卡通画

描述已自动生成

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 项目名称： |  |
|  |  |
| 姓 名： | 黄俊淇 |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1804 |
| 学 号： | U201814604 |
| 指导教师： |  |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2020年5月20日

**教师评分页**

|  |  |
| --- | --- |
| 子目标 | 子目标评分 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
|  |  |

目 录

[1课程任务概述 1](#_Toc72347673)

[2软件功能学习 2](#_Toc72347674)

[2.1 任务要求 2](#_Toc72347675)

[2.2 完成过程 2](#_Toc72347676)

[2.3任务总结 2](#_Toc72347677)

[3 SQL练习 3](#_Toc72347678)

[3.1 任务要求 3](#_Toc72347679)

[3.2 完成过程 3](#_Toc72347680)

[3.3任务总结 3](#_Toc72347681)

[4 应用系统设计 4](#_Toc72347682)

[4.1系统设计目标 4](#_Toc72347683)

[4.2 需求分析 4](#_Toc72347684)

[4.3 总体设计 4](#_Toc72347685)

[4.4 数据库设计 4](#_Toc72347686)

[4.5 详细设计与实现 4](#_Toc72347687)

[4.6 系统测试 4](#_Toc72347688)

[4.7 系统设计与实现总结 4](#_Toc72347689)

[5课程总结 5](#_Toc72347690)

[附录：实验源代码 6](#_Toc72347691)

[参考文献 7](#_Toc72347692)

# 1课程任务概述

**实验一 软件功能学习**

1.使用Navicat Premium平台进行实验

2.练习数据导入、数据导出、数据备份、新增用户（包含分配权限）

**实验二 Sql练习**

1.建表

2.数据更新

3.数据查询

4.了解系统的查询性能分析功能（选做）

5. DBMS函数及存储过程和事务（选做）

# 2软件功能学习

## 2.1 任务要求

完成下列1~2题，并在实践报告中叙述过程，可适当辅以插图（控制在A4三页篇幅以内）

1）练习SQL Server或其他某个主流关系数据库管理系统软件的备份方式：数据和日志文件的脱机备份、系统的备份功能。

2）练习在新增的数据库上增加用户并配置权限的操作，通过用创建的用户登录数据库并且执行未经授权的SQL语句验证自己的权限配置是否成功。

## 2.2 完成过程

实验环境如下：

操作系统:macOS Big Sur

数据库:MySQL

交互图形界面:Navicat Premium

### 2.2.1 数据库备份

热备：直接使用navicat自带的备份功能即可，如下图所示。

点击备份即可，备份成功如下图所示

文本

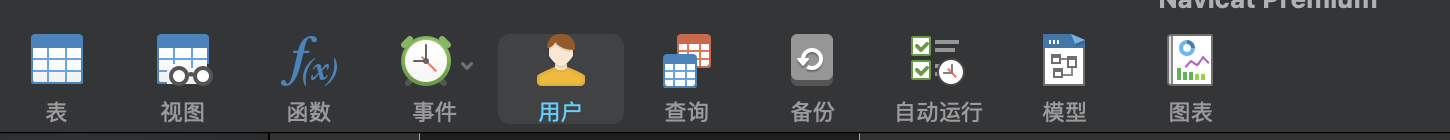
描述已自动生成

备份完成。

冷备：不用登陆mysql，直接找到mysql相应的文件夹，物理备份即可。

### 2.2.2 配置权限

同样在navicat中集成了相应的创建用户和配置权限的功能，如下图所示



进入创建用户界面，如下图所示

电脑屏幕的手机截图

描述已自动生成

在这个界面中还可以直接配置权限，如下图所示

图形用户界面, 文本, 应用程序

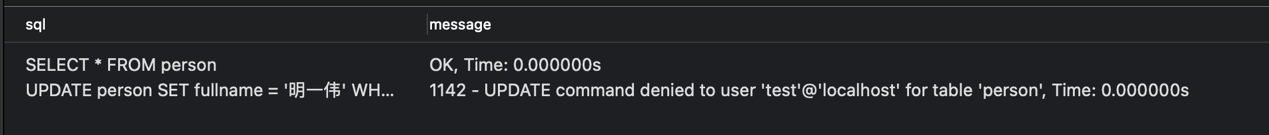
描述已自动生成

在这里我给了修改，创建，索引，查询，插入的权限，而没有给其他权限，用select和update进行测试，sql语句如下图所示

图形用户界面, 文本, 应用程序

描述已自动生成

测试结果如下图所示



可以看到select执行成功，但update因为没有权限失败。

## 2.3任务总结

本次实验使用了集成度非常高的navicat软件，使得备份和用户功能变得极为简单，不需要自己写sql语句，省去了大量时间。

# 3 SQL练习

## 3.1 任务要求

假设在某个区域内的所有地点都存储在地点表中。该地区中的所有人员信息存储在人员表中。根据收集到的人员行程建立了疫情期间某个地区的人员行程表。

该地区的部分人员进行了核酸检测，检测结果保存在诊断表中。根据诊断表中的检测结果，对新冠确诊和无症状感染者的密切接触者进行隔离，密切接触者的信息存储到密切接触表中，隔离信息存储到隔离表中。隔离地点共设置了4个，并且每个隔离地点都有容量限制，隔离点的信息存储在隔离地点表中。

完成下列任务：

1. 建表
2. 数据更新
3. 数据查询
4. 了解系统的查询优化功能
5. DBMS函数以及存储过程

## 3.2 完成过程

### 3.2.1 建表

（1）根据每个表对应的要求建立相应基本表，sql语句见附录，如图3.1所示

电脑屏幕截图

描述已自动生成

图3.1 建表

（2）观察性实验：验证在设置外码时是否一定要参考被参照关系的主码，并在实验报告中简述过程和结果。

person中fullname不是主码，因此使用fullname作为外码建表尝试，结果如图3.2所示

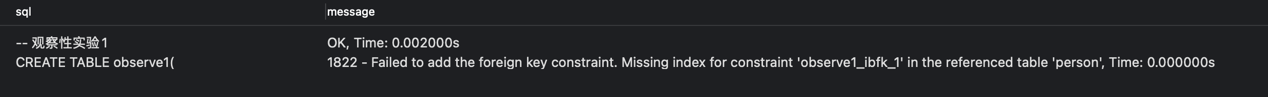


图3.2 观察性实验1

可以看到建表失败，因为fullname不是主码。

（3）数据准备

使用老师给的sql文件，运行即可。

### 3.2.2 数据更新

1. **分别用一条sql语句完成对人员表基本的增、删、改的操作；**

使用头歌平台完成该任务，结果如图3.3所示

文本

描述已自动生成

图3.3 数据增删改

1. **批处理操作**

将行程表中所有到达地点2的记录插入到新表location\_record\_2中。

使用下列sql语句即可

CREATE TABLE location\_record\_2 AS(

SELECT \* FROM itinerary

WHERE loc\_id = 2

);

结果如图3.4所示

图片包含 文本

描述已自动生成

图3.4 批处理

1. **数据导入导出**

使用navicat集成的导入导出功能即可，导出结果如图3.5所示

图形用户界面, 应用程序, Teams

描述已自动生成

图3.5 导出结果

1. **观察性实验**

如果不设置主码，可插入重复元组。sql语句见附录，如图3.6所示。

图片包含 图形用户界面

描述已自动生成

图3.6 观察性实验2

1. **触发器实验**

在diagnose\_record表上设置触发器，触发时机为AFTER update和insert，对每一行设置行触发器，更新isolation\_record中的字段即可。使用头歌平台测试，结果如图3.7所示

delimiter ;;

DROP TRIGGER IF EXISTS mytrigger1;

CREATE TRIGGER mytrigger1

AFTER UPDATE ON diagnose\_record

FOR EACH ROW

BEGIN

UPDATE isolation\_record SET state = 3 WHERE(p\_id = new.p\_id AND new.result = 1);

END;;

delimiter ;;

DROP TRIGGER IF EXISTS mytrigger2;

CREATE TRIGGER mytrigger2

AFTER INSERT ON diagnose\_record

FOR EACH ROW

BEGIN

IF(new.result = 1)

THEN

BEGIN

INSERT INTO isolation\_record VALUES(new.id,new.p\_id,NULL,NULL,3,3);

END;

ELSE INSERT INTO isolation\_record VALUES(new.id,new.p\_id,NULL,NULL,3,1);

END IF;

END;;

文本, 信件

描述已自动生成

图3.7 触发器

### 3.2.3 数据查询

查询测试统一使用头歌平台进行测试，sql语句统一见附录。

1. 查询截至目前的人流量大于30的地点，并按照从高到低排序

使用group by和having可以统计满足条件的分组，使用order by进行排序，嵌套查询即可，结果如图3.8所示

select location\_name,count(loc\_id) as visitors

from location,itinerary

where location.id = itinerary.loc\_id

group by location.id

having count(loc\_id)>30

order by visitors desc,location\_name;



图3.8 查询人流量大于30的地点

1. 查询出每个隔离地中正在进行隔离的人数，并按数量由多到少排序

使用派生表以及嵌套查询即可，统计人数使用聚集函数count，结果如图3.9所示

select location\_name,count(p\_id) as number

from isolation\_record,isolation\_location

where isolation\_location.id = isolation\_record.isol\_loc\_id

and state = 1

group by isolation\_location.id

order by number desc,location\_name;



图3.9 每个隔离地中正在进行隔离的人数

1. 查询行程表中连续行程中的一对行程地点（例如，方斯雪，在2021-02-02 03:02:12至2021-02-02 03:51:12 在博物馆中，2021-02-02 03:51:12 至 2021-02-02 04:23:12出现超市中，输出【姓名，重合时间，起始地点id，起始地点，结束地点id，结束地点】，查询结果如下图所示）

图形用户界面, 应用程序

描述已自动生成

自身链接查询，使用表别名进行区分，结果图3.10所示

select person.id,

person.fullname,

person.telephone,

t1.e\_time as reclosing\_time,

t1.loc\_id as loc1,

l1.location\_name as address1,

t2.loc\_id as loc2,

l2.location\_name as address2

from person,

location l1,

location l2,

itinerary t1,

itinerary t2

where person.id > 30

and person.id = t1.p\_id

and l1.id = t1.loc\_id

and l2.id = t2.loc\_id

and t1.p\_id = t2.p\_id

and t1.e\_time = t2.s\_time

order by person.id, reclosing\_time;



图3.10 接续行程

1. 查询贾涵山的行程情况

因为头歌平台与实验任务书中有区别，结果如图3.11所示

select a.fullname, a.telephone, tmp.location\_name, tmp.s\_time, tmp.e\_time

from person a left join (

select b.fullname, location\_name, s\_time, e\_time

from person b,

location,

itinerary

where b.id = itinerary.p\_id

and itinerary.loc\_id = location.id

) tmp on a.fullname = tmp.fullname

where a.fullname = "贾涵山"

order by a.id desc;

注意left join联合on使用。



图3.11贾涵山的行程情况

1. 查询地名中带有‘店’字的地点名称

select id, location\_name

from location

where location\_name like "%店%"

order by id;

使用like即可，结果如图3.12所示



图3.12 地名中带有‘店’字的地点名称

1. 新发现一位确诊者，已知他在2021.2.2日20:05:40到21:25:40之间在“活动中心”，查询他接触到的人员的名字

select fullname, telephone

from person,

itinerary,

location

where person.id = itinerary.p\_id

and itinerary.loc\_id = location.id

and location\_name = "活动中心"

and (itinerary.s\_time between "2021-02-02 20:05:40" and "2021-02-02 21:25:40"

or itinerary.e\_time between "2021-02-02 20:05:40" and "2021-02-02 21:25:40"

)

order by fullname;

使用between and语句，结果如图3.13所示。



图3.13 确诊者的接触者

1. 查询正在使用的隔离区名（使用DISTINCT关键字）

select location\_name

from isolation\_location

where exists(

select \*

from isolation\_record

where isolation\_location.id = isolation\_record.isol\_loc\_id

and state = 1

)

order by id;

使用exists关键字，结果如图3.14所示



图3.14 正在使用的隔离点

1. 查询有出行记录的人员（使用EXISTS关键字）

结果如图3.15所示

select fullname, telephone

from person

where exists(

select \*

from itinerary

where person.id = itinerary.p\_id

)

order by person.id

limit 30

;



图3.15 有出行记录的人员

1. 查询没有达到地点“Today便利店”的人数（使用NOT EXISTS 关键字），结果如图3.16所示

select count(person.id) as number

from person

where not exists(

select \*

from itinerary,

location

where person.id = itinerary.p\_id

and itinerary.loc\_id = location.id

and location.location\_name = "Today便利店"

);



图3.16没有达到地点“Today便利店”的人数

1. 查询去过所有地点的人员

全称量词使用双重否定表示肯定，结果如图3.17所示

select fullname

from person

where not exists(

select \*

from location

where location.id not in (

select itinerary.loc\_id

from itinerary

where itinerary.loc\_id = location.id

and person.id = itinerary.p\_id

)

)

order by fullname;



图3.17 去过所有地点的人员

1. 新建一个视图，里面有每个隔离点的id,名称，已用房间，容量

结果如图3.18所示

create view isolation\_location\_status as

select l.id, l.location\_name, l.capacity, ifnull(tmp.cut, 0) as occupied

from isolation\_location l left join (

select r.isol\_loc\_id, count(r.isol\_loc\_id) as cut

from isolation\_record r

where r.state = 1

group by r.isol\_loc\_id

) tmp

on l.id = tmp.isol\_loc\_id;



图3.18 隔离点的现状视图

1. 从视图中查询还有空房间的隔离点的剩余房间数目

使用表达式作为查询结果，结果如图3.19所示

select location\_name, capacity - occupied as available\_rooms

from isolation\_location\_status

order by id;



图3.19 各隔离点的剩余房间数

1. 查询靳宛儿传染的人有哪些

结果如图3.20所示

select fullname, telephone

from person,

itinerary,

(

select i.loc\_id, i.s\_time, i.e\_time

from person p,

itinerary i

where p.id = i.p\_id

and p.fullname = "靳宛儿"

) tmp

where person.fullname <> "靳宛儿"

and person.id = itinerary.p\_id

and itinerary.loc\_id = tmp.loc\_id

and (

itinerary.s\_time between tmp.s\_time and tmp.e\_time

or itinerary.e\_time between tmp.s\_time and tmp.e\_time

or tmp.s\_time between itinerary.s\_time and itinerary.e\_time

or tmp.e\_time between itinerary.s\_time and itinerary.e\_time

)

order by fullname;

****

图3.20 靳宛儿传染的人

1. 查询每个地点的密切接触者的数量

结果如图3.21所示

select location\_name, count(loc\_id) as close\_contact\_number

from location,

close\_contact

where location.id = close\_contact.loc\_id

group by close\_contact.loc\_id

order by close\_contact\_number desc, location\_name;



图3.21 每个地点的密切接触者的数量

1. 查询感染人数最多的用户的名字和感染人数

使用limit关键字，结果如图3.22所示

select case\_p\_id, fullname, count(case\_p\_id) as infected\_number

from close\_contact,

person

where person.id = close\_contact.case\_p\_id

group by case\_p\_id

order by infected\_number desc

limit 1;



图3.22 感染人数最多的用户

1. 查询2021-02-02 10:00:00到14:00:00，行程记录最频繁的五个人及行程记录数目

因头歌平台与实验任务书有所区别，sql语句如下，结果如图3.23所示

select \*

from (

select fullname, count(p\_id) as record\_number

from person,

itinerary

where person.id = itinerary.p\_id

and (itinerary.s\_time between "2021-02-02 10:00:00" and "2021-02-02 14:00:00"

or itinerary.e\_time between "2021-02-02 10:00:00" and "2021-02-02 14:00:00"

or "2021-02-02 10:00:00" between itinerary.s\_time and itinerary.e\_time)

group by itinerary.p\_id) tmp

order by record\_number desc, fullname

limit 5;



图3.23 行程记录最频繁的五个人

### 3.2.4 了解系统的查询优化功能

使用EXPLAIN关键字可以看到某个查询的查询过程，sql语句见附录，结果如图3.24所示，其中对应列的取值意义如下：

**一、 id**

SELECT识别符。这是SELECT的查询序列号

我的理解是SQL执行的顺序的标识，SQL从大到小的执行

1. id相同时，执行顺序由上至下

2. 如果是子查询，id的序号会递增，id值越大优先级越高，越先被执行

3. id如果相同，可以认为是一组，从上往下顺序执行；在所有组中，id值越大，优先级越高，越先执行

**二、select\_type**

示查询中每个select子句的类型

(1) SIMPLE(简单SELECT，不使用UNION或子查询等)

(2) PRIMARY(子查询中最外层查询，查询中若包含任何复杂的子部分，最外层的select被标记为PRIMARY)

(3) UNION(UNION中的第二个或后面的SELECT语句)

(4) DEPENDENT UNION(UNION中的第二个或后面的SELECT语句，取决于外面的查询)

(5) UNION RESULT(UNION的结果，union语句中第二个select开始后面所有select)

(6) SUBQUERY(子查询中的第一个SELECT，结果不依赖于外部查询)

(7) DEPENDENT SUBQUERY(子查询中的第一个SELECT，依赖于外部查询)

(8) DERIVED(派生表的SELECT, FROM子句的子查询)

(9) UNCACHEABLE SUBQUERY(一个子查询的结果不能被缓存，必须重新评估外链接的第一行)

**三、table**

显示这一步所访问数据库中表名称（显示这一行的数据是关于哪张表的），有时不是真实的表名字，可能是简称，例如上面的e，d，也可能是第几步执行的结果的简称

**四、type**

对表访问方式，表示MySQL在表中找到所需行的方式，又称“访问类型”。

常用的类型有： ALL、index、range、 ref、eq\_ref、const、system、NULL（从左到右，性能从差到好）

ALL：Full Table Scan， MySQL将遍历全表以找到匹配的行

index: Full Index Scan，index与ALL区别为index类型只遍历索引树

range:只检索给定范围的行，使用一个索引来选择行

ref: 表示上述表的连接匹配条件，即哪些列或常量被用于查找索引列上的值

eq\_ref: 类似ref，区别就在使用的索引是唯一索引，对于每个索引键值，表中只有一条记录匹配，简单来说，就是多表连接中使用primary key或者 unique key作为关联条件

const、system: 当MySQL对查询某部分进行优化，并转换为一个常量时，使用这些类型访问。如将主键置于where列表中，MySQL就能将该查询转换为一个常量，system是const类型的特例，当查询的表只有一行的情况下，使用system

NULL: MySQL在优化过程中分解语句，执行时甚至不用访问表或索引，例如从一个索引列里选取最小值可以通过单独索引查找完成。

**五、possible\_keys**

指出MySQL能使用哪个索引在表中找到记录，查询涉及到的字段上若存在索引，则该索引将被列出，但不一定被查询使用（该查询可以利用的索引，如果没有任何索引显示 null）

**六、Key**

key列显示MySQL实际决定使用的键（索引），必然包含在possible\_keys中

**七、key\_len**

表示索引中使用的字节数，可通过该列计算查询中使用的索引的长度（key\_len显示的值为索引字段的最大可能长度，并非实际使用长度，即key\_len是根据表定义计算而得，不是通过表内检索出的）

**八、ref**

列与索引的比较，表示上述表的连接匹配条件，即哪些列或常量被用于查找索引列上的值

**九、rows**

估算出结果集行数，表示MySQL根据表统计信息及索引选用情况，估算的找到所需的记录所需要读取的行数

**十、Extra**

该列包含MySQL解决查询的详细信息,有以下几种情况：

Using where:不用读取表中所有信息，仅通过索引就可以获取所需数据，这发生在对表的全部的请求列都是同一个索引的部分的时候，表示mysql服务器将在存储引擎检索行后再进行过滤

Using temporary：表示MySQL需要使用临时表来存储结果集，常见于排序和分组查询，常见 group by ; order by

Using filesort：当Query中包含 order by 操作，而且无法利用索引完成的排序操作称为“文件排序”

Using join buffer：改值强调了在获取连接条件时没有使用索引，并且需要连接缓冲区来存储中间结果。如果出现了这个值，那应该注意，根据查询的具体情况可能需要添加索引来改进能。

Impossible where：这个值强调了where语句会导致没有符合条件的行（通过收集统计信息不可能存在结果）。

Select tables optimized away：这个值意味着仅通过使用索引，优化器可能仅从聚合函数结果中返回一行

No tables used：Query语句中使用from dual 或不含任何from子句

using index ：使用覆盖索引的时候就会出现

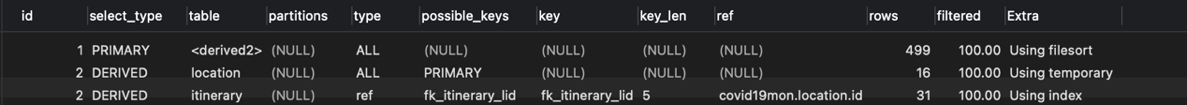


图3.24 性能分析

### 3.2.5 DBMS函数以及存储过程

1) 编写一个依据人员编号计算其达到所有地点的次数的自定义函数，并利用其查询至少到达过3个地点的人员。

输入为某个id，返回为该人员到达所有地点的次数。函数如下所示。

DROP FUNCTION IF EXISTS Count\_Records;

DELIMITER ;;

CREATE FUNCTION Count\_Records(a int)

RETURNS INT

READS SQL DATA

DETERMINISTIC

BEGIN

DECLARE c INT;

SELECT count(\*) INTO c FROM itinerary WHERE p\_id = a;

RETURN c;

END;;

DELIMITER ;

(2) 利用创建的函数，仅用一条SQL语句查询在行程表中至少有3条行程记录的人员信息，查询结果依人员编号排序。

SELECT \* FROM person WHERE Count\_Records(id)>=3;

结果如图3.25所示图片包含 文本

描述已自动生成

图3.25 DBMS函数

2) 尝试编写DBMS的存储过程，建立每个隔离点的人数统计表，并通过存储过程更新该表。sql语句如下：

DROP TABLE IF EXISTS location\_count;

CREATE TABLE location\_count(

loc\_id INT PRIMARY KEY NOT NULL,

times INT NOT NULL,

FOREIGN KEY(loc\_id) REFERENCES isolation\_location(id)

);

DROP PROCEDURE IF EXISTS sumLocId;

DELIMITER ;;

CREATE PROCEDURE sumLocId()

BEGIN

INSERT INTO location\_count SELECT isol\_loc\_id,COUNT(\*) FROM isolation\_record GROUP BY isolation\_record.isol\_loc\_id;

END;;

CALL sumLocId();

测试结果如图3.26所示

图形用户界面, 应用程序, 表格

描述已自动生成

图3.26 DBMS存储过程

## 3.3任务总结

本次实验难点在查询，如何使用select进行对应的查询是非常难的，比如全称量词需要双重否定，又比如要统计某个分组需要使用having而不是where。

此外，可以发现sql也可以有类似函数的功能，可以在函数里实现循环或者判断，减去了重复sql语句带来的冗余，可以更加方便操作数据库。

# 4 应用系统设计

## 4.1系统设计目标

### 4.1.1 系统应用背景

数据库是数据管理的有效技术，是计算机科学的重要分支。今天，信息资源 已成为各个部门的重要财富和资源，建立一个满足各级部门的信息处理要求的行 之有效的信息系统也成为一个企业或组织生存和发展的重要条件。因此，作为信 息处理系统核心和基础的数据库技术得到越来越广泛的应用。从小型单项事务处 理系统到大型信息系统，从联机事务处理到联机分析处理，从一般企业管理到计 算机辅助设计与制造，计算机集成制造系统、电子政务、电子商务、地理信息系 统等，越来越多的应用领域采用数据库技术来存储和处理信息资源。特别是随着互联网的发展，广大用户可以之间访问并使用数据库，例如通过网上订购图书，日用品，机票，火车票。通过网上银行存款取款、检索和管理账户，等等，数据库已经成为每个人生活中不可缺少的部分。

而现在出行驾车是短途路程的优选，其次有驾照的人很多，而买车的人少，因此出现了很多租车公司为相当一部分人群提供服务。因此，本次选题汽车租赁信息系统，采用C/S模式实现。

### 4.1.2 总体目标

采用B/S或C/S模式实现一个汽车租借信息系统。完成用户、车辆、租借情况、交通违规罚款等信息的管理。

要求：

1. 实现不同权限的浏览和更新。
2. 能够根据车辆使用情况计算押金退还金额。
3. 能查看用户正在租借记录，以及进行会员制和非会员制的客户管理。
4. 能够管理车辆信息；
5. 能够生成租借公司的财务报表。
6. 系统需包含事务（包含commit，rollback），存储过程/触发器，视图，函数。

7）需要在程序中体现SQL和编程语言的结合。

### 4.1.3 具体目标

对于用户：

1. 可以登陆软件
2. 可以租车
3. 可以还车
4. 可以退出登录

对于公司：

1. 可以登陆软件
2. 可以管理用户
3. 可以管理车辆
4. 可以查询正在租赁车辆相关信息
5. 可以查询收支记录
6. 可以退出登录

## 4.2 需求分析

租车公司，需求分析如图4.1所示

一些文字和图案

描述已自动生成

图4.1 需求分析

### 4.2.1 数据

由需求分析可以得知需要以下几个表：

1. 租车公司表（公司名，公司密码）
2. 客户等级表（客户ID，租车次数，超时次数，客户等级）
3. 车辆信息表（车辆ID，车辆名，租车价格（按天算），租车押金）
4. 租车记录表（记录ID，客户ID，车辆ID，车辆使用情况，租车日期，截止日期，超时）
5. 用户信息表（客户ID，客户名，客户密码）
6. 收入记录表（收支ID，用户ID，车辆ID，收支时间，收支明细，押金退还金额，租车价格，租车押金）

其中，客户等级=（租车次数-超时次数）/100，

车辆使用情况分为“良好”，“未加满油”，“交通违规”，

若为良好，收支明细=租车价格\*天数/客户等级，退还所有押金

若为未加满油，收支明细=租车价格\*天数/客户等级+100，即押金中扣除100

若为交通违规，收支明细=租车价格\*天数/客户等级+押金

### 4.2.2 数据流

其中，表1、5的数据由用户键入，用于登陆系统。

表2、3、6数据由表4得到，用于计算收入。

表4数据由用户键入，用于客户租借车。

### 4.2.3 数据流图

数据流图顶层如图4.2所示

图示

描述已自动生成

图4.2 顶层数据流

数据流图底层如图4.3所示

图示, 示意图

描述已自动生成

图4.3 底层数据流

## 4.3 总体设计

登陆界面有两个输入框，分别输入用户ID/公司名和密码，还有两个选项选择进入用户/公司界面。

### 4.3.1 用户功能设计

用户功能流程图如图4.2所示。

图示

描述已自动生成

图4.2 用户功能流程图设计

用户登陆后有两个按钮：租车和还车，并且还车界面显示该用户的租车记录。租车时应插入对应租车记录，还车时应计算收支，并删除租车记录。

### 4.3.2 租车公司功能设计

租车公司功能设计流程图如图4.3所示

图示

描述已自动生成

图4.3 租车公司功能设计

公司登陆成功后，有三个按钮对应用户查询，租车记录查询，收支记录查询，车辆信息管理，其中，用户查询有两个按钮可以添加和删除用户；车辆信息管理有两个按钮可以用来添加和删除对应车辆信息。

## 4.4 数据库设计

### 4.4.1 ER图设计

本系统ER图如图4.4所示。

图示, Teams

描述已自动生成

图4.4 ER图

### 4.4.2 数据库表设计

1.租车公司表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 | 备注 |
| cName | 公司名 | VARCHAR（20） | 主码 | not null |
| cPassword | 公司密码 | VARCHAR（20） |  | not null |

2.客户等级表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 | 备注 |
| userID | 客户ID | VARCHAR（20） | 外码 | not null |
| borrowTimes | 租车次数 | INT |  | default=0 |
| overTimes | 超时次数 | INT |  | default=0 |
| userLevel | 客户等级 | INT |  | default=1 |

3.车辆信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 | 备注 |
| carID | 车辆ID | VARCHAR（20） | 主码 | not null |
| carName | 车辆名 | VARCHAR（20） |  | not null |
| carPrice | 租车价格 | VARCHAR（20） |  | not null |
| borrowPrice | 押金 | VARCHAR（20） |  | not null |

4.租车记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 | 备注 |
| recordID | 记录ID | VARCHAR（20） | 主码 | not null |
| userID | 客户ID | VARCHAR（20） | 外码 | not null |
| carID | 车辆ID | VARCHAR（20） | 外码 | not null |
| condition | 车辆使用情况 | VARCHAR（20） |  | not null |
| borrowTime | 租车时间 | date |  |  |
| dueTime | 截止日期 | date |  |  |
| overTime | 超时 | VARCHAR（20） |  | default=否 |

5.用户信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 | 备注 |
| userID | 客户ID | VARCHAR（20） | 主码 | not null |
| username | 客户名 | VARCHAR（20） |  | not null |
| userPassword | 客户密码 | VARCHAR（20） |  | not null |

6.收入记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文说明 | 字段类型 | 完整性约束 | 备注 |
| moneyID | 收支ID | VARCHAR（20） | 主码 | not null |
| userID | 用户ID | VARCHAR（20） |  |  |
| carID | 车辆ID | VARCHAR（20） |  |  |
| moneyTime | 收支时间 | date |  |  |
| moneyDetail | 收支明细 | VARCHAR（20） |  | not null |
| moneyback | 押金退还金额 | VARCHAR（20） |  | not null |
| price | 租车价格 | VARCHAR（20） |  |  |
| back | 押金 | VARCHAR（20） |  |  |

### 4.4.3 触发器设置

（1）当用户注册后，应对应生成用户的等级信息表

DROP TRIGGER IF EXISTS customer\_trigger;

delimiter ;;

CREATE TRIGGER customer\_trigger

AFTER INSERT ON customer

FOR EACH ROW

BEGIN

INSERT INTO customer\_level VALUES(new.userID,DEFAULT,DEFAULT,DEFAULT);

END;;

delimiter ;

（2）当用户还车后，应更新用户等级表以及在收支表中插入对应收支记录。

DROP TRIGGER IF EXISTS borrow\_trigger;

delimiter ;;

CREATE TRIGGER borrow\_trigger

BEFORE DELETE ON borrow

FOR EACH ROW

BEGIN

UPDATE customer\_level SET borrowTimes = borrowTimes + 1 WHERE userID = old.userID;

UPDATE customer\_level SET overTimes = overTimes + 1 WHERE userID = old.userID AND CURRENT\_DATE() > old.dueTime;

UPDATE customer\_level SET userLevel = userLevel+((borrowTimes - overTimes)/100) WHERE userID = old.userID;

IF(old.carCondition = '良好')

THEN

BEGIN

DECLARE price1 VARCHAR(20);

DECLARE back1 VARCHAR(20);

DECLARE level1 INT;

SET price1 = (SELECT borrowPrice FROM car WHERE carID = old.carID);

SET back1 = (SELECT carPrice FROM car WHERE carID = old.carID);

SET level1 = (SELECT userLevel FROM customer\_level WHERE userID = old.userID);

INSERT INTO money(userID,carID,moneyTime,moneyDetail,moneyback,price,back) VALUES(old.userID,old.carID,CURRENT\_DATE()

,CONCAT('+',CONVERT(CONVERT(TO\_DAYS(CURRENT\_DATE())-TO\_DAYS(old.borrowTime),UNSIGNED)\*CONVERT(price1,UNSIGNED)/level1,CHAR(20)))

,back1,price1,back1) ;

END;

END IF;

IF(old.carCondition = '未加满油')

THEN

BEGIN

DECLARE price1 VARCHAR(20);

DECLARE back1 VARCHAR(20);

DECLARE level1 INT;

SET price1 = (SELECT borrowPrice FROM car WHERE carID = old.carID);

SET back1 = (SELECT carPrice FROM car WHERE carID = old.carID);

SET level1 = (SELECT userLevel FROM customer\_level WHERE userID = old.userID);

INSERT INTO money(userID,carID,moneyTime,moneyDetail,moneyback,price,back) VALUES(old.userID,old.carID,CURRENT\_DATE()

,CONCAT('+',CONVERT(CONVERT(TO\_DAYS(CURRENT\_DATE())-TO\_DAYS(old.borrowTime),UNSIGNED)\*CONVERT(price1,UNSIGNED)/level1+100,CHAR(20)))

,CONVERT(CONVERT(back1,UNSIGNED)-100,char(20)),price1,back1) ;

END;

END IF;

IF(old.carCondition = '交通违规')

THEN

BEGIN

DECLARE price1 VARCHAR(20);

DECLARE back1 VARCHAR(20);

DECLARE level1 INT;

SET price1 = (SELECT borrowPrice FROM car WHERE carID = old.carID);

SET back1 = (SELECT carPrice FROM car WHERE carID = old.carID);

SET level1 = (SELECT userLevel FROM customer\_level WHERE userID = old.userID);

INSERT INTO money(userID,carID,moneyTime,moneyDetail,moneyback,price,back) VALUES(old.userID,old.carID,CURRENT\_DATE()

,CONCAT('+',CONVERT(CONVERT(TO\_DAYS(CURRENT\_DATE())-TO\_DAYS(old.borrowTime),UNSIGNED)\*CONVERT(price1,UNSIGNED)/level1+CONVERT(back1,UNSIGNED),CHAR(20)))

,'0',price1,back1) ;

END;

END IF;

END;;

## 4.5 详细设计与实现

使用Java swing实现交互界面，使用mysql数据库存储数据，使用JDBC操作数据库。

## 4.6 系统测试

## 4.7 系统设计与实现总结

# 5课程总结

# 附录：实验源代码

**建表**

-- 请在以下适当的空白位置填写SQL语句完成任务书的要求。空白位置不够的话，可以通过回车换行增加。

-- 表1 人员表(person)

DROP TABLE IF EXISTS person;

create table person(

id int,

fullname char(20) not null,

telephone char(11) not null,

constraint pk\_person primary key (id)

);

-- 表2 地点表(location)

DROP TABLE IF EXISTS location;

create table location(

id int,

location\_name char(20) not null,

constraint pk\_location primary key (id)

);

-- 表3 行程表（itinerary）

DROP TABLE IF EXISTS itinerary;

create table itinerary(

id int,

p\_id int,

loc\_id int,

s\_time datetime,

e\_time datetime,

constraint fk\_itinerary\_pid foreign key (p\_id) references person(id),

constraint fk\_itinerary\_lid foreign key (loc\_id) references location(id),

constraint pk\_itinerary primary key (id)

);

-- 表4 诊断表（diagnose\_record）

DROP TABLE IF EXISTS diagnose\_record;

create table diagnose\_record (

id int,

p\_id int,

diagnose\_date datetime,

result int,

constraint fk\_diagnose\_pid foreign key (p\_id) references person(id),

constraint pk\_diagnose\_record primary key (id)

);

-- 表5 密切接触者表（close\_contact）

DROP TABLE IF EXISTS close\_contact;

create table close\_contact (

id int,

p\_id int,

contact\_date datetime,

loc\_id int,

case\_p\_id int,

constraint fk\_contact\_pid foreign key (p\_id) references person(id),

constraint fk\_contact\_lid foreign key (loc\_id) references location(id),

constraint fk\_contact\_caseid foreign key (case\_p\_id) references person(id),

constraint pk\_close\_contact primary key (id)

);

-- 表6 隔离地点表（isolation\_location）

DROP TABLE IF EXISTS isolation\_location;

create table isolation\_location (

id int,

location\_name char(20),

capacity int,

constraint pk\_isolation\_loc primary key (id)

);

-- 表7 隔离表（isolation\_record）

DROP TABLE IF EXISTS isolation\_record;

create table isolation\_record (

id int,

p\_id int,

s\_date datetime,

e\_date datetime,

isol\_loc\_id int,

state int,

constraint fk\_isolation\_pid foreign key (p\_id) references person(id),

constraint fk\_isolation\_lid foreign key (isol\_loc\_id) references isolation\_location(id),

constraint pk\_isolation primary key (id)

);

-- 代码结束

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

# 参考文献

[1] 王珊 萨师煊. 数据库系统概论(第 5 版)高等教育出版社，2014.

[2] Abraham Silberschatz Henry F. Korth S.Sudarshan. 数据库系统概念(第 6 版) 杨冬青 李红燕 唐世渭，译. 机械工业出版社，2013.

[3] Lynn Beighley Michael Morrison. Head First PHP & MySQL 苏金国 徐阳，译. 中国电力出版社，2011

[4] Nicholas C. Zakas. JavaScript 高级程序设计(第 3 版) 李松峰 曹力，译. 人 民邮电出版社，2016.

[5] W3School 中文文档:http://www.w3school.com.cn/