSelfManagerModify | 视图动作 | 得到经理自身信息 |
| FirmList | 视图动作 | 公司列表 |
| ModifySelfInfo | 视图动作 | 修改自身信息 |
| LogOut | 视图动作 | 退出 |
| DeleteDriver | 视图动作 | 删除司机 |
| SaveDriver | 视图动作 | 保存 |
| SaveDriverInfo | 视图动作 | 保存司机信息 |
| EmployeeList | 视图动作 | 工号列表 |
| CarInfoDisplay | 视图动作 | 车辆信息 |
| Backup | 视图动作 | 备份 |

## 5.2主要视图类

|  |  |
| --- | --- |
| 主要类 | 功能 |
| Driver | 抽象数据库中的driver表，为逻辑层建立实体数据对象，供给controller和Handler控制和处理，并作为表示在view中的数据模型，类似于JavaEE中的JavaBean。 |
| Driverhandle | 负责处理相应的driver类的数据，包括从数据库的函数中提取查询结果集，将数据通过存储过程存储到数据库中。还有一些关于name和age的附加处理方法，实现了司机列表的分页查询，模糊搜索等功能。 |
| News | 抽象数据库中的news表，具体功能与driver相类似 |
| Newshandle | 负责处理新闻，新闻同样具有模糊搜索功能，分页，还有动态格式添加显示，html版式解码和编码等功能 |
| Notice | 抽象数据库中的notice表，具体功能与driver类似 |
| NoticeHandle | 负责处理notice的类，主要跟frontpage的交互有关联，负责notice和noticeContent显示的数据控制 |
| Employee | 抽象数据库中的employee表，具体功能跟driver的类似 |
| HomeController | 添加HomeController中的相应的处理driver，news，notice的接口。具体函数包括：AddDriver、EditDriver、DeleteDriver、SaveDriverInfo、DriverInfo，以及Notice、News、NoticeContent、NewsContent等接口 |
| FrontController | FrontController的相应处理函数，包括对Notice和news的显示控制和分页控制等 |

车辆管理

Model类

1. TaxiRepository

直接使用LINQ to SQL DataContext 类,让维护和测试更方便的方法是使用Repository 模式。Repository 类帮助封装数据查询和存储逻辑，从应用程序中抽象隔离具体的数据存储实现。除了是应用程序代码更加简洁外，使用 Repository 模式使将来更改数据库存储实现更加方便，并且有助于在没有真实数据库的情况下，进行应用程序进行单元测试。

方法说明，例：

private DataClasses1DataContext db = new DataClasses1DataContext() ;

public IQueryable<getAllTaxiInformationResult> FindAllTaxis(){

return db.getAllTaxiInformation();

}

说明：在TaxiRepository类中定义LINQ to SQL DataContext类，调用数据库中的函数或存储过程完成数据存储。FindAllTaxis() 方法返回的IQueryable<getAllTaxiInformationResult> 对象封装了使用LINQ to SQL 从数据库中检索getAllTaxiInformationResult对象的查询。

1. PaginatedList

简单PaginatedList 辅助类是本系统中的分页实现方法之一，用于车辆管理模块，继承.NET Framework 内置的List<T>集合类。它实现了一个可重

用的集合类， 用来在任何IQueryable 序列中实现分页。在NerdDinner 范例程序中， 用来对

IQueryable< getAllTaxiInformationResult> 结果集进行分页。

方法说明，例：

public PaginatedList(IQueryable<T> source, int pageIndex, int pageSize) {

PageIndex = pageIndex;

PageSize = pageSize;

TotalCount = source.Count();

TotalPages = (int) Math.Ceiling(TotalCount / (double)PageSize);

this.AddRange(source.Skip(PageIndex \* PageSize).Take(PageSize));

}

该方法通过接受源集合，页面索引，每页显示条目数量，返回页码为pageIndex的分页结果

1. 此方法在技术实现上并无太大难点，但很典型的展现了controller类与View的交互和数据传递，处理，与响应。

我们将同时支持HTTP GET 和HTTP POST 方法访问这些新的URL 地址。HTTP GET 对这些URL 地址的请求

将显示初始的HTML 视图，EditCarInfo 将显示 getTaxiInformationByPlatenumberResult数据

/Controller/HomeController

首先实现edit action 方法的HTTP GET 动作：

public ActionResult EditCarInfo(string id)

{

getTaxiInformationByPlatenumberResult Taxi = taxiRepository.FindTaxiByPlatenumber(id);

IQueryable<getAllHostResult> Hosts = taxiRepository.GetAllHosts();

IQueryable<getAllTaxiPlatenumberResult> plateNos = taxiRepository.GetAllPlateNumbers();

List<int> hosts\_id = new List<int>();

List<string> plate\_numbers = new List<string>();

foreach (var Host in Hosts)

{

hosts\_id.Add(Host.empolyee\_id);

}

foreach (var plate\_number in plateNos)

{

plate\_numbers.Add(plate\_number.plate\_number);

}

ViewData["hosts"] = new SelectList(hosts\_id, Taxi.host\_empolyee\_id);

ViewData["plate\_numbers"] = new SelectList(plate\_numbers, Taxi.plate\_number);

IQueryable<getAllDriverResult> Drivers = taxiRepository.GetAllDrivers();

List<int> drivers\_id = new List<int>();

foreach (var Driver in Drivers)

{

drivers\_id.Add(Driver.empolyee\_id);

}

ViewData["drivers"] = new SelectList(drivers\_id, Taxi.driver\_empolyee\_id);

if (Taxi == null)

return View("Error");

else

return View(Taxi);

}

上述页面的HTML 脚本如下所示，就是一个标准的HTML-有一个<form>元素。在点击Save 保存按钮后，

向 /Home/EditCarInfo/\* 执行HTTP POST 动作。其中HTML 元素 - <input type=”text” />元素用来展示可编辑的文本框。

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult EditCarInfo(string id, FormCollection formValues)

{

getTaxiInformationByPlatenumberResult Taxi = taxiRepository.FindTaxiByPlatenumber(id);

// try

//{

Taxi.plate\_number = Request.Form["plate\_number"];

Taxi.taxi\_color = Request.Form["taxi\_color"];

Taxi.taxi\_brand = Request.Form["taxi\_brand"];

Taxi.host\_empolyee\_id = int.Parse(Request.Form["host\_employee\_id"]);

taxiRepository.UpdateTaxi(Taxi.plate\_number, Taxi.taxi\_color, Taxi.taxi\_brand, Taxi.host\_empolyee\_id);

return RedirectToAction("CarInfoDisplay", new { id = Taxi.plate\_number });

//}}