**数据库课程设计报告**

-------物业管理信息系统

软件学院软件工程专业2014级4班

姓名：雷超 学号：201400301291

任课教师：任国珍

**目录**

[一、系统开发平台 1](#_Toc431114790)

[二、数据库规划 1](#_Toc431114791)

[2.1 任务陈述 1](#_Toc431114792)

[2.2 任务目标 1](#_Toc431114793)

[三、系统定义 2](#_Toc431114794)

[3.1 系统边界 2](#_Toc431114795)

[3.2 用户视图 2](#_Toc431114796)

[四、需求分析 3](#_Toc431114797)

[4.1 用户需求说明 3](#_Toc431114798)

[4.1.1 数据需求 3](#_Toc431114799)

[4.1.2 事务需求 3](#_Toc431114800)

[4.2 系统需求说明 4](#_Toc431114801)

[五、数据库逻辑设计 5](#_Toc431114802)

[5.1 ER图 5](#_Toc431114803)

[5.2 数据字典 5](#_Toc431114804)

[5.3 关系表 7](#_Toc431114805)

[六、数据库物理设计 11](#_Toc431114806)

[6.1 索引 11](#_Toc431114807)

[6.2 安全机制 12](#_Toc431114808)

[6.3 其他 12](#_Toc431114809)

[七、应用程序设计 12](#_Toc431114810)

[7.1 功能模块 12](#_Toc431114811)

[7.2 界面设计 13](#_Toc431114812)

[7.3 事务设计 20](#_Toc431114813)

[7.4 C/S交互设计 20](#_Toc431114813)

[7.5 Linux内核设计 20](#_Toc431114813)

[八、测试和运行 23](#_Toc431114814)

[九、总结 27](#_Toc431114815)

[附. 参考文献 27](#_Toc431114816)

# 一、系统开发平台

题目：**物业管理信息系统**

物业管理信息系统软件是一套用于管理居民小区信息的系统,主要信息包括：业主信息、房屋信息、物业管理费用信息、水电煤气、有线电视等费用信息、供暖费用信息,并可输出上述信息报表(月报表、季报表、年报表).

1．记录业主基本信息如:身份证号、楼号、房号、姓名、工作单位、联系电话、建筑面积等,并生成业主的综合信息表（包括所有业主有费用综合信息表、单个业主信息表）；

2．记录月水电煤气抄表数据并打印业主交费通知单,包括水电煤气有线电视费通知单、供暖费通知单、物业管理费通知单；

3．记录业主交费情况并打印交费单,包括水电煤气费、供暖费、物业管理费、分期付款；

4．统计业主信息月、季和年报表,包括：物业实收费用汇总表、物业应收未收汇总表、应交未交费用业主信息表、月度房款还款名细表、单用户年度应收代款还款表.

5. 网络多用户操作,具备多人同时联网操作功能

6.异常处理

开发工具:

dev-qt/qt-creator 4.0.3

app-editors/emacs 25.1

开发环境:

dev-qt/\* 5.6.1

数据库设计工具:

dev-db/mysql-workbench 6.3.4-r2

数据库:

dev-db/mysql 5.6.33

操作系统:

Linux EVA 4.7.4-gentoo-REDEMPTION x86\_64 Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz GenuineIntel

硬件环境:

00:00.0 Host bridge: Intel Corporation Xeon E3-1200 v3/4th Gen Core Processor DRAM Controller (rev 06)   
 00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v3/4th Gen Core Processor PCI Express x16 Controller (rev 06)   
 00:02.0 VGA compatible controller: Intel Corporation 4th Gen Core Processor Integrated Graphics Controller (rev 06)   
 00:03.0 Audio device: Intel Corporation Xeon E3-1200 v3/4th Gen Core Processor HD Audio Controller (rev 06)   
 00:14.0 USB controller: Intel Corporation 8 Series/C220 Series Chipset Family USB xHCI (rev 05)   
 00:16.0 Communication controller: Intel Corporation 8 Series/C220 Series Chipset Family MEI Controller #1 (rev 04)   
 00:1a.0 USB controller: Intel Corporation 8 Series/C220 Series Chipset Family USB EHCI #2 (rev 05)   
 00:1b.0 Audio device: Intel Corporation 8 Series/C220 Series Chipset High Definition Audio Controller (rev 05)   
 00:1c.0 PCI bridge: Intel Corporation 8 Series/C220 Series Chipset Family PCI Express Root Port #1 (rev d5)   
 00:1c.1 PCI bridge: Intel Corporation 8 Series/C220 Series Chipset Family PCI Express Root Port #2 (rev d5)   
 00:1c.2 PCI bridge: Intel Corporation 8 Series/C220 Series Chipset Family PCI Express Root Port #3 (rev d5)   
 00:1c.4 PCI bridge: Intel Corporation 8 Series/C220 Series Chipset Family PCI Express Root Port #5 (rev d5)   
 00:1d.0 USB controller: Intel Corporation 8 Series/C220 Series Chipset Family USB EHCI #1 (rev 05)   
 00:1f.0 ISA bridge: Intel Corporation HM86 Express LPC Controller (rev 05)   
 00:1f.2 SATA controller: Intel Corporation 8 Series/C220 Series Chipset Family 6-port SATA Controller 1 [AHCI mode] (rev 05)   
 00:1f.3 SMBus: Intel Corporation 8 Series/C220 Series Chipset Family SMBus Controller (rev 05)   
 01:00.0 3D controller: NVIDIA Corporation GM107M [GeForce GTX 860M] (rev a2)   
 08:00.0 Network controller: Intel Corporation Wireless 3160 (rev 93)   
 09:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller (rev 10)   
 0a:00.0 Unassigned class [ff00]: Realtek Semiconductor Co., Ltd. RTS5249 PCI Express Card Reader (rev 01)

# 二、数据库规划

## 2.1 任务陈述

居民小区的物业信息管理需要对业主信息,楼栋房屋信息,抄表信息,缴费信息进行管理,管理信息系统及时了解各个环节中信息的变更,有利于提高管理效率.

物业信息管理系统的主要功能有：业主信息的管理,楼栋信息的记录,房屋信息的记录 ,抄表数据的记录,每月账单以及支付情况记录,以及收费标准的管理.

系统开发的总体任务是实现居民小区各种信息的系统化、规范化和自动化.

## 2.2 任务目标

实现居民小区日常管理的正常快速高效的工作

业主信息的管理

抄表数据准确高效的录入

每月账单的准确及时的生成

保证业主每月账单支付情况的准确记录

房屋信息的记录,包括入住状态的记录

楼栋信息的记录

收费标准的更改

月度应收、未收、实收费用报表的准确快速生成

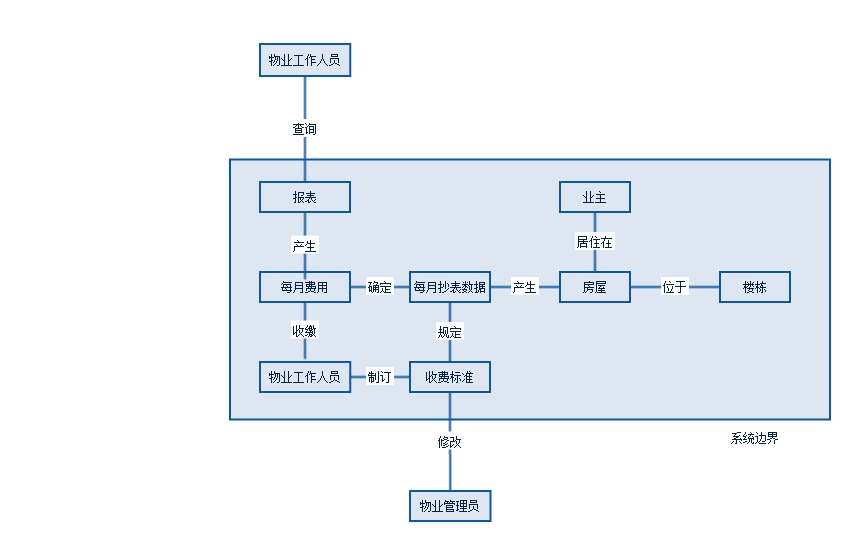
季度应收、未收、实收费用报表的准确快速生成

年度应收、未收、实收费用报表的准确快速生成

交费通知单的打印.

# 三、系统定义

## 3.1 系统边界



## 3.2 用户视图

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **抄表数据录入** | **收取业主费用** | **查看当月应交,未交业主信息** | **月度应收,未收,实收标报表** | **季度度应收,未收,实收标报表** | **年度应收,未收,实收标报表** | **业主信息管理** | **楼栋房屋信息管理** | **物业工作人员的添加删除** | **收费标准的制订** |
| **物业工作人员** | YES | YES | NO | NO | NO | NO | NO | NO | NO | NO |
| **物业管理员** | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES |

# 四、需求分析

## 4.1 用户需求说明

### 4.1.1 数据需求

业主信息,包括身份证号,楼号,房号,姓名,工作单位,联系电话,月房贷数额,余额.

每位业主拥有一套房,房屋信息包括房号,建筑面积,是否入住

每个楼栋属性包括楼号,建成时间,层数

每月每户的水电煤气抄表数据,包括抄表日期,楼号,房号,水表读数,电表读数,煤气表读数.

每月账单,记录每个业主每月的费用,包括业主名,楼号,房号,月份,水电煤气,有线电视费用,供暖费用,物业管理费用,月度房款费用,总和,完成情况.

收费标准,记录各种费用的单价,包括收费标准开始使用的年月份,水费单价(元/立方米),电费单价（元/°）,有线电视单价（元/月）,煤气费用(元/立方米).

管理人员信息,包括管理人员的登录名,密码,姓名

### 4.1.2 事务需求

**数据录入**

A． 录入新业主的详细信息,所住的房屋,个人信息,月房贷数目,物业费余额等

B． 录入每月的抄表数据,水表,电表,煤气表抄表数据

C． 录入收费标准

D． 录入新的物业管理人员

E． 录入楼栋信息

F． 录入房屋信息

**数据更新/删除**

A． 更新/删除业主信息

B． 更新/收费标准

C． 更新/删除物业管理人员信息

**数据查询**

A． 列出所有业主的信息

B． 列出所有业主的带综合费用信息

C． 列出单个业主的信息表

D． 打印收费通知单

E． 打印用户应交未交汇总表

F． 统计报表.统计业主信息月、季和年报表,包括：物业实收费用汇总表、物业应收未收汇总表、应交未交费用业主信息表、月度房款还款名细表、单用户年度应收代款还款表.

## 4.2 系统需求说明

**初始数据库大小**

1. 大约有4个楼栋

2. 大约40个房屋

3. 大约40个业主

4. 大约5个管理人员

5. 大约80条费用记录

6. 大约120条抄表数据

**数据库增长速度**

1. 每月大约有300条费用信息

2. 每年新入住10位业主

3. 每年新离开10位业主

4. 每年新来一名管理人员

5. 每年离开一名管理人员

**记录查找的类型和平均数量**

1. 查询每月费用大概是每月300次,月底是高峰期

2. 查询季度报表大概每季度2次

3. 查询年报表大概每年2次

**网络和共享需求**

1. 多个用户共享数据

**性能**

1. 单个记录查询时间少于1秒,高峰期少于5秒

2. 多个记录查询时间少于5秒,高峰期少于10秒

3. 更新/保存记录时间少于1秒,高峰期少于5秒

**安全性**

1. 数据库必须有口令保护

2. 每个用户分配特定的角色,每个角色具有特定的权限

3. 用户只能在适合他们完成工作需要的窗口中看到需要的数据

4. 传输过程中暴露尽可能少的信息,传输的报文最好进行加密认证

**备份和恢复**

1. 每月1号备份

**用户界面**

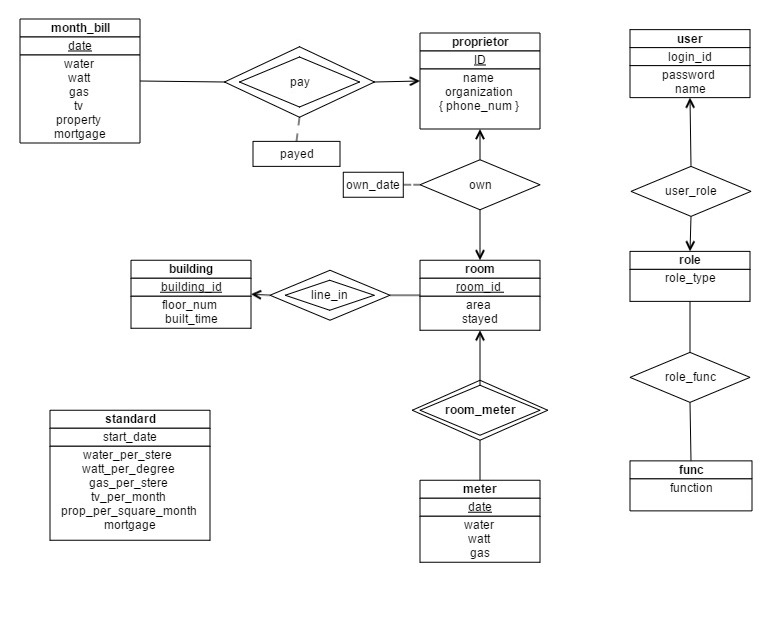
菜单驱动,联机帮助

**法律问题**

对业主和管理人员的信息管理,遵守法律.

# 五、数据库逻辑设计

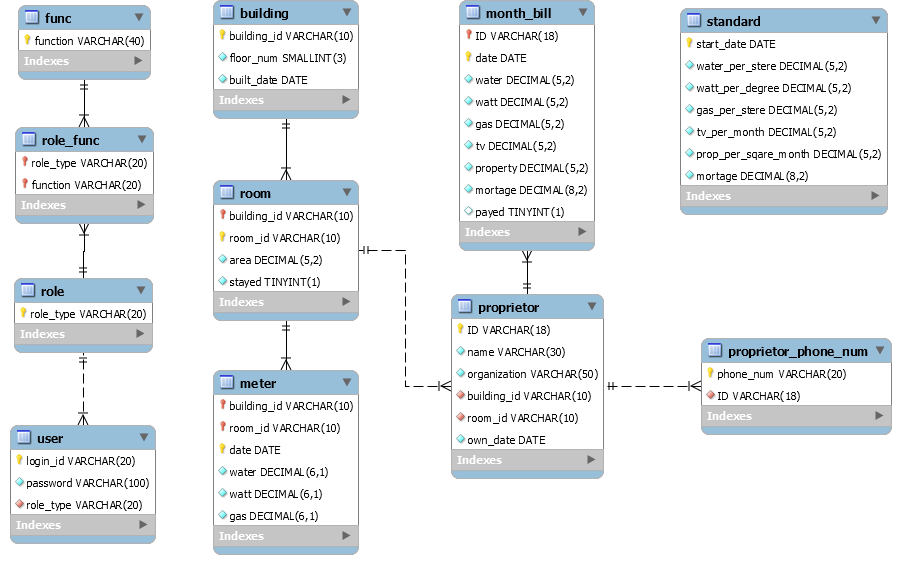
## 5.1 ER图



## 5.2 数据字典

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实体 | 属性 | 描述 | 键 | 类型 | 是否为空 | 是否多值 | 约束 |
| 业主 | 身份证号 | 唯一标识业主 | 主键 | VARCHAR(20) | NO | NO | 数字表示 |
| 姓名 | 业主姓名 |  | VARCHAR(30) | NO | NO | 不能有特殊符号 |
| 工作单位 | 业主工作单位 |  | VARCHAR(50) | NO | NO | 不能有特殊符号 |
| 楼栋号 | 所住楼栋号 |  | VARCHAR(10) | NO | NO | 不能有特殊符号 |
| 入住时间 | 所住房屋号 |  | VARCHAR(10) | NO | NO | yyyy-MM-dd格式 |
| 手机号 | 业主手机号 |  | DATE | NO | YES | 数字表示 |
| 楼栋 | 楼栋号 | 唯一区分楼栋 | 主键 | VARCHAR(10) | NO | NO | 不能有特殊符号 |
| 楼层数 | 楼层的数目 |  | SMALLINT(3) | NO | NO | 数字表示 |
| 建成日期 | 建成的日期 |  | DATE | NO | NO | yyyy-MM-dd格式 |
| 房屋 | 楼栋号 | 房屋所在楼栋 | 主键 | VARCHAR(10) | NO | NO | yyyy-MM-dd格式 |
| 房屋号 | 房屋号 | 主键 | VARCHAR(10) | NO | NO | yyyy-MM-dd格式 |
| 面积 | 面积 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 是否入住 | 入住情况 |  | BOOLEAN | NO | NO | true或false |
| 账单 | 业主身份证号 | 标识业主 | 主键 | VARCHAR(18) | NO | NO | 数字表示 |
| 账单日期 | 账单生成日期 | 主键 | DATE | NO | NO | yyyy-MM-dd格式 |
| 水费 | 当月水费 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 电费 | 当月电费 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 燃气费 | 当月燃气费 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 电视费 | 当月电视费 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 物业费 | 当月物业费 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 月房贷 | 当月房贷 |  | NUMERIC(8,2) | NO | NO | 数字表示 |
| 支付情况 | 是否支付 |  | BOOLEAN | NO | NO | true或false |
| 抄表数据 | 楼栋号 | 抄表楼栋号 | 主键 | VARCHAR(18) | NO | NO | 不能有特殊符号 |
| 房屋号 | 抄表房屋号 | 主键 | VARCHAR(10) | NO | NO | 不能有特殊符号 |
| 抄表日期 | 抄表的日期 | 主键 | DATE | NO | NO | yyyy-MM-dd格式 |
| 水表读数 | 水表的读数 |  | NUMERIC(6,1) | NO | NO | 数字表示 |
| 电表读数 | 电表的读数 |  | NUMERIC(6,1) | NO | NO | 数字表示 |
| 气表读数 | 煤气表的读数 |  | NUMERIC(6,1) | NO | NO | 数字表示 |
| 收费标准 | 起效日期 | 起效的日期 | 主键 | DATE | NO | NO | yyyy-MM-dd格式 |
| 水费单价 | 每立方米水价 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 电费单价 | 每度电价 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 燃气费单价 | 每立方米气价 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 电视费单价 | 每月电视费用 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 物业费单价 | 每月平方价格 |  | NUMERIC(5,2) | NO | NO | 数字表示 |
| 月房贷 | 每月房贷数目 |  | NUMERIC(8,2) | NO | NO | 数字表示 |
| 用户 | 登陆id | 唯一标识用户 | 主键 | VARCHAR(20) | NO | NO | 不能有特殊符号 |
| 密码 | 登陆密码 |  | VARCHAR(100) | NO | NO | Md5串 |
| 角色类型 | 担任的角色 |  | VARCHAR(20) | NO | NO | CHARGER或者MANAGER |
| 角色 | 角色类型 | 各种角色 | 主键 | VARCHAR(20) | NO | NO | CHARGER或者MANAGER |
| 功能 | 功能类型 | 各种功能类型 | 主键 | VARCHAR(20) | NO | NO | 几个常量之一 |

## 5.3 关系表



-- MySQL dump 10.13 Distrib 5.6.33, for Linux (x86\_64)

--

-- Host: localhost Database: propertydata

-- ------------------------------------------------------

-- Server version 5.6.33-log

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `building`

--

DROP TABLE IF EXISTS `building`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `building` (

`building\_id` varchar(10) NOT NULL COMMENT '楼栋id',

`floor\_num` smallint(3) NOT NULL COMMENT '楼层数目',

`built\_date` date NOT NULL COMMENT '建成时间',

PRIMARY KEY (`building\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `building`

--

LOCK TABLES `building` WRITE;

/\*!40000 ALTER TABLE `building` DISABLE KEYS \*/;

INSERT INTO `building` VALUES ('1#',5,'2016-01-01'),('2#',5,'2016-02-01'),('3#',6,'2016-03-01'),('4#',6,'2016-04-01');

/\*!40000 ALTER TABLE `building` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `func`

--

DROP TABLE IF EXISTS `func`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `func` (

`function` varchar(40) NOT NULL COMMENT '功能',

PRIMARY KEY (`function`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `func`

--

LOCK TABLES `func` WRITE;

/\*!40000 ALTER TABLE `func` DISABLE KEYS \*/;

INSERT INTO `func` VALUES ('ADD\_DEL\_BUILDING'),('ADD\_DEL\_CHARGER'),('ADD\_DEL\_PROPRIETOR'),('ADD\_DEL\_ROOM'),('CHARGE\_FEE'),('RECORD\_METER'),('REPORT\_FORMS'),('SET\_STANDARD');

/\*!40000 ALTER TABLE `func` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `meter`

--

DROP TABLE IF EXISTS `meter`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `meter` (

`building\_id` varchar(10) NOT NULL COMMENT '楼栋id',

`room\_id` varchar(10) NOT NULL COMMENT '房屋id',

`date` date NOT NULL COMMENT '抄表时间',

`water` decimal(6,1) NOT NULL COMMENT '水表读数',

`watt` decimal(6,1) NOT NULL COMMENT '电表读数',

`gas` decimal(6,1) NOT NULL COMMENT '煤气表读数',

PRIMARY KEY (`room\_id`,`building\_id`,`date`),

KEY `building\_id` (`building\_id`,`room\_id`),

CONSTRAINT `meter\_ibfk\_1` FOREIGN KEY (`building\_id`, `room\_id`) REFERENCES `room` (`building\_id`, `room\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `meter`

--

LOCK TABLES `meter` WRITE;

/\*!40000 ALTER TABLE `meter` DISABLE KEYS \*/;

INSERT INTO `meter` VALUES ('1#','101','2016-07-01',1.0,2.0,3.0),('1#','101','2016-08-02',2.0,4.0,3.0);

/\*!40000 ALTER TABLE `meter` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `month\_bill`

--

DROP TABLE IF EXISTS `month\_bill`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `month\_bill` (

`ID` varchar(18) NOT NULL COMMENT '业主身份证号',

`date` date NOT NULL COMMENT '月账单产生日期',

`water` decimal(5,2) NOT NULL COMMENT '月水费数额',

`watt` decimal(5,2) NOT NULL COMMENT '月电费数额',

`gas` decimal(5,2) NOT NULL COMMENT '月煤气费数额',

`tv` decimal(5,2) NOT NULL,

`property` decimal(5,2) NOT NULL COMMENT '月物业费数额',

`mortage` decimal(8,2) NOT NULL COMMENT '月房贷数额',

`payed` varchar(1) DEFAULT NULL,

PRIMARY KEY (`ID`,`date`),

CONSTRAINT `month\_bill\_ibfk\_1` FOREIGN KEY (`ID`) REFERENCES `proprietor` (`ID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `month\_bill`

--

LOCK TABLES `month\_bill` WRITE;

/\*!40000 ALTER TABLE `month\_bill` DISABLE KEYS \*/;

INSERT INTO `month\_bill` VALUES ('610115199501236279','2016-08-02',2.00,4.00,0.00,18.00,100.00,6000.00,'0');

/\*!40000 ALTER TABLE `month\_bill` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `proprietor`

--

DROP TABLE IF EXISTS `proprietor`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `proprietor` (

`ID` varchar(18) NOT NULL COMMENT '业主身份证号',

`name` varchar(30) NOT NULL COMMENT '业主姓名',

`organization` varchar(50) NOT NULL DEFAULT 'none' COMMENT '业主工作单位',

`building\_id` varchar(10) NOT NULL COMMENT '楼栋id',

`room\_id` varchar(10) NOT NULL COMMENT '房屋id',

`own\_date` date NOT NULL COMMENT '入住时间',

PRIMARY KEY (`ID`),

KEY `building\_id` (`building\_id`,`room\_id`),

CONSTRAINT `proprietor\_ibfk\_1` FOREIGN KEY (`building\_id`, `room\_id`) REFERENCES `room` (`building\_id`, `room\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `proprietor`

--

LOCK TABLES `proprietor` WRITE;

/\*!40000 ALTER TABLE `proprietor` DISABLE KEYS \*/;

INSERT INTO `proprietor` VALUES ('610115199501236279','elvis','SDU','1#','101','2016-07-01');

/\*!40000 ALTER TABLE `proprietor` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `proprietor\_phone\_num`

--

DROP TABLE IF EXISTS `proprietor\_phone\_num`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `proprietor\_phone\_num` (

`phone\_num` varchar(20) NOT NULL COMMENT '业主电话号码',

`ID` varchar(18) NOT NULL COMMENT '业主身份证号',

PRIMARY KEY (`phone\_num`),

KEY `ID` (`ID`),

CONSTRAINT `proprietor\_phone\_num\_ibfk\_1` FOREIGN KEY (`ID`) REFERENCES `proprietor` (`ID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `proprietor\_phone\_num`

--

LOCK TABLES `proprietor\_phone\_num` WRITE;

/\*!40000 ALTER TABLE `proprietor\_phone\_num` DISABLE KEYS \*/;

INSERT INTO `proprietor\_phone\_num` VALUES ('13335313557','610115199501236279'),('17865192687','610115199501236279');

/\*!40000 ALTER TABLE `proprietor\_phone\_num` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `role`

--

DROP TABLE IF EXISTS `role`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `role` (

`role\_type` varchar(20) NOT NULL COMMENT '角色类型',

PRIMARY KEY (`role\_type`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `role`

--

LOCK TABLES `role` WRITE;

/\*!40000 ALTER TABLE `role` DISABLE KEYS \*/;

INSERT INTO `role` VALUES ('CHARGER'),('MANAGER');

/\*!40000 ALTER TABLE `role` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `role\_func`

--

DROP TABLE IF EXISTS `role\_func`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `role\_func` (

`role\_type` varchar(20) NOT NULL COMMENT '角色类型',

`function` varchar(20) NOT NULL COMMENT '功能',

PRIMARY KEY (`role\_type`,`function`),

KEY `function` (`function`),

CONSTRAINT `role\_func\_ibfk\_1` FOREIGN KEY (`role\_type`) REFERENCES `role` (`role\_type`),

CONSTRAINT `role\_func\_ibfk\_2` FOREIGN KEY (`function`) REFERENCES `func` (`function`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `role\_func`

--

LOCK TABLES `role\_func` WRITE;

/\*!40000 ALTER TABLE `role\_func` DISABLE KEYS \*/;

INSERT INTO `role\_func` VALUES ('MANAGER','ADD\_DEL\_BUILDING'),('MANAGER','ADD\_DEL\_CHARGER'),('MANAGER','ADD\_DEL\_PROPRIETOR'),('MANAGER','ADD\_DEL\_ROOM'),('CHARGER','CHARGE\_FEE'),('MANAGER','CHARGE\_FEE'),('CHARGER','RECORD\_METER'),('MANAGER','RECORD\_METER'),('MANAGER','REPORT\_FORMS'),('MANAGER','SET\_STANDARD');

/\*!40000 ALTER TABLE `role\_func` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `room`

--

DROP TABLE IF EXISTS `room`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `room` (

`building\_id` varchar(10) NOT NULL COMMENT '楼栋id',

`room\_id` varchar(10) NOT NULL COMMENT '每栋楼上房屋id',

`area` decimal(5,2) NOT NULL COMMENT '房屋面积',

`stayed` varchar(1) NOT NULL DEFAULT '0',

PRIMARY KEY (`building\_id`,`room\_id`),

CONSTRAINT `room\_ibfk\_1` FOREIGN KEY (`building\_id`) REFERENCES `building` (`building\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `room`

--

LOCK TABLES `room` WRITE;

/\*!40000 ALTER TABLE `room` DISABLE KEYS \*/;

INSERT INTO `room` VALUES ('1#','101',100.00,'1'),('1#','102',100.00,'0'),('1#','201',100.00,'0'),('1#','202',100.00,'0'),('1#','301',100.00,'0'),('1#','302',100.00,'0'),('1#','401',100.00,'0'),('1#','402',100.00,'0'),('1#','501',100.00,'0'),('1#','502',100.00,'0'),('2#','101',100.00,'0'),('2#','102',100.00,'0'),('2#','201',100.00,'0'),('2#','202',100.00,'0'),('2#','301',100.00,'0'),('2#','302',100.00,'0'),('2#','401',100.00,'0'),('2#','402',100.00,'0'),('2#','501',100.00,'0'),('2#','502',100.00,'0'),('3#','101',120.00,'0'),('3#','102',120.00,'0'),('3#','201',120.00,'0'),('3#','202',120.00,'0'),('3#','301',120.00,'0'),('3#','302',120.00,'0'),('3#','401',120.00,'0'),('3#','402',120.00,'0'),('3#','501',120.00,'0'),('3#','502',120.00,'0'),('3#','601',120.00,'0'),('3#','602',120.00,'0'),('4#','101',120.00,'0'),('4#','102',120.00,'0'),('4#','201',120.00,'0'),('4#','202',120.00,'0'),('4#','301',120.00,'0'),('4#','302',120.00,'0'),('4#','401',120.00,'0'),('4#','402',120.00,'0'),('4#','501',120.00,'0'),('4#','502',120.00,'0'),('4#','601',120.00,'0'),('4#','602',120.00,'0');

/\*!40000 ALTER TABLE `room` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `standard`

--

DROP TABLE IF EXISTS `standard`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `standard` (

`start\_date` date NOT NULL COMMENT '标准启动时间',

`water\_per\_stere` decimal(5,2) NOT NULL DEFAULT '0.00' COMMENT '每立方米水价(元)',

`watt\_per\_degree` decimal(5,2) NOT NULL DEFAULT '0.00' COMMENT '每度电价格(元)',

`gas\_per\_stere` decimal(5,2) NOT NULL DEFAULT '0.00' COMMENT '每立方米煤气价(元)',

`tv\_per\_month` decimal(5,2) NOT NULL DEFAULT '0.00' COMMENT '每月有线电视价格',

`prop\_per\_sqare\_month` decimal(5,2) NOT NULL DEFAULT '0.00' COMMENT '每平米每月物业费',

`mortage` decimal(8,2) NOT NULL DEFAULT '0.00',

PRIMARY KEY (`start\_date`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `standard`

--

LOCK TABLES `standard` WRITE;

/\*!40000 ALTER TABLE `standard` DISABLE KEYS \*/;

INSERT INTO `standard` VALUES ('2016-01-20',2.00,2.00,2.50,18.00,1.00,6000.00),('2016-10-01',2.00,2.50,3.00,18.00,0.50,6000.00),('2017-01-01',4.00,4.00,4.00,4.00,4.00,8000.00);

/\*!40000 ALTER TABLE `standard` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `user`

--

DROP TABLE IF EXISTS `user`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `user` (

`login\_id` varchar(20) NOT NULL COMMENT '系统用户登录名',

`password` varchar(100) NOT NULL COMMENT '登录密码哈希密文',

`role\_type` varchar(20) NOT NULL,

PRIMARY KEY (`login\_id`),

KEY `role\_type` (`role\_type`),

CONSTRAINT `user\_ibfk\_1` FOREIGN KEY (`role\_type`) REFERENCES `role` (`role\_type`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `user`

--

LOCK TABLES `user` WRITE;

/\*!40000 ALTER TABLE `user` DISABLE KEYS \*/;

INSERT INTO `user` VALUES ('elvis','37ed2c07cda30a82f8442c52944475ea','MANAGER'),('manager','96e79218965eb72c92a549dd5a330112','MANAGER'),('operator','96e79218965eb72c92a549dd5a330112','CHARGER');

/\*!40000 ALTER TABLE `user` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2016-10-09 21:39:48

# 六、数据库物理设计

## 6.1 索引

|  |  |  |
| --- | --- | --- |
| 表名 | 主键 | 外键 |
| proprietor | ID | BUILDING\_ID,  ROOM\_ID |
| meter | BUILDING\_ID,  ROOM\_ID,  DATE | BUILDING\_ID,  ROOM\_ID |
| month\_bill | ID,  DATE | ID |
| room | BUILDING\_ID,  ROOM\_ID, |  |
| standard | START\_DATE |  |
| proprietor\_phone\_num | PHONE\_NUM | ID |
| building | BUILDING\_ID |  |
| func | FUNCTION |  |
| role\_func | ROLE,  FUNCTION | FUNCTION |
| role | ROLE\_TYPE |  |
| user | LOGIN\_ID | ROLE\_TYPE |

**为每张表的主键创建索引,以提高查询效率**

## 6.2 安全机制

**系统安全**

1. 为用户的输入做边界检查,检查用户的输入,做好充足的异常处理
2. 系统登陆时凭借用户名和口令进入,并且此用户为数据库服务器分配给本应用系统的数据库,只提供最小权限,保证数据库的安全.

**数据安全**

1. 根据用户身份,将应用系统的用户分为不同的角色,不同角色权限不同,所见到的界面也不同,只有管理员才能进行一些信息的修改,从而保证数据库不被随意修改,保证数据安全.
2. 系统共包含两种角色,物业工作人员,物业管理员,权限分配如下
3. 物业工作人员:录入抄表数据,收缴业主费用,查看所有业主的已交,未交,全部账单；打印缴费通知单.
4. 物业管理员: 录入抄表数据,收缴业主费用,查看所有业主的已交,未交,全部账单;打印缴费通知单；查看月度应收、未收、已收报表; 查看季度应收、未收、已收报表; 查看年度应收、未收、已收报表;添加业主；删除业主；查看全部业主信息；查看全部房屋信息；添加收费人员；删除收费人员；设置收费标准.

## 6.3 其他

month\_bill表中water,watt,gas,tv,property,mortage都是派生数据类型,可以由其他表格产生,但是引入冗余之后,月账单的查询效率大大提升.

# 七、应用程序设计

## 7.1 功能模块

系统包含两个角色：物业工作人员,物业管理员.

1. 物业工作人员：

录入抄表数据

收缴业主费用,打印缴费通知单;

查看所有业主的已交,未交,全部账单；

1. 物业管理员：

录入抄表数据;

收缴业主费用,打印缴费通知单；

查看所有业主的已交,未交,全部账单;

查看月度应收、未收、已收报表;

查看季度应收、未收、已收报表;

查看年度应收、未收、已收报表;

添加业主；删除业主；

查看全部业主信息；

查看全部房屋信息；

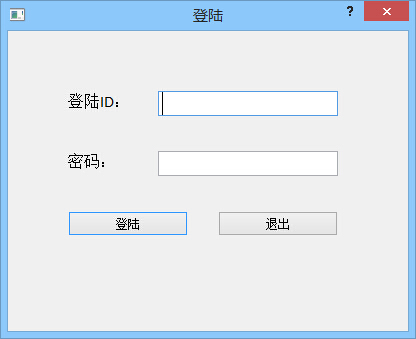
添加收费人员；

删除收费人员；

设置收费标准.

## 7.2 界面设计

登录界面



抄表数据录入

业主缴费



查询当月交费情况



月度报表



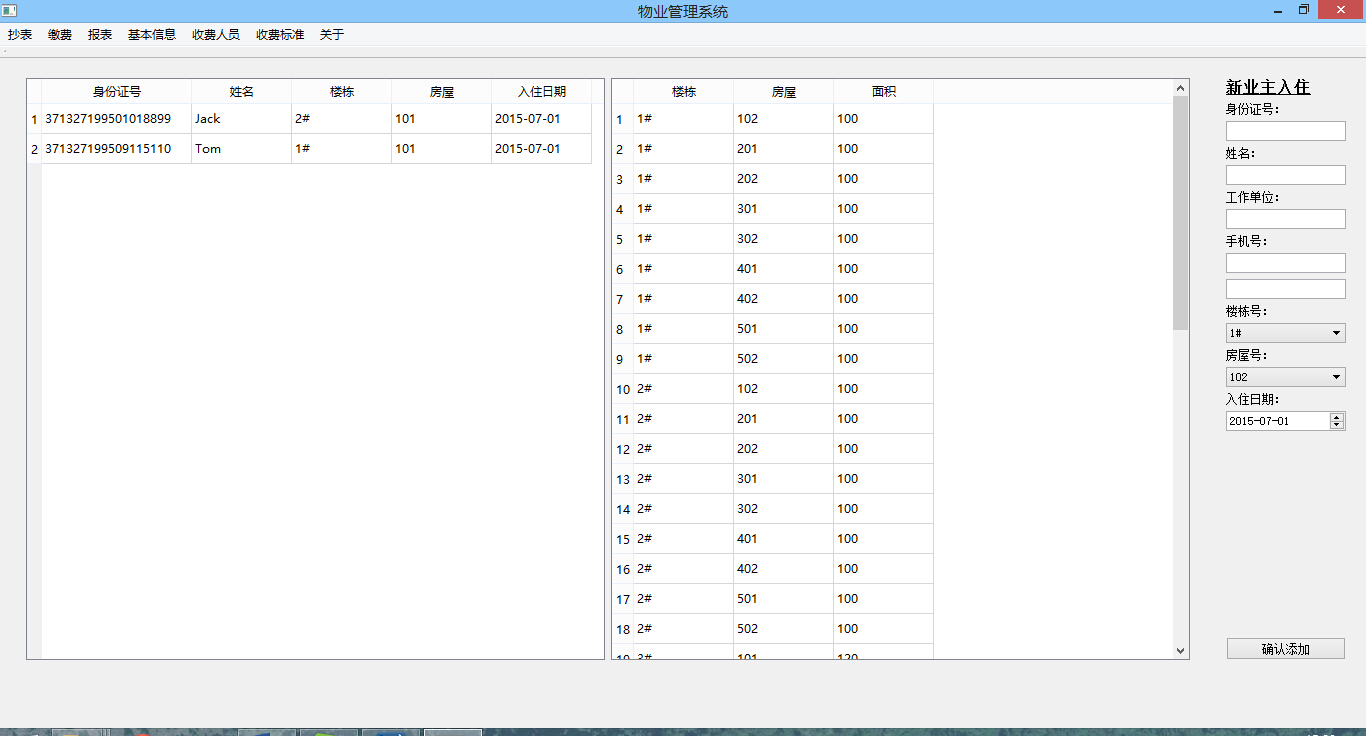
季度报表



年度报表



增加业主



删除业主



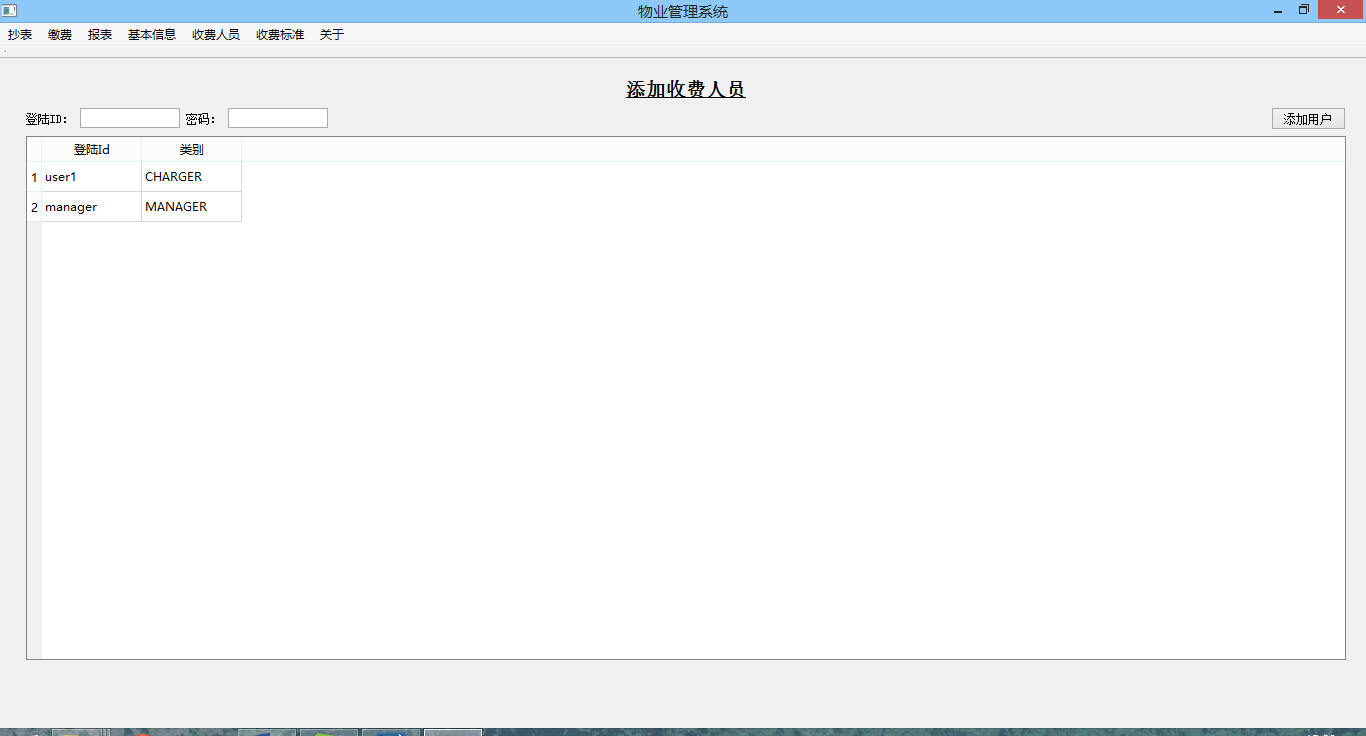
查看全部业主信息



查看全部房屋信息



添加收费人员



删除收费人员



设定新的收费标准



## 7.3 事务设计

**增加业主**

void AddProprietorWidget::on\_pushButton\_clicked()

{

if(ui->proprietorLineEdit->text().length()!=18 && ui->proprietorLineEdit->text().length()!=15)

{

ui->proprietorLineEdit->clear();

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("身份证号错误"),QMessageBox::*Yes*);

return;

}

if(ui->nameLineEdit->text().length()==0)

{

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("姓名不能为空"),QMessageBox::*Yes*);

return;

}

if(ui->organizationLineEdit->text().length()==0)

{

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("工作单位不能为空"),QMessageBox::*Yes*);

return;

}

if(ui->phone1LineEdit->text().length()==0)

{

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("电话号码不能为空"),QMessageBox::*Yes*);

return;

}

// http from 7

int index = 7 , postSize = 0 , sequence = 0 , size = 0 , rows = 0;

QList<QByteArray> info;

QString postSQL = "";

QString ID = ui->proprietorLineEdit->text();

QString name = ui->nameLineEdit->text();

QString organization = ui->organizationLineEdit->text();

QString building\_id = ui->buildingComboBox->currentText();

QString room\_id = ui->roomComboBox->currentText();

QString own\_date = ui->ownDateDateTimeEdit->text();

bool res ;

postSize = 6;

postSQL = "insert into proprietor values(?,?,?,?,?,?)";

list[index].clear();

list[index].setCommit(false);

list[index].setFlag(true);

list[index].setSync(true);

list[index].setLength(postSize);

list[index].addParameter(ID);

list[index].addParameter(name);

list[index].addParameter(organization);

list[index].addParameter(building\_id);

list[index].addParameter(room\_id);

list[index].addParameter(own\_date);

list[index].postRequest(postSQL);

info = list[index].getInfo().split('\n');

size = info[0].toInt();

rows = info[1].toInt();

if(list[index].getFlag())

{

res = (QString(info[2]).compare("true") == 0);

}

for(int i = 0 ; i < size ; i ++)

{

for(int j = 0 ; j < rows ; j++)

{

sequence = i\*rows + j + 2;

}

}

index++;

postSize = 2;

postSQL = "insert into proprietor\_phone\_num values(?,?)";

list[index].clear();

list[index].setCommit(false);

list[index].setFlag(true);

list[index].setSync(true);

list[index].setLength(postSize);

list[index].addParameter(ui->phone1LineEdit->text());

list[index].addParameter(ui->proprietorLineEdit->text());

list[index].postRequest(postSQL);

info = list[index].getInfo().split('\n');

size = info[0].toInt();

rows = info[1].toInt();

if(list[index].getFlag())

{

res &= (QString(info[2]).compare("true") == 0);

}

index++;

if(ui->phone2Line->text().length() > 1)

{

postSize = 2;

postSQL = "insert into proprietor\_phone\_num values(?,?)";

list[index].clear();

list[index].setCommit(false);

list[index].setFlag(true);

list[index].setSync(true);

list[index].setLength(postSize);

list[index].addParameter(ui->phone2Line->text());

list[index].addParameter(ui->proprietorLineEdit->text());

list[index].postRequest(postSQL);

info = list[index].getInfo().split('\n');

size = info[0].toInt();

rows = info[1].toInt();

if(list[index].getFlag())

{

res &= (QString(info[2]).compare("true") == 0);

}

index++;

}

postSize = 2;

postSQL = "update room set stayed = true where building\_id=? and room\_id=?";

list[index].clear();

list[index].setCommit(false);

list[index].setFlag(true);

list[index].setSync(true);

list[index].setLength(postSize);

list[index].addParameter(building\_id);

list[index].addParameter(room\_id);

list[index].postRequest(postSQL);

info = list[index].getInfo().split('\n');

size = info[0].toInt();

rows = info[1].toInt();

if(list[index].getFlag())

{

res &= (QString(info[2]).compare("true") == 0);

}

index++;

if(!res)

{

HTTP temp;

temp.setLength(0);

temp.setRollback(true);

temp.setSync(true);

temp.postRequest("");

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("操作失败"),QMessageBox::*Yes*);

return;

}

else

{

HTTP temp;

temp.setLength(0);

temp.setCommit(true);

temp.setSync(true);

temp.postRequest("");

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("操作成功"),QMessageBox::*Yes*);

}

postSize = 1;

postSQL = "select room\_id from room where building\_id=? and stayed = false";

ui->roomComboBox->clear();

list[index].clear();

list[index].setCommit(false);

list[index].setFlag(false);

list[index].setSync(true);

list[index].setLength(postSize);

list[index].addParameter(ui->buildingComboBox->currentText());

list[index].postRequest(postSQL);

info = list[index].getInfo().split('\n');

size = info[0].toInt();

rows = info[1].toInt();

for(int i = 0 ; i < size ; i ++)

{

for(int j = 0 ; j < rows ; j++)

{

sequence = i\*rows + j + 2;

if(j == 0)

{

QString id = QString(info[sequence]);

ui->roomComboBox->addItem(id);

}

}

}

index++;

postSize = 1;

postSQL = "select id,name, building\_id, room\_id, own\_date from proprietor where building\_id=? ";

ui->proprietorTableView->model()->deleteLater();

old->deleteLater();

QStandardItemModel\* pmodel = new QStandardItemModel();

pmodel->setColumnCount(5);

pmodel->*setHeaderData*(0,Qt::*Horizontal*,QString::fromUtf8("身份证号"));

pmodel->*setHeaderData*(1,Qt::*Horizontal*,QString::fromUtf8("姓名"));

pmodel->*setHeaderData*(2,Qt::*Horizontal*,QString::fromUtf8("楼栋"));

pmodel->*setHeaderData*(3,Qt::*Horizontal*,QString::fromUtf8("房屋"));

pmodel->*setHeaderData*(4,Qt::*Horizontal*,QString::fromUtf8("入住日期"));

int rowNum = 0;

list[index].clear();

list[index].setCommit(false);

list[index].setFlag(false);

list[index].setSync(true);

list[index].setLength(postSize);

list[index].addParameter(ui->buildingComboBox->currentText());

list[index].postRequest(postSQL);

info = list[index].getInfo().split('\n');

size = info[0].toInt();

rows = info[1].toInt();

for(int i = 0 ; i < size ; i ++)

{

rowNum = pmodel->*rowCount*();

for(int j = 0 ; j < rows ; j++)

{

sequence = i\*rows + j + 2;

pmodel->setItem(rowNum,j,new QStandardItem(QString(info[sequence])));

}

}

ui->proprietorTableView->*setModel*(pmodel);

ui->proprietorTableView->horizontalHeader()->setDefaultAlignment(Qt::*AlignCenter*);

ui->proprietorTableView->setColumnWidth(0,150);

index++;

postSize = 1;

postSQL = "select building\_id, room\_id, area from room where stayed = false and building\_id=?";

ui->tableView->model()->deleteLater();

QStandardItemModel\* model = new QStandardItemModel();

model->setColumnCount(3);

model->*setHeaderData*(0,Qt::*Horizontal*,QString::fromUtf8("楼栋"));

model->*setHeaderData*(1,Qt::*Horizontal*,QString::fromUtf8("房屋"));

model->*setHeaderData*(2,Qt::*Horizontal*,QString::fromUtf8("面积"));

ui->tableView->*setModel*(model);

ui->tableView->horizontalHeader()->setDefaultAlignment(Qt::*AlignCenter*);

ui->tableView->setColumnWidth(0,100);

list[index].clear();

list[index].setCommit(false);

list[index].setFlag(false);

list[index].setTransaction(false);

list[index].setSync(true);

list[index].setLength(postSize);

list[index].addParameter(ui->buildingComboBox->currentText());

list[index].postRequest(postSQL);

info = list[index].getInfo().split('\n');

size = info[0].toInt();

rows = info[1].toInt();

for(int i = 0 ; i < size ; i ++)

{

rowNum = model->*rowCount*();

for(int j = 0 ; j < rows ; j++)

{

sequence = i\*rows + j + 2;

model->setItem(rowNum,j,new QStandardItem(QString(info[sequence])));

}

}

index++;

}

**删除业主**

void DelProprietorWidget::on\_pushButton\_clicked()

{

// start from 8

QString build , room , id ;

build = ui->buildComboBox->currentText();

room = ui->roomComboBox->currentText();

id = ui->proprietorIdLabel->text();

HTTP temp;

bool res = true;

temp.clear();

temp.setSync(true);

temp.setTransaction(true);

temp.setLength(0);

temp.postRequest("");

temp.clear();

temp.setTransaction(false);

temp.setFlag(true);

temp.setSync(true);

temp.setRollback(false);

temp.setCommit(false);

temp.setLength(1);

temp.addParameter(id);

temp.postRequest("delete from proprietor\_phone\_num where id=?");

{

QList<QByteArray> info = temp.getInfo().split('\n');

res &= (QString(info[2]).compare("true") == 0);

}

temp.clear();

temp.setTransaction(false);

temp.setFlag(true);

temp.setSync(true);

temp.setRollback(false);

temp.setCommit(false);

temp.setLength(1);

temp.addParameter(id);

temp.postRequest("delete from month\_bill where id=?");

{

QList<QByteArray> info = temp.getInfo().split('\n');

res &= (QString(info[2]).compare("true") == 0);

}

temp.clear();

temp.setTransaction(false);

temp.setFlag(true);

temp.setSync(true);

temp.setRollback(false);

temp.setCommit(false);

temp.setLength(2);

temp.addParameter(build);

temp.addParameter(room);

temp.postRequest("delete from proprietor where building\_id=? and room\_id = ?");

{

QList<QByteArray> info = temp.getInfo().split('\n');

res &= (QString(info[2]).compare("true") == 0);

}

temp.clear();

temp.setTransaction(false);

temp.setFlag(true);

temp.setSync(true);

temp.setRollback(false);

temp.setCommit(false);

temp.setLength(2);

temp.addParameter(build);

temp.addParameter(room);

temp.postRequest("update room set stayed=false where building\_id=? and room\_id = ?");

{

QList<QByteArray> info = temp.getInfo().split('\n');

res &= (QString(info[2]).compare("true") == 0);

}

if(!res)

{

temp.clear();

temp.setTransaction(false);

temp.setFlag(false);

temp.setSync(true);

temp.setRollback(true);

temp.setCommit(false);

temp.setLength(0);

temp.postRequest("");

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("操作失败"),QMessageBox::*Yes*);

return;

}

else

{

temp.clear();

temp.setTransaction(false);

temp.setFlag(false);

temp.setSync(true);

temp.setRollback(false);

temp.setCommit(true);

temp.setLength(0);

temp.postRequest("");

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("操作成功"),QMessageBox::*Yes*);

}

ui->roomComboBox->clear();

list[11].clear();

list[11].setCommit(false);

list[11].setFlag(false);

list[11].setLength(1);

list[11].addParameter(ui->buildComboBox->currentText());

list[11].postRequest("select room\_id from room where building\_id=? and stayed = true");

QObject::connect(list[11].getReply() , &QNetworkReply::finished , [&]()

{

QList<QByteArray> info = list[11].getInfo().split('\n');

int size = info[0].toInt();

int rows = info[1].toInt();

int sequence;

for(int i = 0 ; i < size ; i ++)

{

for(int j = 0 ; j < rows ; j++)

{

sequence = i\*rows + j + 2;

ui->roomComboBox->addItem(QString(info[sequence]));

}

}

{

list[13].clear();

list[13].setCommit(false);

list[13].setFlag(false);

list[13].setLength(2);

list[13].addParameter(ui->buildComboBox->currentText());

list[13].addParameter(ui->roomComboBox->currentText());

list[13].postRequest("select id,name from proprietor where building\_id=? and room\_id = ?");

QObject::connect(list[13].getReply() , &QNetworkReply::finished , [&]()

{

QList<QByteArray> info = list[13].getInfo().split('\n');

int size = info[0].toInt();

int rows = info[1].toInt();

int sequence;

for(int i = 0 ; i < size ; i ++)

{

for(int j = 0 ; j < rows ; j++)

{

sequence = i\*rows + j + 2;

if(j == 0)

{

ui->proprietorIdLabel->setText(QString(info[sequence]));

}

else if(j == 1)

{

ui->proprietorLabel->setText(QString(info[sequence]));

}

}

}

});

}

});

list[12].clear();

list[12].setCommit(false);

list[12].setFlag(false);

list[12].setLength(1);

list[12].addParameter(ui->buildComboBox->currentText());

list[12].postRequest("select id,name,organization,building\_id, room\_id,own\_date from proprietor where building\_id=?");

QObject::connect(list[12].getReply() , &QNetworkReply::finished , [&]()

{

ui->tableView->model()->deleteLater();

QStandardItemModel\* pmodel = new QStandardItemModel();

pmodel->setColumnCount(6);

pmodel->*setHeaderData*(0,Qt::*Horizontal*,QString::fromUtf8("身份证号"));

pmodel->*setHeaderData*(1,Qt::*Horizontal*,QString::fromUtf8("姓名"));

pmodel->*setHeaderData*(2,Qt::*Horizontal*,QString::fromUtf8("工作单位"));

pmodel->*setHeaderData*(3,Qt::*Horizontal*,QString::fromUtf8("楼栋"));

pmodel->*setHeaderData*(4,Qt::*Horizontal*,QString::fromUtf8("房屋"));

pmodel->*setHeaderData*(5,Qt::*Horizontal*,QString::fromUtf8("入住日期"));

ui->tableView->*setModel*(pmodel);

//表头信息显示居左

ui->tableView->horizontalHeader()->setDefaultAlignment(Qt::*AlignCenter*);

ui->tableView->setColumnWidth(0,150);

int rowNum = 0;

QList<QByteArray> info = list[12].getInfo().split('\n');

int size = info[0].toInt();

int rows = info[1].toInt();

int sequence;

for(int i = 0 ; i < size ; i ++)

{

rowNum = pmodel->*rowCount*();

for(int j = 0 ; j < rows ; j++)

{

sequence = i\*rows + j + 2;

pmodel->setItem(rowNum,j,new QStandardItem(QString(info[sequence])));

}

}

});

}

**月账单的生成**

void RecordMeterWidget::on\_confirmPushButton\_clicked()

{

// QSqlQuery query;

HTTP temp;

double water\_per\_stere;

double watt\_per\_degree;

double gas\_per\_stere;

double tv\_per\_month;

double prop\_per\_sqare\_month;

double mortage;

//get standard

temp.clear();

temp.setCommit(false);

temp.setFlag(true);

temp.setRollback(false);

temp.setTransaction(false);

temp.setSync(true);

temp.setLength(0);

temp.postRequest(QString("select water\_per\_stere , watt\_per\_degree , gas\_per\_stere ,tv\_per\_month , prop\_per\_sqare\_month , mortage from standard where start\_date = (select max(start\_date) from standard where start\_date<='%1')").arg(ui->dateEdit->text()));

{

QList<QByteArray> info = temp.getInfo().split('\n');

int size = info[0].toInt();

int rows = info[1].toInt();

int sequence;

if(temp.getFlag())

{

bool b = (QString(info[2]).compare("true") == 0);

if(!b)

{

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("出现错误"),QMessageBox::*Yes*);

return;

}

}

for(int i = 0 ; i < size ; i ++)

{

for(int j = 0 ; j < rows; j++)

{

sequence = i\*rows + j + 3;

switch (j)

{

case 0:

water\_per\_stere = info[sequence].toDouble();

break;

case 1:

watt\_per\_degree = info[sequence].toDouble();

break;

case 2:

gas\_per\_stere = info[sequence].toDouble();

break;

case 3:

tv\_per\_month = info[sequence].toDouble();

break;

case 4:

prop\_per\_sqare\_month = info[sequence].toDouble();

break;

case 5:

mortage = info[sequence].toDouble();

break;

default:

break;

}

}

}

}

//number of record to be inserted

int rowCount = model->*rowCount*();

//current record info

QString building\_id;

QString room\_id;

QString date;

double water;

double watt;

double gas;

//meter of last month

double last\_water = 0;

double last\_watt = 0;

double last\_gas = 0;

//fee of this month

double water\_fee ;

double watt\_fee;

double gas\_fee;

double tv\_fee;

double property\_fee;

double mortage\_fee;

//for every record in table view to be inserted

for(int i = 0; i < rowCount ; i++)

{

// res of query.exec()

bool res = true;

temp.clear();

temp.setSync(true);

temp.setTransaction(true);

temp.setLength(0);

temp.postRequest("");

// get current meters

building\_id = model->item(i,0)->text();

room\_id = model->item(i,1)->text();

date = model->item(i,2)->text();

water = model->item(i,3)->text().toDouble();

watt = model->item(i,4)->text().toDouble();

gas = model->item(i,5)->text().toDouble();

//get last meters

temp.clear();

temp.setCommit(false);

temp.setFlag(true);

temp.setRollback(false);

temp.setTransaction(false);

temp.setSync(true);

temp.setLength(4);

temp.addParameter(building\_id);

temp.addParameter(room\_id);

temp.addParameter(building\_id);

temp.addParameter(room\_id);

temp.postRequest("select water,watt,gas from meter where building\_id=? and room\_id=? and date = (select max(date) from meter where building\_id=? and room\_id=?)");

{

QList<QByteArray> info = temp.getInfo().split('\n');

int size = info[0].toInt();

int rows = info[1].toInt();

int sequence;

res &= (QString(info[2]).compare("true") == 0);

if(!res)

{

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("获取历史记录失败"),QMessageBox::*Yes*);

continue;

}

for(int i = 0 ; i < size ; i ++)

{

for(int j = 0 ; j < rows ; j++)

{

sequence = i\*rows + j + 3;

switch (j)

{

case 0:

last\_water = info[sequence].toDouble();

break;

case 1:

last\_watt = info[sequence].toDouble();

break;

case 2:

last\_gas = info[sequence].toDouble();

break;

default:

break;

}

}

}

}

temp.clear();

temp.setCommit(false);

temp.setFlag(true);

temp.setRollback(false);

temp.setTransaction(false);

temp.setSync(true);

temp.setLength(6);

temp.addParameter(building\_id);

temp.addParameter(room\_id);

temp.addParameter(date);

temp.addParameter(water);

temp.addParameter(watt);

temp.addParameter(gas);

temp.postRequest("insert into meter values(? , ? , ? , ? , ? , ?)");

{

QList<QByteArray> info = temp.getInfo().split('\n');

res &= (QString(info[2]).compare("true") == 0);

if(!res)

{

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("插入数据失败,键值重复"),QMessageBox::*Yes*);

continue;

}

}

temp.clear();

temp.setCommit(false);

temp.setFlag(false);

temp.setRollback(false);

temp.setTransaction(false);

temp.setSync(true);

temp.setLength(2);

temp.addParameter(building\_id);

temp.addParameter(room\_id);

temp.postRequest("select id , area from proprietor natural join room where building\_id=? and room\_id=?");

QString proprietorId;

double area;

{

QList<QByteArray> info = temp.getInfo().split('\n');

int size = info[0].toInt();

int rows = info[1].toInt();

int sequence;

for(int i = 0 ; i < size ; i ++)

{

for(int j = 0 ; j < rows ; j++)

{

sequence = i\*rows + j + 2;

switch (j)

{

case 0:

proprietorId = info[sequence];

break;

case 1:

area = info[sequence].toDouble();

break;

default:

break;

}

}

}

}

water\_fee = (water - last\_water)\*water\_per\_stere;

watt\_fee = (watt - last\_watt)\*watt\_per\_degree;

gas\_fee = (gas - last\_gas)\*gas\_per\_stere;

tv\_fee = tv\_per\_month;

property\_fee = prop\_per\_sqare\_month\*area;

mortage\_fee = mortage;

temp.clear();

temp.setCommit(false);

temp.setFlag(true);

temp.setRollback(false);

temp.setTransaction(false);

temp.setSync(true);

temp.setLength(8);

temp.addParameter(proprietorId);

temp.addParameter(date);

temp.addParameter(water\_fee);

temp.addParameter(watt\_fee);

temp.addParameter(gas\_fee);

temp.addParameter(tv\_fee);

temp.addParameter(property\_fee);

temp.addParameter(mortage\_fee);

temp.postRequest("insert into month\_bill values(? , ? , ? , ? , ? , ? , ? , ? , false)");

{

QList<QByteArray> info = temp.getInfo().split('\n');

res &= (QString(info[2]).compare("true") == 0);

}

if(!res)

{

temp.clear();

temp.setLength(0);

temp.setRollback(true);

temp.setCommit(false);

temp.setFlag(false);

temp.setTransaction(false);

temp.setSync(true);

temp.postRequest("");

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("操作失败"),QMessageBox::*Yes*);

return;

}

else

{

temp.clear();

temp.setLength(0);

temp.setRollback(false);

temp.setCommit(true);

temp.setFlag(false);

temp.setTransaction(false);

temp.setSync(true);

temp.postRequest("");

QMessageBox::warning(0,QMessageBox::trUtf8("提示"),QMessageBox::trUtf8("操作成功"),QMessageBox::*Yes*);

}

}

}

## 7.4 交互设计

本次数据库实验设计使用的架构模式是C/S模式,针对于现实情况与结合数据库课程设计的意义,最终我决定在服务器与客户端的交互以及服务器与数据库的交互上进行着重设计.

客户端与服务器进行交互的时候使用的传输协议是HTTP协议,在进行C/S交互的时候,为了既保证传输速度与执行语句的正确性,在客户端的代码中使用的是串行与并行相结合的技术;同时基于HTTP1.1协议,在客户端中使用HTTP使用长链接的方式,减少二次查询的网路传输开销.

客户端串行并行结合使用的HTTP 发送端底层主要实现:

void HTTP::startRequest(const QUrl &requestedUrl)

{

if(isPost())

{

url = requestedUrl;

httpRequestAborted = false;

QByteArray postData;

QNetworkRequest request;

postData.append("&sql=" + sql);

postData.append("&length=" + QString::number(getLength() , 10));

postData.append("&commit=" + QString(getCommit()?"true":"false"));

postData.append("&flag=" + QString(getFlag()?"true":"false"));

postData.append("&transaction=" + QString(getTransaction()?"true":"false"));

postData.append("&rollback=" + QString(getRollback()?"true":"false"));

postData.append("&parameter=" + QString(getParameter()));

request.setUrl(requestedUrl);

request.setHeader(QNetworkRequest::*ContentTypeHeader*,"application/x-www-form-urlencoded");

request.setRawHeader("Accept","text/html, application/xhtml+xml, \*/\*");

request.setRawHeader("Referer","http://localhost:8080");

request.setRawHeader("Accept-Language","en-US,\*");

request.setRawHeader("X-Requested-With","XMLHttpRequest");

request.setRawHeader("User-Agent","Mozilla/5.0");

request.setRawHeader("Accept-Encoding","gzip,deflate");

request.setRawHeader("Host","localhost:8080");

request.setRawHeader("Connection","close"); // "Connection" , "Keey-Alive"

request.setRawHeader("Cache-Control","no-cache");

QByteArray size;

size.clear();

size.append(postData.size());

reply = qnam.post(request , postData);

if(isSync())

{

while(!reply->isFinished())

{

QApplication::processEvents();

}

httpReadyRead();

httpFinished();

}

else

{

connect(reply, &QIODevice::readyRead, this, &HTTP::httpReadyRead);

connect(reply, &QNetworkReply::finished , this , &HTTP::httpFinished);

}

}

else

{

url = requestedUrl;

httpRequestAborted = false;

reply = qnam.get(QNetworkRequest(url));

if(isSync())

{

while(!reply->isFinished())

{

QApplication::processEvents();

}

httpReadyRead();

httpFinished();

}

else

{

connect(reply, &QIODevice::readyRead, this, &HTTP::httpReadyRead);

connect(reply, &QNetworkReply::finished , this , &HTTP::httpFinished);

}

}

}

客户端串行并行结合发送请求举例:

void DelChargerWidget::on\_pushButton\_clicked()

{

list[1].clear();

list[1].setFlag(true);

list[1].setCommit(false);

list[1].setRollback(false);

list[1].setTransaction(false);

list[1].setSync(true);

list[1].setLength(1);

list[1].addParameter(ui->comboBox->currentText());

list[1].postRequest("delete from user where login\_id = ?");

{

QList<QByteArray> info = list[1].getInfo().split('\n');

bool exec = (QString(info[2]).compare("true") == 0);

if(exec)

{

QMessageBox::information(0,"提示","删除成功",QMessageBox::*Ok*);

}

else

{

QMessageBox::information(0,"提示","删除失败",QMessageBox::*Ok*);

}

}

list[2].setFlag(false);

list[2].setCommit(false);

list[2].setLength(0);

list[2].postRequest("select login\_id,role\_type from user where role\_type='CHARGER'");

QObject::connect(list[2].getReply() , &QNetworkReply::finished , [&]()

{

int rowNum = 0;

ui->comboBox->clear();

model->clear();

QList<QByteArray> info = list[2].getInfo().split('\n');

int size = info[0].toInt();

int rows = info[1].toInt();

int sequence;

for(int i = 0 ; i < size ; i ++)

{

rowNum = model->*rowCount*();

for(int j = 0 ; j < rows ; j++)

{

sequence = i\*rows + j + 2;

model->setItem(rowNum ,j ,new QStandardItem(QString(info[sequence])));

if(j == 0)

{

ui->comboBox->addItem(QString(info[sequence]));

}

}

}

});

}

出于对服务器简洁与效率的考虑,我最后决定使用C++与QT基于QtNetwork类来实现HTTP协议与客户端进行交互,并且针对于服务器端,使用cookie技术与线程池调度,加速与客户端的交互,提高并发访问时候的速度,访问时的延迟.

服务器端cookie优化部分:

#include "httpcookie.h"

HttpCookie::HttpCookie()

{

version=1;

maxAge=0;

secure=false;

}

HttpCookie::HttpCookie(const QByteArray name, const QByteArray value, const int maxAge, const QByteArray path, const QByteArray comment, const QByteArray domain, const bool secure, const bool httpOnly)

{

this->name=name;

this->value=value;

this->maxAge=maxAge;

this->path=path;

this->comment=comment;

this->domain=domain;

this->secure=secure;

this->httpOnly=httpOnly;

this->version=1;

}

HttpCookie::HttpCookie(const QByteArray source)

{

version=1;

maxAge=0;

secure=false;

QList<QByteArray> list=splitCSV(source);

foreach(QByteArray part, list)

{

// Split the part into name and value

QByteArray name;

QByteArray value;

int posi=part.indexOf('=');

if (posi)

{

name=part.left(posi).trimmed();

value=part.mid(posi+1).trimmed();

}

else

{

name=part.trimmed();

value="";

}

// Set fields

if (name=="Comment")

{

comment=value;

}

else if (name=="Domain")

{

domain=value;

}

else if (name=="Max-Age")

{

maxAge=value.toInt();

}

else if (name=="Path")

{

path=value;

}

else if (name=="Secure")

{

secure=true;

}

else if (name=="HttpOnly")

{

httpOnly=true;

}

else if (name=="Version")

{

version=value.toInt();

}

else {

if (this->name.isEmpty())

{

this->name=name;

this->value=value;

}

else

{

qWarning("HttpCookie: Ignoring unknown %s=%s",name.data(),value.data());

}

}

}

}

QByteArray HttpCookie::toByteArray() const

{

QByteArray buffer(name);

buffer.append('=');

buffer.append(value);

if (!comment.isEmpty())

{

buffer.append("; Comment=");

buffer.append(comment);

}

if (!domain.isEmpty())

{

buffer.append("; Domain=");

buffer.append(domain);

}

if (maxAge!=0)

{

buffer.append("; Max-Age=");

buffer.append(QByteArray::number(maxAge));

}

if (!path.isEmpty())

{

buffer.append("; Path=");

buffer.append(path);

}

if (secure) {

buffer.append("; Secure");

}

if (httpOnly) {

buffer.append("; HttpOnly");

}

buffer.append("; Version=");

buffer.append(QByteArray::number(version));

return buffer;

}

void HttpCookie::setName(const QByteArray name)

{

this->name=name;

}

void HttpCookie::setValue(const QByteArray value)

{

this->value=value;

}

void HttpCookie::setComment(const QByteArray comment)

{

this->comment=comment;

}

void HttpCookie::setDomain(const QByteArray domain)

{

this->domain=domain;

}

void HttpCookie::setMaxAge(const int maxAge)

{

this->maxAge=maxAge;

}

void HttpCookie::setPath(const QByteArray path)

{

this->path=path;

}

void HttpCookie::setSecure(const bool secure)

{

this->secure=secure;

}

void HttpCookie::setHttpOnly(const bool httpOnly)

{

this->httpOnly=httpOnly;

}

QByteArray HttpCookie::getName() const

{

return name;

}

QByteArray HttpCookie::getValue() const

{

return value;

}

QByteArray HttpCookie::getComment() const

{

return comment;

}

QByteArray HttpCookie::getDomain() const

{

return domain;

}

int HttpCookie::getMaxAge() const

{

return maxAge;

}

QByteArray HttpCookie::getPath() const

{

return path;

}

bool HttpCookie::getSecure() const

{

return secure;

}

bool HttpCookie::getHttpOnly() const

{

return httpOnly;

}

int HttpCookie::getVersion() const

{

return version;

}

QList<QByteArray> HttpCookie::splitCSV(const QByteArray source)

{

bool inString=false;

QList<QByteArray> list;

QByteArray buffer;

for (int i=0; i<source.size(); ++i)

{

char c=source.at(i);

if (inString==false)

{

if (c=='\"')

{

inString=true;

}

else if (c==';')

{

QByteArray trimmed=buffer.trimmed();

if (!trimmed.isEmpty())

{

list.append(trimmed);

}

buffer.clear();

}

else

{

buffer.append(c);

}

}

else

{

if (c=='\"')

{

inString=false;

}

else {

buffer.append(c);

}

}

}

QByteArray trimmed=buffer.trimmed();

if (!trimmed.isEmpty())

{

list.append(trimmed);

}

return list;

}

服务器线程池调度:

main.cpp

#include <QCoreApplication>

#include <QDir>

#include <iostream>

#include <string>

#include "httplistener.h"

#include "templatecache.h"

#include "httpsessionstore.h"

#include "staticfilecontroller.h"

#include "filelogger.h"

#include "requestmapper.h"

#include "httpserver.h"

#include "src/database/sql.h"

#include "src/database/serialquery.h"

/\*\* Cache for template files \*/

TemplateCache\* templateCache;

/\*\* Storage for session cookies \*/

HttpSessionStore\* sessionStore;

/\*\* Controller for static files \*/

StaticFileController\* staticFileController;

/\*\* Redirects log messages to a file \*/

FileLogger\* logger;

/\*\* Search the configuration file \*/

QString searchConfigFile()

{

QString binDir=QCoreApplication::applicationDirPath();

QString appName=QCoreApplication::applicationName();

QString fileName(appName+".ini");

QStringList searchList;

searchList.append(binDir);

searchList.append(binDir+"/etc");

searchList.append(binDir+"/../etc");

searchList.append(binDir+"/../../etc"); // for development without shadow build

searchList.append(binDir+"/../"+appName+"/etc"); // for development with shadow build

searchList.append(binDir+"/../../"+appName+"/etc"); // for development with shadow build

searchList.append(binDir+"/../../../"+appName+"/etc"); // for development with shadow build

searchList.append(binDir+"/../../../../"+appName+"/etc"); // for development with shadow build

searchList.append(binDir+"/../../../../../"+appName+"/etc"); // for development with shadow build

searchList.append(QDir::rootPath()+"etc/opt");

searchList.append(QDir::rootPath()+"etc");

foreach (QString dir, searchList)

{

QFile file(dir+"/"+fileName);

if (file.exists())

{

// found

fileName=QDir(file.*fileName*()).canonicalPath();

qDebug("Using config file %s",qPrintable(fileName));

return fileName;

}

}

// not found

foreach (QString dir, searchList)

{

qWarning("%s/%s not found",qPrintable(dir),qPrintable(fileName));

}

qFatal("Cannot find config file %s",qPrintable(fileName));

return 0;

}

/\*

Entry point of the program.

\*/

// The main program starts the HTTP server

int main(int argc, char \*argv[])

{

QCoreApplication\* app = new QCoreApplication(argc,argv);

app->setApplicationName("DBServer");

// app->setOrganizationName("Butterfly");

// Find the configuration file

QString configFileName= "/etc/qt/server.ini";

// Configure logging into a file

QSettings\* settings = new QSettings(configFileName,QSettings::*IniFormat*,app);

// Init the sql databases

SQL\* sql = new SQL(Q\_NULLPTR);

if(sql->openDatabase("propertydata"))

{

if(sql->logIn("elvis" , "Idon'tknow"))

{

qDebug() << "MAIN : LOG IN SUCCESSFUL .";

}

else

{

qDebug() << "MAIN : LOG IN FAILED .";

return EXIT\_FAILURE;

}

}

else

{

app->destroyed();

qDebug() << "MAIN : OPEN DATABASE FAILED .";

return EXIT\_FAILURE;

}

settings->beginGroup("listener");

// Init the server

HttpServer \* handler = new HttpServer(app);

handler->setSQL(sql);

handler->setDatabase(sql->getDatabase());

HttpListener\* listener=new HttpListener(settings,handler,app);

return app->exec();

}

httplistener.cpp

#include "httplistener.h"

#include "httpconnectionhandler.h"

#include "httpconnectionhandlerpool.h"

#include <QCoreApplication>

HttpListener::HttpListener(QSettings\* settings, HttpRequestHandler\* requestHandler, QObject \*parent)

: QTcpServer(parent)

{

Q\_ASSERT(settings!=0);

Q\_ASSERT(requestHandler!=0);

pool=NULL;

this->settings=settings;

this->requestHandler=requestHandler;

// Reqister type of socketDescriptor for signal/slot handling

qRegisterMetaType<tSocketDescriptor>("tSocketDescriptor");

// Start listening

listen();

}

HttpListener::~*HttpListener*()

{

close();

qDebug("HttpListener: destroyed");

}

void HttpListener::listen()

{

if (!pool)

{

pool=new HttpConnectionHandlerPool(settings,requestHandler);

}

QString host = settings->value("host").toString();

int port=settings->value("port").toInt();

QTcpServer::listen(host.isEmpty() ? QHostAddress::*Any* : QHostAddress(host), port);

if (!isListening())

{

qCritical("HttpListener: Cannot bind on port %i: %s",port,qPrintable(errorString()));

}

else {

qDebug("HttpListener: Listening on port %i",port);

}

}

void HttpListener::close() {

QTcpServer::close();

qDebug("HttpListener: closed");

if (pool) {

delete pool;

pool=NULL;

}

}

void HttpListener::*incomingConnection*(tSocketDescriptor socketDescriptor) {

#ifdef SUPERVERBOSE

qDebug("HttpListener: New connection");

#endif

HttpConnectionHandler\* freeHandler=NULL;

if (pool)

{

freeHandler=pool->getConnectionHandler();

}

// Let the handler process the new connection.

if (freeHandler)

{

// The descriptor is passed via signal/slot because the handler lives in another

// thread and cannot open the socket when directly called by another thread.

connect(this,SIGNAL(handleConnection(tSocketDescriptor)),freeHandler,SLOT(handleConnection(tSocketDescriptor)));

emit handleConnection(socketDescriptor);

disconnect(this,SIGNAL(handleConnection(tSocketDescriptor)),freeHandler,SLOT(handleConnection(tSocketDescriptor)));

}

else

{

// Reject the connection

qDebug("HttpListener: Too many incoming connections");

QTcpSocket\* socket=new QTcpSocket(this);

socket->*setSocketDescriptor*(socketDescriptor);

connect(socket, SIGNAL(disconnected()), socket, SLOT(deleteLater()));

socket->write("HTTP/1.1 503 too many connections\r\nConnection: close\r\n\r\nToo many connections\r\n");

socket->*disconnectFromHost*();

}

}

httpserver.cpp

#include "httpserver.h"

#include <QDebug>

#include <QSqlQuery>

#include <QSqlRecord>

#include <QSqlError>

#include "../src/database/sql.h"

HttpServer::HttpServer()

{

qDebug() << "HTTP SERVER INIT .";

}

HttpServer::HttpServer(QCoreApplication \*app)

{

qDebug() << "HTTP SERVER WITH PARAMETER INIT .";

sem.release();

}

// The request handler receives and responds HTTP requests

void HttpServer::*service*(HttpRequest& request , HttpResponse& response)

{

// Get a request parameters

// QByteArray username = request.getParameter("username");

sem.acquire(1);

QByteArray sql = request.getParameter("sql");

QByteArray length = request.getParameter("length");

QByteArray commit = request.getParameter("commit");

QByteArray flag = request.getParameter("flag");

QByteArray transaction = request.getParameter("transaction");

QByteArray rollback = request.getParameter("roolback");

QByteArray parameter = request.getParameter("parameter");

qDebug() << "HTTPSERVER : ";

qDebug() << request.getParameterMap();

bool prepare = getSQL()->getQuery()->prepare(sql);

QString temp = parameter;

QStringList list = temp.split('\t');

if(length.toInt() != 0 && temp.length() == 0)

{

qDebug() << "HTTPSERVER : ERROR FOR LENGTH ! ";

qDebug() << "HTTPSERVER : LENGTH IN MAP " << length.toInt();

qDebug() << "HTTPSERVER : LENGTH IN LIST " << list.length();

response.write(QByteArray("-1\n"));

response.write(QByteArray("-1\n"));

}

else if((temp.compare("") != 0) && length.toInt() != list.length())

{

qDebug() << "HTTPSERVER : ERROR FOR LENGTH ! ";

qDebug() << "HTTPSERVER : LENGTH IN MAP " << length.toInt();

qDebug() << "HTTPSERVER : LENGTH IN LIST " << list.length();

response.write(QByteArray("-1\n"));

response.write(QByteArray("-1\n"));

}

else if(!prepare)

{

if(sql.length() == 0)

{

bool result = true;

if(!QString(commit).compare("true"))

{

result &= getSQL()->getDatabase()->commit();

}

if(!QString(transaction).compare("true"))

{

result &= getSQL()->getDatabase()->transaction();

}

if(!QString(rollback).compare("true"))

{

result &= getSQL()->getDatabase()->rollback();

}

response.write(QByteArray("-1\n"));

response.write(QByteArray("-1\n"));

QByteArray sender = QByteArray(result ?"true\n":"false\n");

response.write(sender);

qDebug() << "HTTPSERVER : SEND MESSAGE FOR : " ;

qDebug() << sender;

}

else

{

qDebug() << "HTTPSERVER : ERROR FOR PREPARE ! ";

response.write(QByteArray("-1\n"));

response.write(QByteArray("-1\n"));

}

}

else

{

for(int i = 0 ; i < length.toInt(); i++)

{

qDebug() << "BINDING FOR : " << list[i];

getSQL()->getQuery()->addBindValue(list[i]);

}

bool result = getSQL()->getQuery()->exec();

QByteArray SIZE;

SIZE.append(QString::number(getSQL()->getQuery()->size()));

QByteArray ROWS;

ROWS.append(QString::number(getSQL()->getQuery()->record().count()));

response.write(SIZE + "\n");

response.write(ROWS + "\n");

qDebug() << "GET THE SIZE FOR : " << SIZE;

qDebug() << "GET THE ROWS FOR : " << ROWS;

qDebug() << "GET THE RESULT FOR : " << result;

if(!QString(commit).compare("true"))

{

getSQL()->getDatabase()->commit();

}

if(!QString(transaction).compare("true"))

{

getSQL()->getDatabase()->transaction();

}

if(!QString(rollback).compare("true"))

{

getSQL()->getDatabase()->rollback();

}

QByteArray sender;

int size = getSQL()->getQuery()->record().count();

if(!QString(flag).compare("true"))

{

sender.append(QString(result?"true":"false") + "\n");

}

while(getSQL()->getQuery()->next())

{

for(int i = 0 ; i < size ; i++)

{

sender.append(getSQL()->getQuery()->value(i).toString() + "\n");

qDebug() << getSQL()->getQuery()->value(i);

}

}

qDebug() << "HTTPSERVER : SEND MESSAGE FOR : " ;

qDebug() << sender;

response.write(sender);

response.flush();

}

sem.release(1);

}

HttpServer::~*HttpServer*()

{

qDebug() << "HTTP SERVER OVER .";

}

QSqlDatabase\* HttpServer::getDatabase()

{

return this->db;

}

void HttpServer::setDatabase(QSqlDatabase\* db)

{

this->db = db;

}

void HttpServer::setSQL(SQL \*point)

{

this->sql = point;

}

SQL\* HttpServer::getSQL()

{

return this->sql;

}

## 7.5 Linux内核设计

针对于服务器而言,由于服务器运行于Gentoo Linux上 , 此时可以很方便的修改,调优内核,从而将内核的调度请求等调整到最适合数据库服务器,从而再一次深度优化服务器端,加速查询效率.

服务器所属的机器的使用的内核版本是4.7.4 , 针对于该内核,主要进行一下方面的调优:

RCU Subsystem

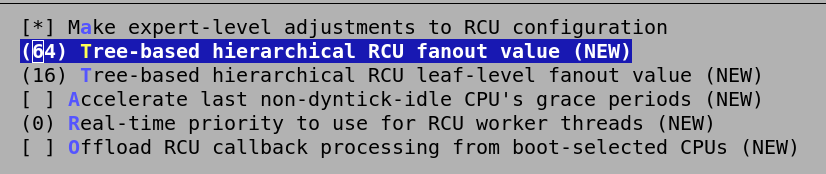
[RCU(Read-Copy Update)子系统](http://www.ibm.com/developerworks/cn/linux/l-rcu/).它允许程序查看到正在被修改/更新的文件.在读多写少的情况下,这是一个高性能的锁机制,对于被RCU保护的共享数据结构,读者不需要获得任何锁就可以访问它(速度非常快),但写者在访问它时首先拷贝一个副本,然后对副本进行修改,最后使用一个回调机制在适当的时机把指向原来数据的指针重新指向新的被修改的数据,速度非常慢.RCU只适用于读多写少的情况:如网络路由表的查询更新,设备状态表的维护,数据结构的延迟释放以及多径I/O设备的维护等.

Preemptible tree-based hierarchical RCU  
 CONFIG\_TREE\_PREEMPT\_RCU

抢占式基于树型分层结构的实现.最适用于那些要求快速响应的多CPU实时系统.

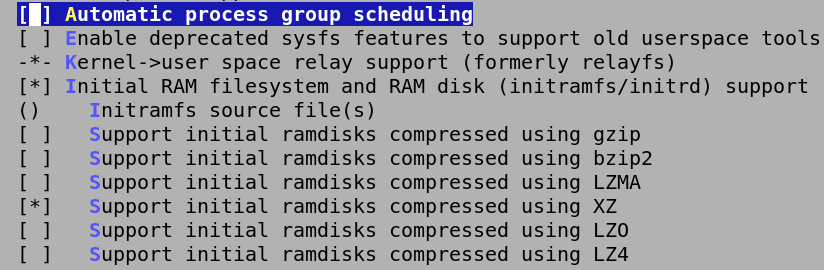
Accelerate last non-dyntick-idle CPU's grace periods  
 CONFIG\_RCU\_FAST\_NO\_HZ

即使CPU还在忙碌,也允许进入dynticks-idle状态,并且阻止RCU每4个滴答就唤醒一次该CPU,这样能够更有效的使用电力,同时也拉长了RCU grace period的时间,造成性能降低.服务器上关闭此选项.



Automatic process group scheduling  
 CONFIG\_SCHED\_AUTOGROUP

每个TTY动态地创建任务分组(cgroup),这样就可以降低高负载情况下的桌面延迟.也就是传说中的桌面"[鸡血补丁](http://wowubuntu.com/kernel-patch.html)",在服务器上关闭该项.



SLUB per cpu partial cache  
 CONFIG\_SLUB\_CPU\_PARTIAL

让SLUB内存分配器使用基于每个CPU的局部缓存,这样可以加速分配和释放属于此CPU范围内的对象,服务器则选"Y".

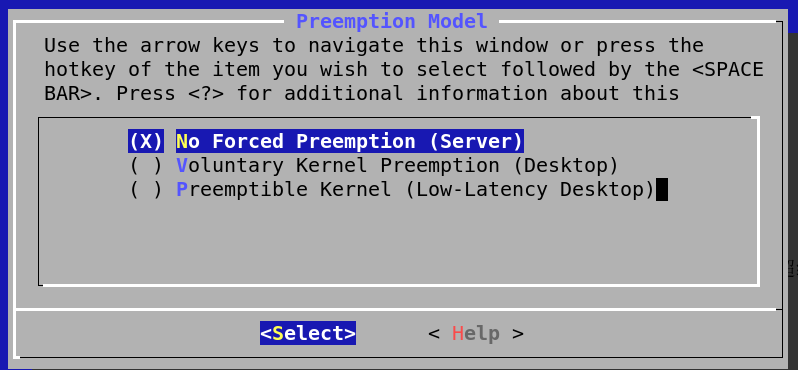
QQ图片20161010170804

Preemption Model

内核抢占模式

No Forced Preemption (Server)  
 CONFIG\_PREEMPT\_NONE

禁止内核抢占,可以得到最大的吞吐量,服务器和科学计算环境选中.



Opportunistic sleep  
 CONFIG\_PM\_AUTOSLEEP

这是一种从[安卓借鉴过来的休眠方式](https://lwn.net/Articles/479841/).这个特性在安卓系统上被称为"suspend blockers"或"wakelocks".这是一种更激进的电源管理模式,以尽可能节约电力为目的.系统默认就处于休眠状态,仅为内存和少数唤醒系统所必须的设备供电,当有任务(唤醒源)需要运行的时候才唤醒相关组件工作,工作完成后又立即进入休眠状态.不过这些特性需要相应的设备驱动程序的支持.目前除了安卓设备,在PC和服务器领域,能够利用此特性的驱动还比较少,不过这是一项非常有前途的电源技术,在这里打开这个选项.

QQ图片20161010171335

Watchdog Timer Support  
 CONFIG\_WATCHDOG

选"Y"并选中下面相应的驱动之后,再创建一个主/次设备号为10/130的字符设备"/dev/watchdog",即可拥有一只[看门狗](https://www.ibm.com/developerworks/cn/linux/l-cn-watchdog/).其工作原理是:当/dev/watchdog设备被打开后,如果[喂狗守护进程](http://ibiblio.org/pub/Linux/system/daemons/watchdog/)超过60秒没有喂狗(写入"/dev/watchdog"),那么底层的看门狗硬件将会触发整个机器硬重启(相当于按下面板上的"RESET"按钮).这对于提高服务器的在线率来说意义重大.

WatchDog Timer Driver Core  
CONFIG\_WATCHDOG\_CORE

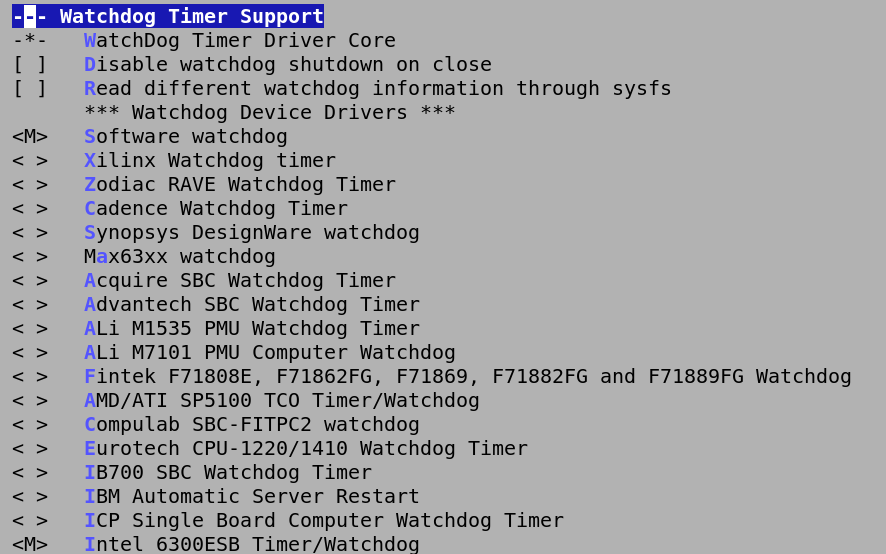
看门狗核心驱动,它为所有特定于具体硬件的看门狗驱动提供了统一的框架和"/dev/watchdog"接口(未来还会包括sysfs接口).使用看门狗的必选.

Disable watchdog shutdown on close  
CONFIG\_WATCHDOG\_NOWAYOUT

默认情况下(此项="N")如果喂狗进程关闭"/dev/watchdog"文件,那么表示停止看门狗功能.开启此项后,看门狗一旦启用就不能被停止(即使关闭"/dev/watchdog"文件也不会停止).

Software watchdog  
CONFIG\_SOFT\_WATCHDOG

内核提供的"软看门狗".使用它不需要有任何硬件的支持,但可靠性不如硬件看门狗,仅能应对喂狗进程的崩溃,不能应对内核本身的崩溃.在某些情况下(例如Oracle数据库),CONFIG\_HANGCHECK\_TIMER是比"软看门狗"更好的选择.



InfiniBand support  
 CONFIG\_INFINIBAND

[InfiniBand](http://server.it168.com/a2012/0924/1401/000001401869_all.shtml)是一种低延迟/高带宽数据中心互联架构,对虚拟化技术也提供了良好的支持.主要用于服务器领域.

CPU/Task time and stats accounting

CPU/进程的时间及状态统计

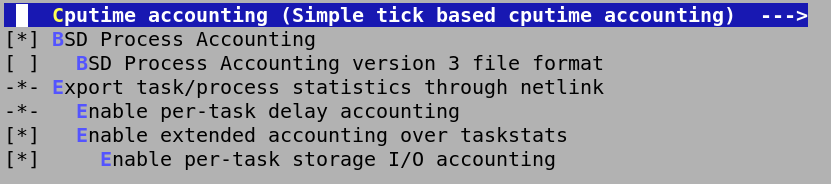
Cputime accounting

CPU时间统计方式

Simple tick based cputime accounting

CONFIG\_TICK\_CPU\_ACCOUNTING

简单的基于滴答的统计,适用于大多数场合



Enable AIO support  
 CONFIG\_AIO

开启POSIX异步IO支持.它常常被高性能的多线程程序使用,开启

Enable madvise/fadvise syscalls  
 CONFIG\_ADVISE\_SYSCALLS

开启内核的[madvise()](http://docs.oracle.com/cd/E19253-01/819-7052/lgroups-23/)/[fadvise()](http://blog.csdn.net/vah101/article/details/7317557)系统调用支持(2.6.16版本开始引入).以允许应用程序预先提示内核,它将如何使用特定的内存与文件.这种措施有助于提升应用程序的性能.

Enable 1GB pages for kernel pagetables  
 CONFIG\_DIRECT\_GBPAGES

允许[内核页表](http://www.sigma.me/2011/01/22/mem-page-tlb.html)使用大小为1GB的[Hugepages](http://wiki.debian.org/Hugepages)并进行直线映射(linear mapping),需要高端CPU的支持(可以用"grep pdpe1gb /proc/cpuinfo"命令检查).这可以减小[页表缓存](http://zh.wikipedia.org/wiki/%E8%BD%89%E8%AD%AF%E5%BE%8C%E5%82%99%E7%B7%A9%E8%A1%9D%E5%8D%80)([Translation Lookaside Buffer](http://en.wikipedia.org/wiki/Translation_lookaside_buffer))的压力,从而提升系统的性能,这对于拥有海量内存并且运行某些特定应用(PosgreSQL,MySQL,Java,Memcached,KVM,Xen...)的系统来说比较有意义.此时打开该选项.

CPU Frequency scaling  
 CONFIG\_CPU\_FREQ

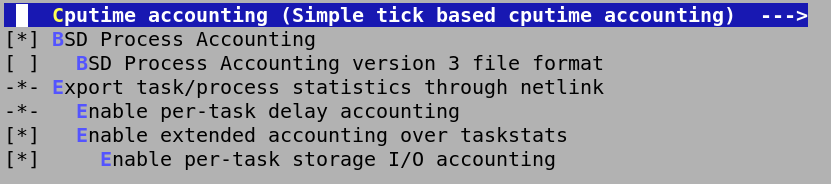
[CPUfreq](http://www.ibm.com/developerworks/cn/linux/l-cpufreq-1/)子系统允许动态改变CPU主频,达到省电和降温的目的.现如今的CPU都已经支持动态频率调整,开启支持.

Default CPUFreq governor

默认的CPU频率[调节策略](http://www.ibm.com/developerworks/cn/linux/l-cpufreq-2/).不同策略拥有不同的[调节效果](http://www.ibm.com/developerworks/cn/linux/l-cpufreq-3/).

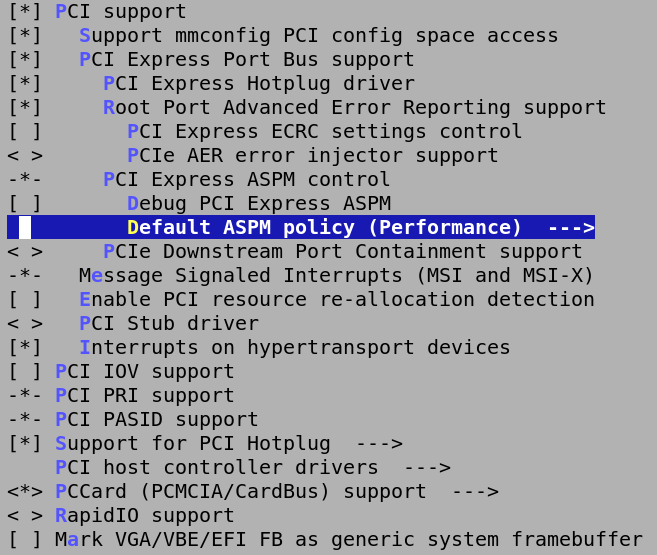
'performance' governor  
CONFIG\_CPU\_FREQ\_GOV\_PERFORMANCE

'性能'优先,静态的将频率设置为cpu支持的最高频率.最耗电,发热量最大,在这里为了性能调优,使用这个选项.



Default ASPM policy

默认的ASPM电源管理策略.下面的三个选项:"BIOS default"表示使用BIOS中的设置作为默认."Powersave"表示在可能的情况下,默认使用"L0s"和"L1",以尽可能节约电力."Performance"表示禁止使用"L0s"和"L1"(即使BIOS开启也同样禁止),以保证最高性能.



Cache target

CONFIG\_DM\_CACHE

[dm-cache](http://www.oschina.net/p/dm-cache)通过将频繁使用的热点数据缓存到一个容量较小但性能很高的存储设备上,从而提升块设备的性能.它支持[writeback和writethrough](http://blog.csdn.net/string19820108/article/details/7327552)两种模式,并可以使用多种[缓存策略(policy)](http://wangxu.me/blog/p/787)以判断哪些是热点数据以及哪些数据需要从缓存中移除.

MQ Cache Policy  
 CONFIG\_DM\_CACHE\_MQ

MQ缓存策略.这是目前唯一真正可用的缓存策略.

KCopy

内核Copy

Memory-to-memory copies using kernel assist  
CONFIG\_KCOPY

高性能的进程间内存复制(可以减少一次向共享内存的复制动作).主要用于高性能并行计算领域,比如基于[消息传递接口](http://zh.wikipedia.org/wiki/%E8%A8%8A%E6%81%AF%E5%82%B3%E9%81%9E%E4%BB%8B%E9%9D%A2)([Message Passing Interface](http://wenku.baidu.com/view/ee8bf3390912a216147929f3.html))协议的开发的并行程序.

除此之外,在内核调优的过程中还减去了许多影响服务器性能的内核选项,通过深度裁剪Linux内核,充分发挥了服务器端并行查询的能力,在这里由于时间关系等原因,无法一一截图列举,所以只能在这里下附上/usr/src/linux/.config文件供做服务器调优配置的参考.

# 八、测试和运行

测试用例.

**-- MySQL dump 10.13 Distrib 5.6.33, for Linux (x86\_64)**

**--**

**-- Host: localhost Database: propertydata**

**-- ------------------------------------------------------**

**-- Server version 5.6.33-log**

**/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;**

**/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;**

**/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;**

**/\*!40101 SET NAMES utf8 \*/;**

**/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;**

**/\*!40103 SET TIME\_ZONE='+00:00' \*/;**

**/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;**

**/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;**

**/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;**

**/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;**

**--**

**-- Table structure for table `building`**

**--**

**DROP TABLE IF EXISTS `building`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `building` (**

**`building\_id` varchar(10) NOT NULL COMMENT '楼栋id',**

**`floor\_num` smallint(3) NOT NULL COMMENT '楼层数目',**

**`built\_date` date NOT NULL COMMENT '建成时间',**

**PRIMARY KEY (`building\_id`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `building`**

**--**

**LOCK TABLES `building` WRITE;**

**/\*!40000 ALTER TABLE `building` DISABLE KEYS \*/;**

**INSERT INTO `building` VALUES ('1#',5,'2016-01-01'),('2#',5,'2016-02-01'),('3#',6,'2016-03-01'),('4#',6,'2016-04-01');**

**/\*!40000 ALTER TABLE `building` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**--**

**-- Table structure for table `func`**

**--**

**DROP TABLE IF EXISTS `func`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `func` (**

**`function` varchar(40) NOT NULL COMMENT '功能',**

**PRIMARY KEY (`function`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `func`**

**--**

**LOCK TABLES `func` WRITE;**

**/\*!40000 ALTER TABLE `func` DISABLE KEYS \*/;**

**INSERT INTO `func` VALUES ('ADD\_DEL\_BUILDING'),('ADD\_DEL\_CHARGER'),('ADD\_DEL\_PROPRIETOR'),('ADD\_DEL\_ROOM'),('CHARGE\_FEE'),('RECORD\_METER'),('REPORT\_FORMS'),('SET\_STANDARD');**

**/\*!40000 ALTER TABLE `func` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**--**

**-- Table structure for table `meter`**

**--**

**DROP TABLE IF EXISTS `meter`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `meter` (**

**`building\_id` varchar(10) NOT NULL COMMENT '楼栋id',**

**`room\_id` varchar(10) NOT NULL COMMENT '房屋id',**

**`date` date NOT NULL COMMENT '抄表时间',**

**`water` decimal(6,1) NOT NULL COMMENT '水表读数',**

**`watt` decimal(6,1) NOT NULL COMMENT '电表读数',**

**`gas` decimal(6,1) NOT NULL COMMENT '煤气表读数',**

**PRIMARY KEY (`room\_id`,`building\_id`,`date`),**

**KEY `building\_id` (`building\_id`,`room\_id`),**

**CONSTRAINT `meter\_ibfk\_1` FOREIGN KEY (`building\_id`, `room\_id`) REFERENCES `room` (`building\_id`, `room\_id`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `meter`**

**--**

**LOCK TABLES `meter` WRITE;**

**/\*!40000 ALTER TABLE `meter` DISABLE KEYS \*/;**

**/\*!40000 ALTER TABLE `meter` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**--**

**-- Table structure for table `month\_bill`**

**--**

**DROP TABLE IF EXISTS `month\_bill`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `month\_bill` (**

**`ID` varchar(18) NOT NULL COMMENT '业主身份证号',**

**`date` date NOT NULL COMMENT '月账单产生日期',**

**`water` decimal(5,2) NOT NULL COMMENT '月水费数额',**

**`watt` decimal(5,2) NOT NULL COMMENT '月电费数额',**

**`gas` decimal(5,2) NOT NULL COMMENT '月煤气费数额',**

**`tv` decimal(5,2) NOT NULL,**

**`property` decimal(5,2) NOT NULL COMMENT '月物业费数额',**

**`mortage` decimal(8,2) NOT NULL COMMENT '月房贷数额',**

**`payed` varchar(1) DEFAULT NULL,**

**PRIMARY KEY (`ID`,`date`),**

**CONSTRAINT `month\_bill\_ibfk\_1` FOREIGN KEY (`ID`) REFERENCES `proprietor` (`ID`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `month\_bill`**

**--**

**LOCK TABLES `month\_bill` WRITE;**

**/\*!40000 ALTER TABLE `month\_bill` DISABLE KEYS \*/;**

**/\*!40000 ALTER TABLE `month\_bill` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**--**

**-- Table structure for table `proprietor`**

**--**

**DROP TABLE IF EXISTS `proprietor`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `proprietor` (**

**`ID` varchar(18) NOT NULL COMMENT '业主身份证号',**

**`name` varchar(30) NOT NULL COMMENT '业主姓名',**

**`organization` varchar(50) NOT NULL DEFAULT 'none' COMMENT '业主工作单位',**

**`building\_id` varchar(10) NOT NULL COMMENT '楼栋id',**

**`room\_id` varchar(10) NOT NULL COMMENT '房屋id',**

**`own\_date` date NOT NULL COMMENT '入住时间',**

**PRIMARY KEY (`ID`),**

**KEY `building\_id` (`building\_id`,`room\_id`),**

**CONSTRAINT `proprietor\_ibfk\_1` FOREIGN KEY (`building\_id`, `room\_id`) REFERENCES `room` (`building\_id`, `room\_id`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `proprietor`**

**--**

**LOCK TABLES `proprietor` WRITE;**

**/\*!40000 ALTER TABLE `proprietor` DISABLE KEYS \*/;**

**/\*!40000 ALTER TABLE `proprietor` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**--**

**-- Table structure for table `proprietor\_phone\_num`**

**--**

**DROP TABLE IF EXISTS `proprietor\_phone\_num`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `proprietor\_phone\_num` (**

**`phone\_num` varchar(20) NOT NULL COMMENT '业主电话号码',**

**`ID` varchar(18) NOT NULL COMMENT '业主身份证号',**

**PRIMARY KEY (`phone\_num`),**

**KEY `ID` (`ID`),**

**CONSTRAINT `proprietor\_phone\_num\_ibfk\_1` FOREIGN KEY (`ID`) REFERENCES `proprietor` (`ID`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `proprietor\_phone\_num`**

**--**

**LOCK TABLES `proprietor\_phone\_num` WRITE;**

**/\*!40000 ALTER TABLE `proprietor\_phone\_num` DISABLE KEYS \*/;**

**/\*!40000 ALTER TABLE `proprietor\_phone\_num` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**--**

**-- Table structure for table `role`**

**--**

**DROP TABLE IF EXISTS `role`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `role` (**

**`role\_type` varchar(20) NOT NULL COMMENT '角色类型',**

**PRIMARY KEY (`role\_type`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `role`**

**--**

**LOCK TABLES `role` WRITE;**

**/\*!40000 ALTER TABLE `role` DISABLE KEYS \*/;**

**INSERT INTO `role` VALUES ('CHARGER'),('MANAGER');**

**/\*!40000 ALTER TABLE `role` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**--**

**-- Table structure for table `role\_func`**

**--**

**DROP TABLE IF EXISTS `role\_func`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `role\_func` (**

**`role\_type` varchar(20) NOT NULL COMMENT '角色类型',**

**`function` varchar(20) NOT NULL COMMENT '功能',**

**PRIMARY KEY (`role\_type`,`function`),**

**KEY `function` (`function`),**

**CONSTRAINT `role\_func\_ibfk\_1` FOREIGN KEY (`role\_type`) REFERENCES `role` (`role\_type`),**

**CONSTRAINT `role\_func\_ibfk\_2` FOREIGN KEY (`function`) REFERENCES `func` (`function`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `role\_func`**

**--**

**LOCK TABLES `role\_func` WRITE;**

**/\*!40000 ALTER TABLE `role\_func` DISABLE KEYS \*/;**

**INSERT INTO `role\_func` VALUES ('MANAGER','ADD\_DEL\_BUILDING'),('MANAGER','ADD\_DEL\_CHARGER'),('MANAGER','ADD\_DEL\_PROPRIETOR'),('MANAGER','ADD\_DEL\_ROOM'),('CHARGER','CHARGE\_FEE'),('MANAGER','CHARGE\_FEE'),('CHARGER','RECORD\_METER'),('MANAGER','RECORD\_METER'),('MANAGER','REPORT\_FORMS'),('MANAGER','SET\_STANDARD');**

**/\*!40000 ALTER TABLE `role\_func` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**--**

**-- Table structure for table `room`**

**--**

**DROP TABLE IF EXISTS `room`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `room` (**

**`building\_id` varchar(10) NOT NULL COMMENT '楼栋id',**

**`room\_id` varchar(10) NOT NULL COMMENT '每栋楼上房屋id',**

**`area` decimal(5,2) NOT NULL COMMENT '房屋面积',**

**`stayed` varchar(1) NOT NULL DEFAULT '0',**

**PRIMARY KEY (`building\_id`,`room\_id`),**

**CONSTRAINT `room\_ibfk\_1` FOREIGN KEY (`building\_id`) REFERENCES `building` (`building\_id`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `room`**

**--**

**LOCK TABLES `room` WRITE;**

**/\*!40000 ALTER TABLE `room` DISABLE KEYS \*/;**

**INSERT INTO `room` VALUES ('1#','101',100.00,'1'),('1#','102',100.00,'0'),('1#','201',100.00,'0'),('1#','202',100.00,'0'),('1#','301',100.00,'0'),('1#','302',100.00,'0'),('1#','401',100.00,'0'),('1#','402',100.00,'0'),('1#','501',100.00,'0'),('1#','502',100.00,'0'),('2#','101',100.00,'0'),('2#','102',100.00,'0'),('2#','201',100.00,'0'),('2#','202',100.00,'0'),('2#','301',100.00,'0'),('2#','302',100.00,'0'),('2#','401',100.00,'0'),('2#','402',100.00,'0'),('2#','501',100.00,'0'),('2#','502',100.00,'0'),('3#','101',120.00,'0'),('3#','102',120.00,'0'),('3#','201',120.00,'0'),('3#','202',120.00,'0'),('3#','301',120.00,'0'),('3#','302',120.00,'0'),('3#','401',120.00,'0'),('3#','402',120.00,'0'),('3#','501',120.00,'0'),('3#','502',120.00,'0'),('3#','601',120.00,'0'),('3#','602',120.00,'0'),('4#','101',120.00,'0'),('4#','102',120.00,'0'),('4#','201',120.00,'0'),('4#','202',120.00,'0'),('4#','301',120.00,'0'),('4#','302',120.00,'0'),('4#','401',120.00,'0'),('4#','402',120.00,'0'),('4#','501',120.00,'0'),('4#','502',120.00,'0'),('4#','601',120.00,'0'),('4#','602',120.00,'0');**

**/\*!40000 ALTER TABLE `room` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**--**

**-- Table structure for table `standard`**

**--**

**DROP TABLE IF EXISTS `standard`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `standard` (**

**`start\_date` date NOT NULL COMMENT '标准启动时间',**

**`water\_per\_stere` decimal(5,2) NOT NULL DEFAULT '0.00' COMMENT '每立方米水价(元)',**

**`watt\_per\_degree` decimal(5,2) NOT NULL DEFAULT '0.00' COMMENT '每度电价格(元)',**

**`gas\_per\_stere` decimal(5,2) NOT NULL DEFAULT '0.00' COMMENT '每立方米煤气价(元)',**

**`tv\_per\_month` decimal(5,2) NOT NULL DEFAULT '0.00' COMMENT '每月有线电视价格',**

**`prop\_per\_sqare\_month` decimal(5,2) NOT NULL DEFAULT '0.00' COMMENT '每平米每月物业费',**

**`mortage` decimal(8,2) NOT NULL DEFAULT '0.00',**

**PRIMARY KEY (`start\_date`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `standard`**

**--**

**LOCK TABLES `standard` WRITE;**

**/\*!40000 ALTER TABLE `standard` DISABLE KEYS \*/;**

**INSERT INTO `standard` VALUES ('2016-01-20',2.00,2.00,2.50,18.00,1.00,6000.00),('2016-10-01',2.00,2.50,3.00,18.00,0.50,6000.00),('2017-01-01',4.00,4.00,4.00,4.00,4.00,8000.00);**

**/\*!40000 ALTER TABLE `standard` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**--**

**-- Table structure for table `user`**

**--**

**DROP TABLE IF EXISTS `user`;**

**/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;**

**/\*!40101 SET character\_set\_client = utf8 \*/;**

**CREATE TABLE `user` (**

**`login\_id` varchar(20) NOT NULL COMMENT '系统用户登录名',**

**`password` varchar(100) NOT NULL COMMENT '登录密码哈希密文',**

**`role\_type` varchar(20) NOT NULL,**

**PRIMARY KEY (`login\_id`),**

**KEY `role\_type` (`role\_type`),**

**CONSTRAINT `user\_ibfk\_1` FOREIGN KEY (`role\_type`) REFERENCES `role` (`role\_type`)**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;**

**--**

**-- Dumping data for table `user`**

**--**

**LOCK TABLES `user` WRITE;**

**/\*!40000 ALTER TABLE `user` DISABLE KEYS \*/;**

**INSERT INTO `user` VALUES ('elvis','37ed2c07cda30a82f8442c52944475ea','MANAGER'),('operator','96e79218965eb72c92a549dd5a330112','CHARGER');**

**/\*!40000 ALTER TABLE `user` ENABLE KEYS \*/;**

**UNLOCK TABLES;**

**/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;**

**/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;**

**/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;**

**/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;**

**/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;**

**/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;**

**/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;**

**/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;**

**-- Dump completed on 2016-10-09 21:44:00**

# 九、总结

**总结：**

**特色之处**

* 使用C++开发,C/S架构,性能高于B/S架构,程序简洁轻快.
* 使用C++基于QT5.6重新封装HTTP,轻小快.
* 基于Gentoo Linux 进行内核的优化,充分提升数据库查询时候的速度.
* 使用Qt库开发,可以轻易的跨平台,具有实际意义.
* 将UI组件进行分离解耦,降低各个模块间的耦合度,方便后期的维护
* 符合最小授权原理,在安全上作出了保证.
* 动态部署用户的界面,依据用户权限而定,设计合理,用户体验较好.
* 能够进行事务处理,并且可以出错后回滚,保持数据的一致性,并且支持多用户同时操作,同时可以保证并非执行时的效率与速度.
* 可以打印用户缴费等信息.

**不足之处**

1. 相对来讲不够在用户UI界面上的处理不够细致,使用了经典的模式与用户进行交互.
2. 使用的是HTTP协议,由于时间关系,没有实现用HTTPS发送报文向服务器,只是在服务器端完成了HTTPS的初步架构和工作.
3. 对于用户的输入没有保持完全怀疑的态度,在输入的处理上需要再进行一步的改进与创新.

# 附件: