

平顶山学院

课程设计报告

2021-2022 学年第一学期

课	程	数据库原理与应用课程设计
课 程 设 计 题 目		医院信息管理系统
院	(系) :	计算机学院 (软件学院)
姓	名	王佳慧
学	号	201530404
专 业 班 级		2020 级鲲鹏 2 班
指 导 教 师		徐向艺

2021 年 12 月 8 日

目录

摘要	3
1. 1 概述	3
1.2 运行环境	3
2. 1 需求分析	4
2.2 可行性分析	4
3.1 概念结构设计	5
3.2 设计分 E-R 图	5
3.3.1 全局 E-R 图	7
4.1 逻辑结构设计	7
5.1 数据库物理设计与实施	8
6 . 数据操作要求及实现	12
6.2 视图	23
6.3 触发器	24
6.4 存储过程	26
7.基准测试	27
8 . 总结	28
9.参考文献	28

医院信息管理系统

摘要

随着人们生活水平的不断提高,人们更重视医疗设施,医院的业务也不断增加,而对于医院来讲,信息管理系统属于其重要组成部分,确保系统高效、稳定且安全的运行是医院应关注和解决的问题,为使信息系统稳定、安全,高可用,工作人员需管理好信息系统,重视数据库安全,进而使信息数据将自身作用与价值充分发挥出来。医疗关系民脂民生,人民健康.设计人员应当格外小心谨慎,做好零失误.

1.1 概述

依据数据库课程设计要求,我将以 DBA 标准,参照 mysql 高性能, java 开发手册,数据库系统概论等编写设计本系统.由于考虑诸多因素,对数据库的了解结合我目前的经验水平,我将使用 java 连接 mysql 数据库. shell 脚本, jmeter 测试工具进行测试. 本系统使用 mysql 的原因是其在 Linux 下比 sql server 更成熟,同时它具有更高的灵活性. 选用 java 作为数据库连接,因为我对它相较其他语言足够熟悉,同时它也足够强大. shell 和 jmeter 都能很好的辅助我进行基准测试. 以 DBA 的要求,应当多使用逻辑外键,少触发器,存储过程和外键级联操作. 但我会将两种方案一并给出. 具体设计方案详见下文.

1.2 运行环境

mysql-connector-java-8.0.22.jar

Window10

IntelliJ IDEA 2021.2.1

apache-jmeter-5.4.1

2.1 需求分析

2.1.1 基本分类需求分析

1. 控制中心

- 人员管理:实现对员工,病人的增删改;
- 药品管理:实现对药品的入库,出库操作和处方药的登记以及药品类型,药品信息的登记;
- 收费管理:对药品营收的统计;

2. 查询

- 员工,病人的基本信息查询;
- 药品信息的查询;
- 收费情况的查询;

2.1.2 主要关系流程分析

病人看病,先挂号等待分配科室.然后医师给患者看病开票据.病人拿票凭去前台充值,接着去药房拿药.药师根据患者的票凭刷卡取药,完成药品交付和收费流程.

2.2 可行性分析

该系统主要包括基本数据维护、基本业务、数据库管理和信息查询四部分。

1、基本数据维护部份应包括提供管理员添加、修改并维护基本的数据途径。例如添加修改医院和办理病人入住与搬出或者换病房，管理医院里的基本设备。

2、数据库管理部分是对这个数据库的管理，包括医生，病人详细信息等。

3、该系统的技术可行性分析:在系统维护中包括医生和病人信息检索，数据库信息维护。

4、系统技术的可行性分析：

基于 jvm 和 mysql 下本系统可以运行于 windows 和 Linux 操作系统当中,可以为系统提供一个稳定的运行环境。该系统应该说有开发的必要性。

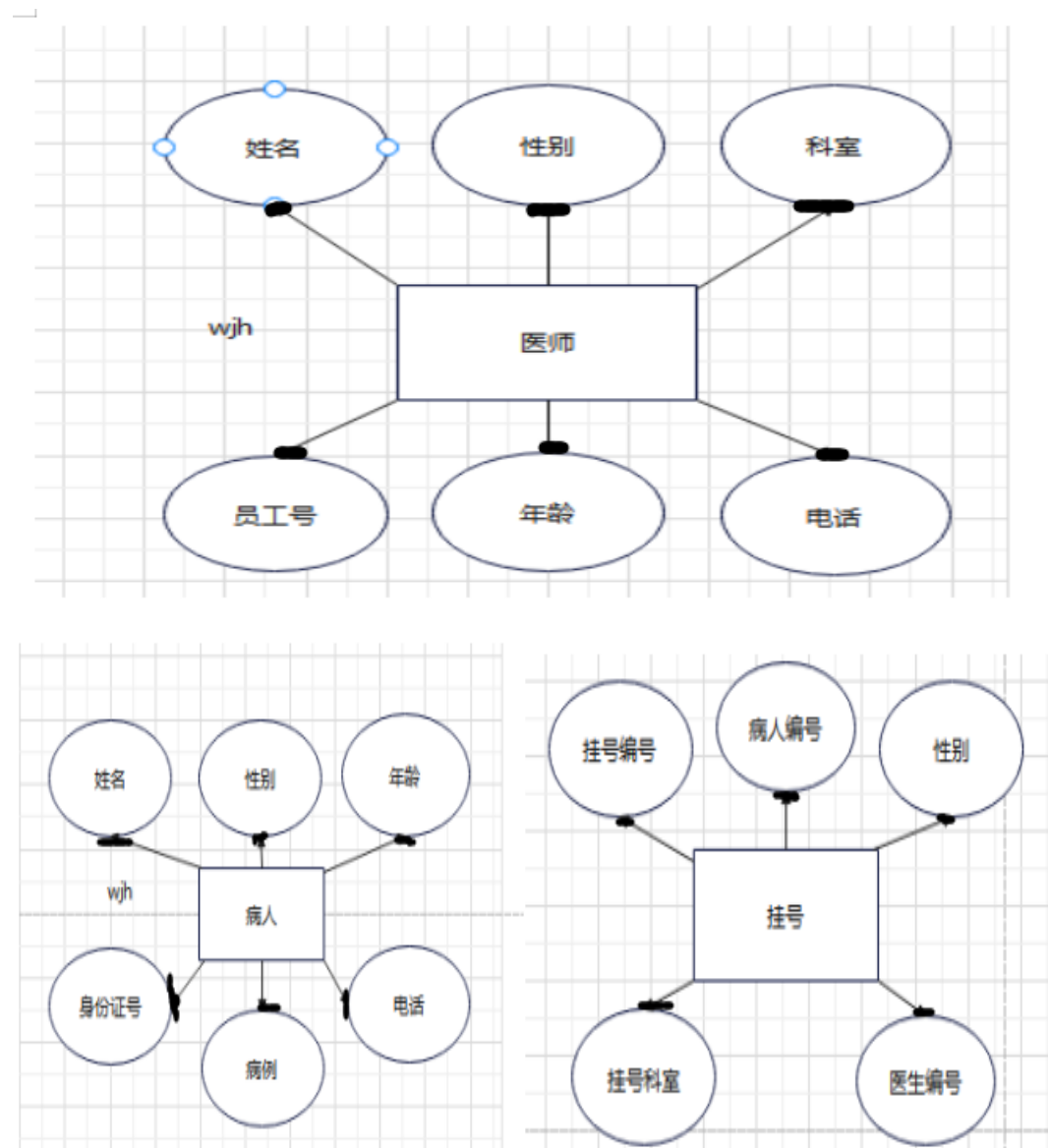
该系统主要由两大部分组成即管理维护和查询。

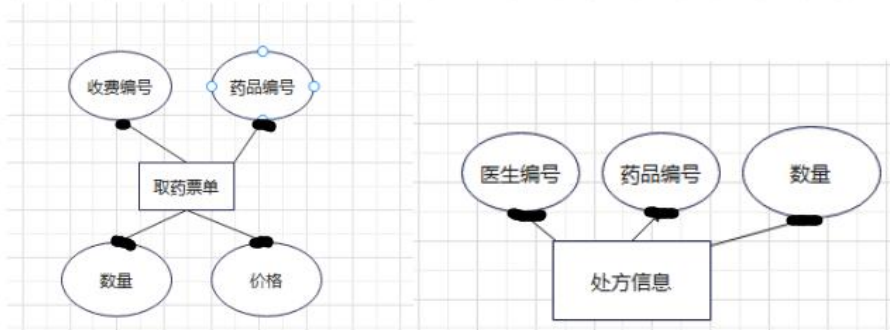
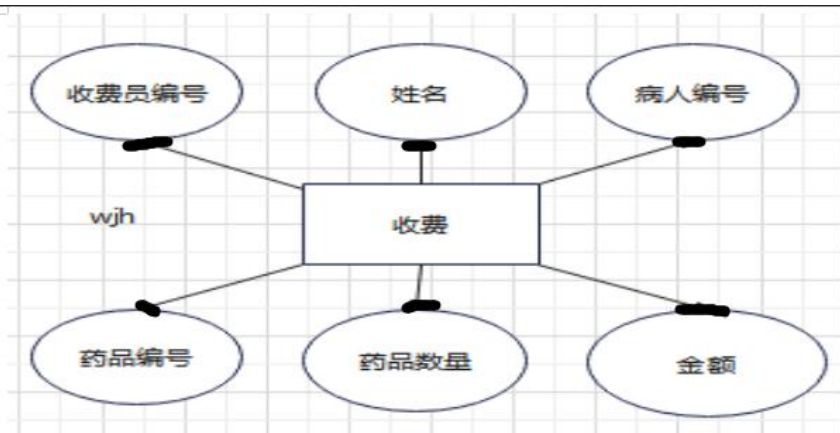
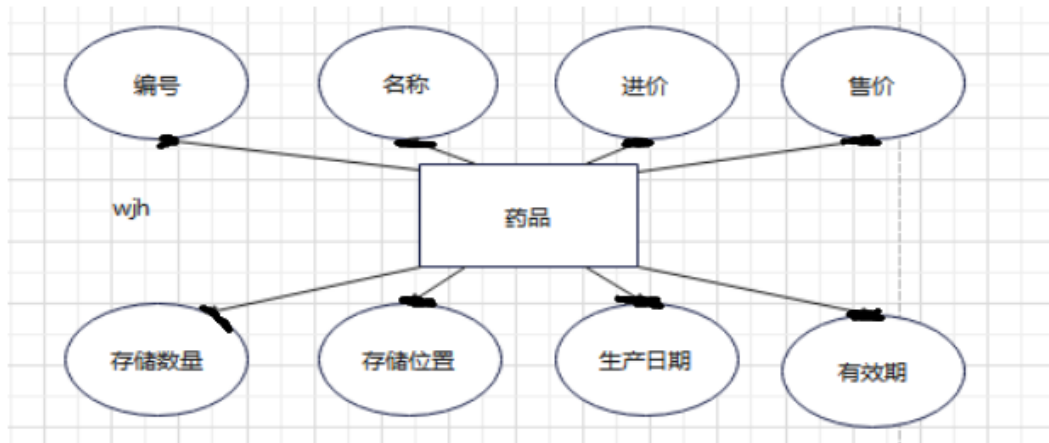
3.1 概念结构设计

3.1.1 抽象出系统的实体

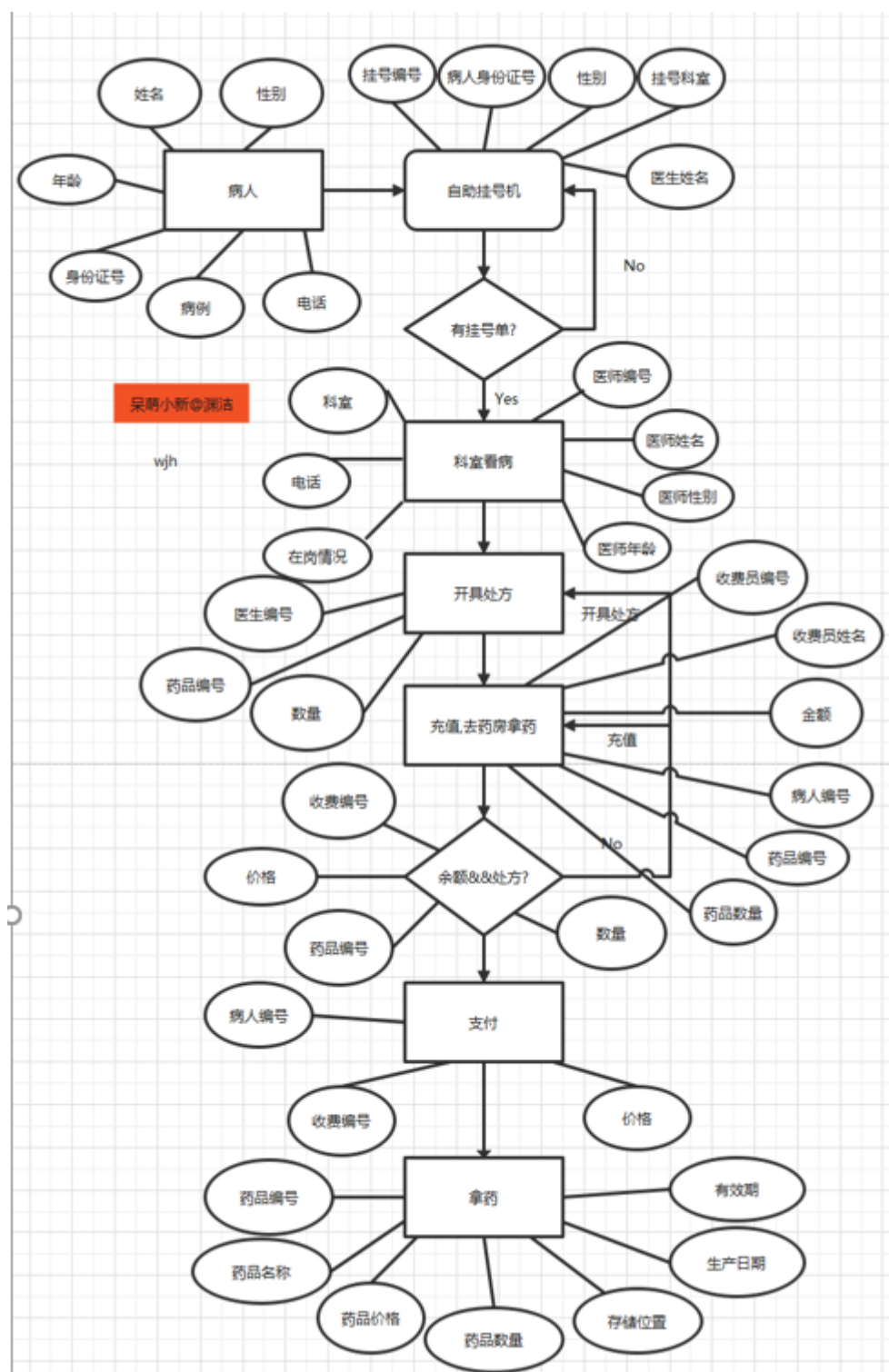
涉及的目标对象: 患者, 科室, 医师, 药师, 前台收银, 药品, 挂号单, 处方信息.

3.2 设计分 E-R 图





3.3.1 全局 E-R 图



4.1 逻辑结构设计

病人病史(身份证号、姓名、性别、年龄、病例);

药品存放记录(药品编号、药品名、进价、售价、药品数量、生产日期、有效期, 存储位置);

挂号(挂号编号、病人编号、性别、挂号科室、医生编号);

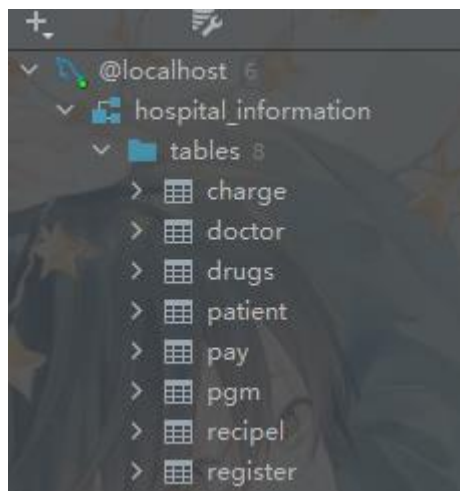
收费(收费员编号, 收费员姓名, 病人编号, 药品编号, 数量, 金额);

医生(医生员工号, 医生姓名, 医生性别, 医生年龄, 科室, 电话);

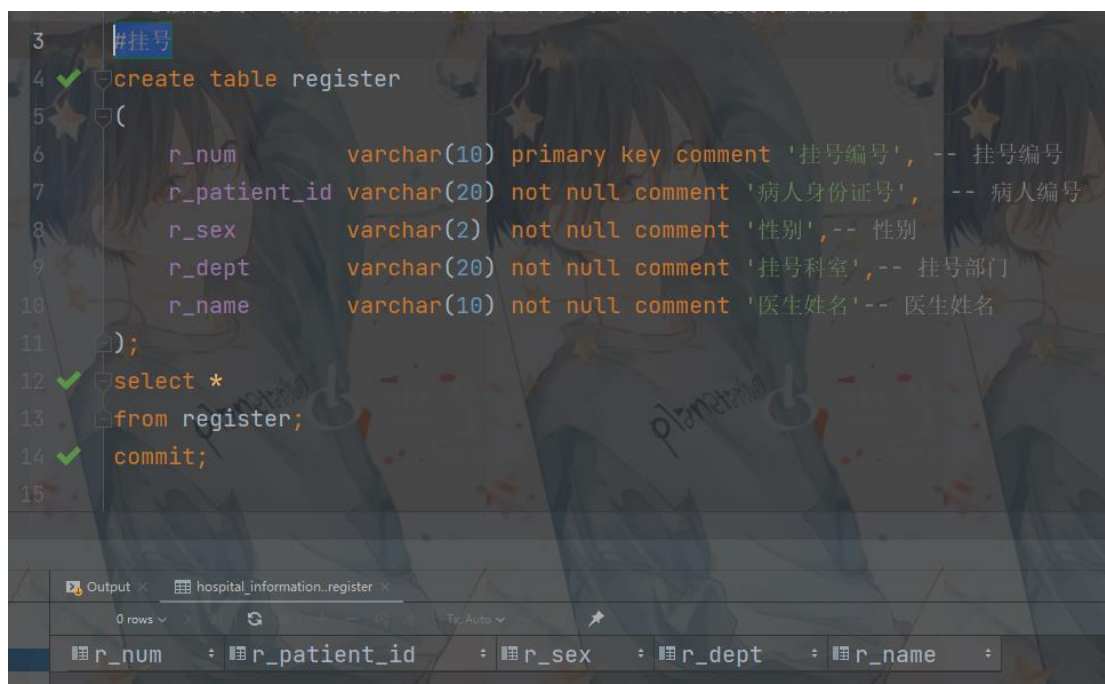
处方(医生员工号, 药品编号, 数量);

支付(支付编号, 收费编号, 价格);

5.1 数据库物理设计与实施



#挂号



医师信息


```
16 # 医师信息
17 create table doctor
18 (
19     d_doctor_id varchar(10) primary key comment '医生编号',
20     d_name      varchar(20)      not null comment '医生姓名',
21     d_sex       varchar(2)       not null comment '医生性别',
22     d_age       tinyint(4) unsigned not null comment '医生年龄',
23     d_dept      varchar(50)      not null comment '科室',
24     d_tel       varchar(20)      not null comment '电话',
25     is_jobbing  tinyint unsigned default 0 not null comment '0为医生不在岗'
26 );
27 select *
28 from doctor;
29 commit;
```

Output | hospital_information.doctor

d_doctor_id	d_name	d_sex	d_age	d_dept	d_tel	is_jobbing
-------------	--------	-------	-------	--------	-------	------------

病人信息

```
1 # 病人信息
2 create table patient
3 (
4     p_patient_id varchar(20) primary key comment '病人身份证号',
5     p_name       varchar(20)      not null comment '病人姓名',
6     p_age       tinyint(4) unsigned not null comment '病人年龄',
7     p_sex       varchar(2)       not null comment '病人性别',
8     p_tel       varchar(20)      not null comment '病人电话',
9     p_inf       varchar(50)      not null comment '病例'
10 );
11 select *
12 from patient;
13 commit;
```

Output | hospital_information.patient

p_patient_id	p_name	p_age	p_sex	p_tel	p_inf
--------------	--------	-------	-------	-------	-------

药品信息

```
create table drugs
(
    drug_id      varchar(10) primary key comment '药品编号',
    drug_name    varchar(50)      not null comment '药品名称',
    drug_price   decimal unsigned not null comment '药品价格',
    drug_quantity varchar(20)      not null comment '药品数量',
    drug_storage varchar(50)      not null comment '存储位置',
    drug_date    datetime         not null comment '生产日期',
    usefull_life datetime         not null comment '有效期'
);
select *
from drugs;
commit;
```

Output | hospital_information.drugs

drug_id	drug_name	drug_price	drug_quantity	drug_storage	drug_date	usefull_life
---------	-----------	------------	---------------	--------------	-----------	--------------

缴费信息

```
60 # 缴费信息
61 create table charge
62 (
63     toll_id      varchar(10) comment '收费员编号',
64     t_name       varchar(10) not null comment '收费员姓名',
65     patient_id   varchar(10) comment '病人编号',
66     drug_id      varchar(10) comment '药品编号',
67     drug_quantity varchar(20) not null comment '药品数量',
68     amount       decimal      not null comment '金额',
69     primary key (toll_id, patient_id, drug_id)
70 );
71 select *
72 from charge;
73 commit;
```

Output × hospital_information_charge ×

0 rows

toll_id	t_name	patient_id	drug_id	drug_quantity	amount
---------	--------	------------	---------	---------------	--------

取药票单

```
74
75 # 取药票单
76 create table PGM
77 (
78     t_id      varchar(10) comment '收费编号',
79     drug_id   varchar(10) comment '药品编号',
80     quantity  varchar(10)      not null comment '数量',
81     price     decimal unsigned not null comment '价格',
82     primary key (t_id, drug_id)
83 );
84 select *
85 from PGM;
86 commit;
```

Output × hospital_information_pgm ×

0 rows

t_id	drug_id	quantity	price
------	---------	----------	-------

处方信息

```
87
88 # 处方信息
89 ✓ create table recipel
90 (
91     doctor_id varchar(10) comment '医生编号',
92     drug_id   varchar(10) comment '药品编号',
93     count     varchar(4) not null comment '数量',
94     primary key (doctor_id, drug_id)
95 );
96 ✓ select *
97   from recipel;
98 ✓ commit;
99
100 # 支付凭据
```

Output × hospital_information..recipel ×

0 rows

doctor_id ÷ drug_id ÷ count ÷

支付凭据

```
99
100 # 支付凭据
101 ✓ create table pay
102 (
103     patient_id varchar(10) comment '病人编号',
104     t_id       varchar(10) comment '收费编号',
105     price      decimal not null comment '价格',
106     primary key (patient_id, t_id)
107 );
108 ✓ select *
109   from pay;
110 ✓ commit;
```

Output × hospital_information..pay ×

0 rows

patient_id ÷ t_id ÷ price ÷

drugs	
drug_id /* 药品编号 */	varchar(10)
drug_name /* 药品名称 */	varchar(50)
drug_price /* 药品价格 */	decimal(10) unsigned
drug_quantity /* 药品数量 */	varchar(20)
drug_storage /* 存储位置 */	varchar(50)
drug_date /* 生产日期 */	datetime
usefull_life /* 有效期 */	datetime

doctor	
d_octor_id /* 医生编号 */	varchar(10)
d_name /* 医生姓名 */	varchar(20)
d_sex /* 医生性别 */	varchar(2)
d_age /* 医生年龄 */	tinyint unsigned
d_dept /* 科室 */	varchar(50)
d_tel /* 电话 */	varchar(20)
is_jobing /* 0为医生不在岗 */	tinyint unsigned

patient	
p_atient_id /* 病人身份证号 */	varchar(20)
p_name /* 病人姓名 */	varchar(20)
p_age /* 病人年龄 */	tinyint unsigned
p_sex /* 病人性别 */	varchar(2)
p_tel /* 病人电话 */	varchar(20)
p_inf /* 病例 */	varchar(50)

charge	
toll_id /* 收费员编号 */	varchar(10)
patient_id /* 病人编号 */	varchar(10)
drug_id /* 药品编号 */	varchar(10)
t_name /* 收费员姓名 */	varchar(10)
drug_quantity /* 药品数量 */	varchar(20)
amount /* 金额 */	decimal(10)

register	
r_num /* 挂号编号 */	varchar(10)
r_patient_id /* 病人身份证号 */	varchar(20)
r_sex /* 性别 */	varchar(2)
r_dept /* 挂号科室 */	varchar(20)
r_name /* 医生姓名 */	varchar(10)

pgm	
t_id /* 收费编号 */	varchar(10)
drug_id /* 药品编号 */	varchar(10)
quantity /* 数量 */	varchar(10)
price /* 价格 */	decimal(10) unsigned

pay	
patient_id /* 病人编号 */	varchar(10)
t_id /* 收费编号 */	varchar(10)
price /* 价格 */	decimal(10)

recipel	
doctor_id /* 医生编号 */	varchar(10)
drug_id /* 药品编号 */	varchar(10)
count /* 数量 */	varchar(4)

6. 数据操作要求及实现

6.1.1 数据查询、更新操作

```
select * from register where register.is_delete=0;
```

```

33 ✓ commit;
34 select *
35 from register where register.is_delete=0;
36 insert into register
37 values ('101', '411282xxxxxxxx5555', '男', '牙科', '王渊洁');
38 insert into register

```

Output hospital_information.register

4 rows

Tc:Auto

DDL

CSV

↑

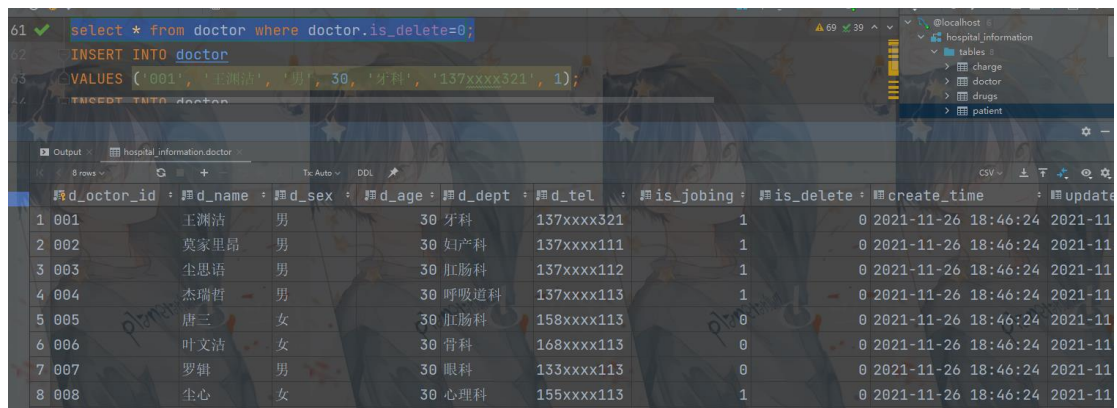
↓

↺

↻

	r_num	r_patient_id	r_sex	r_dept	r_name	is_delete	create_time	update_time
1	101	411282xxxxxxxx5555	男	牙科	王渊洁	0	2021-11-26 18:44:18	2021-11-26 18:44:18
2	102	421282xxxxxxxx5554	女	妇产科	莫家里昂	0	2021-11-26 18:44:18	2021-11-26 18:44:18
3	103	251381xxxxxxxx5553	男	肛肠科	尘思语	0	2021-11-26 18:44:18	2021-11-26 18:44:18
4	104	315213xxxxxxxx5552	女	呼吸道科	杰瑞哲	0	2021-11-26 18:44:18	2021-11-26 18:44:18


```
select * from doctor where doctor.is_delete=0;
```



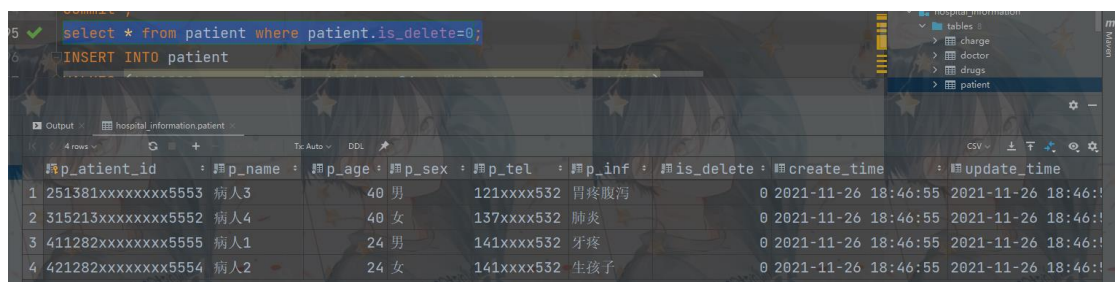
The screenshot shows a database management tool interface. The SQL editor contains the following queries:

```
61 select * from doctor where doctor.is_delete=0;
62 INSERT INTO doctor
63 VALUES ('001', '王渊洁', '男', 30, '牙科', '137xxxx321', 1);
64 INSERT INTO doctor
```

The output window displays the 'hospital_information.doctor' table with 8 rows. The columns are: d_octor_id, d_name, d_sex, d_age, d_dept, d_tel, is_jobing, is_delete, create_time, and update_time.

d_octor_id	d_name	d_sex	d_age	d_dept	d_tel	is_jobing	is_delete	create_time	update_time
1 001	王渊洁	男	30	牙科	137xxxx321	1	0	2021-11-26 18:46:24	2021-11-26 18:46:24
2 002	莫家里昂	男	30	妇产科	137xxxx111	1	0	2021-11-26 18:46:24	2021-11-26 18:46:24
3 003	李思语	男	30	肛肠科	137xxxx112	1	0	2021-11-26 18:46:24	2021-11-26 18:46:24
4 004	杰瑞哲	男	30	呼吸道科	137xxxx113	1	0	2021-11-26 18:46:24	2021-11-26 18:46:24
5 005	唐三	女	30	肛肠科	158xxxx113	0	0	2021-11-26 18:46:24	2021-11-26 18:46:24
6 006	叶文洁	女	30	骨科	168xxxx113	0	0	2021-11-26 18:46:24	2021-11-26 18:46:24
7 007	罗辑	男	30	眼科	133xxxx113	0	0	2021-11-26 18:46:24	2021-11-26 18:46:24
8 008	生心	女	30	心理科	155xxxx113	1	0	2021-11-26 18:46:24	2021-11-26 18:46:24

```
select * from patient where patient.is_delete=0;
```



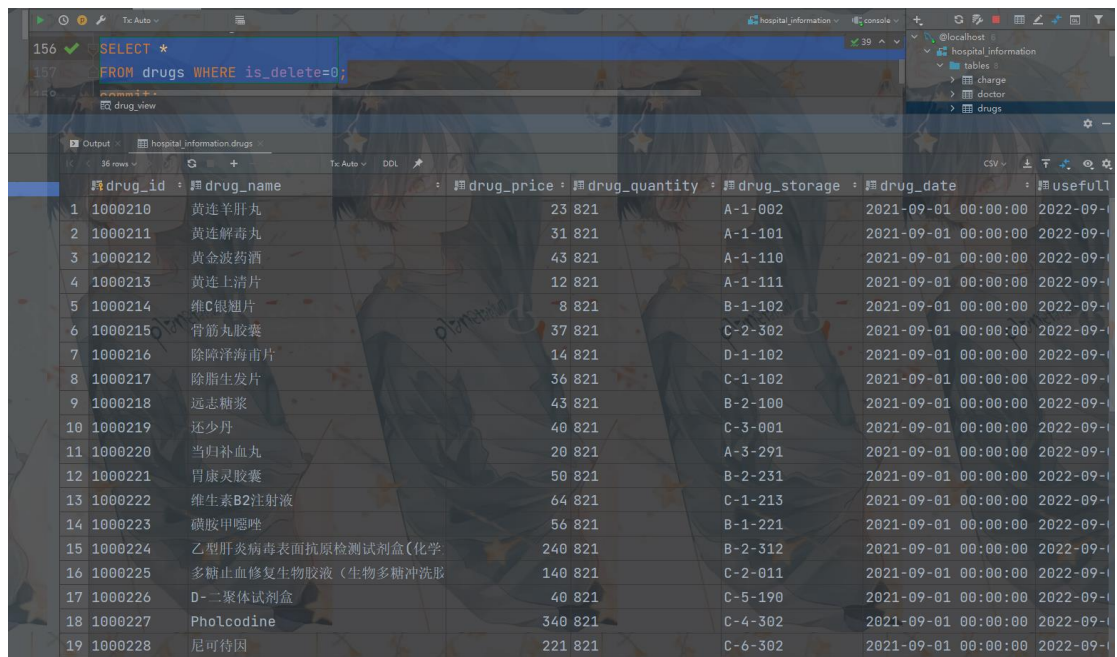
The screenshot shows a database management tool interface. The SQL editor contains the following queries:

```
5 select * from patient where patient.is_delete=0;
6 INSERT INTO patient
```

The output window displays the 'hospital_information.patient' table with 4 rows. The columns are: p_atient_id, p_name, p_age, p_sex, p_tel, p_inf, is_delete, create_time, and update_time.

p_atient_id	p_name	p_age	p_sex	p_tel	p_inf	is_delete	create_time	update_time
1 251381xxxxxxxx5553	病人3	40	男	121xxxx532	胃疼腹泻	0	2021-11-26 18:46:55	2021-11-26 18:46:55
2 315213xxxxxxxx5552	病人4	40	女	137xxxx532	肺炎	0	2021-11-26 18:46:55	2021-11-26 18:46:55
3 411282xxxxxxxx5555	病人1	24	男	141xxxx532	牙疼	0	2021-11-26 18:46:55	2021-11-26 18:46:55
4 421282xxxxxxxx5554	病人2	24	女	141xxxx532	生孩子	0	2021-11-26 18:46:55	2021-11-26 18:46:55

```
SELECT * FROM drugs WHERE is_delete=0;
```



The screenshot shows a database management tool interface. The SQL editor contains the following queries:

```
156 SELECT *
157 FROM drugs WHERE is_delete=0;
```

The output window displays the 'hospital_information.drugs' table with 19 rows. The columns are: drug_id, drug_name, drug_price, drug_quantity, drug_storage, drug_date, and usefull.

drug_id	drug_name	drug_price	drug_quantity	drug_storage	drug_date	usefull
1 1000210	黄连羊肝丸	23 821		A-1-002	2021-09-01 00:00:00	2022-09-01
2 1000211	黄连解毒丸	31 821		A-1-101	2021-09-01 00:00:00	2022-09-01
3 1000212	黄金波药酒	43 821		A-1-110	2021-09-01 00:00:00	2022-09-01
4 1000213	黄连上清片	12 821		A-1-111	2021-09-01 00:00:00	2022-09-01
5 1000214	维C银翘片	8 821		B-1-102	2021-09-01 00:00:00	2022-09-01
6 1000215	骨筋丸胶囊	37 821		C-2-302	2021-09-01 00:00:00	2022-09-01
7 1000216	除障泽海甫片	14 821		D-1-102	2021-09-01 00:00:00	2022-09-01
8 1000217	除脂生发片	36 821		C-1-102	2021-09-01 00:00:00	2022-09-01
9 1000218	远志糖浆	43 821		B-2-100	2021-09-01 00:00:00	2022-09-01
10 1000219	还少丹	40 821		C-3-001	2021-09-01 00:00:00	2022-09-01
11 1000220	当归补血丸	20 821		A-3-291	2021-09-01 00:00:00	2022-09-01
12 1000221	胃康灵胶囊	50 821		B-2-231	2021-09-01 00:00:00	2022-09-01
13 1000222	维生素B2注射液	64 821		C-1-213	2021-09-01 00:00:00	2022-09-01
14 1000223	磺胺甲噁唑	56 821		B-1-221	2021-09-01 00:00:00	2022-09-01
15 1000224	乙型肝炎病毒表面抗原检测试剂盒(化学发光法)	240 821		B-2-312	2021-09-01 00:00:00	2022-09-01
16 1000225	多糖止血修复生物胶液(生物多糖冲洗液)	140 821		C-2-011	2021-09-01 00:00:00	2022-09-01
17 1000226	D-二聚体试剂盒	40 821		C-5-190	2021-09-01 00:00:00	2022-09-01
18 1000227	Pholcodine	340 821		C-4-302	2021-09-01 00:00:00	2022-09-01
19 1000228	尼可待因	221 821		C-6-302	2021-09-01 00:00:00	2022-09-01

```
select * from charge WHERE is_delete=0;
```

```

270 drop table charge;
271 ✓ select *
272   from charge WHERE is_delete=0;
273 INSERT INTO charge(toll_id, t_name, patient_id, drug_id, drug_quantity, amount)
274 VALUES ('101', '收费员1', '411282xxxxxxxx5555', '100023', '2盒', 80.00);

```

toll_id	t_name	patient_id	drug_id	drug_quantity	amount	is_delete	create_time	update_time
1 101	收费员1	411282xxxxxxxx5555	100023	2盒	80	0	2021-11-26 18:47:48	2021-11-26 18:47:48
2 101	收费员1	421282xxxxxxxx5554	1000233	1盒	20	0	2021-11-26 18:47:48	2021-11-26 18:47:48
3 102	收费员1	251381xxxxxxxx5553	1000229	1盒	440	0	2021-11-26 18:47:48	2021-11-26 18:47:48
4 102	收费员1	315213xxxxxxxx5552	1000230	2盒	1080	0	2021-11-26 18:47:48	2021-11-26 18:47:48

select * from PGM WHERE is_delete=0;

```

297 ✓ create view PGM_view as
298   select * from PGM WHERE is_delete=0;
299 commit;
300 select *
301   from PGM;
302 insert into PGM(t_id, drug_id, quantity, price)

```

t_id	drug_id	quantity	price	is_delete	create_time	update_time
1 001	100023	2盒	80	0	2021-11-26 18:48:49	2021-11-26 18:48:49
2 002	1000233	1盒	20	0	2021-11-26 18:48:49	2021-11-26 18:48:49
3 003	1000229	1盒	440	0	2021-11-26 18:48:49	2021-11-26 18:48:49
4 004	1000230	2盒	1080	0	2021-11-26 18:48:49	2021-11-26 18:48:49

select * from recipel WHERE is_delete=0;

```

324 ✓ select * from recipel WHERE is_delete=0;
325 commit;
326 select *
327   from recipel;

```

id	doctor_id	drug_id	count	patient_name	is_delete	create_time	update_time
1	1 001	100023	2盒	病人1	0	2021-11-26 18:49:03	2021-11-26 18:49:03
2	2 002	1000233	1盒	病人2	0	2021-11-26 18:49:03	2021-11-26 18:49:03
3	3 003	1000229	1盒	病人3	0	2021-11-26 18:49:03	2021-11-26 18:49:03
4	4 004	1000230	2盒	病人4	0	2021-11-26 18:49:03	2021-11-26 18:49:03

select * from pay WHERE is_delete=0;

```

356 commit;
357 ✓ select *
358   from pay WHERE is_delete=0;
359 drop table pay;

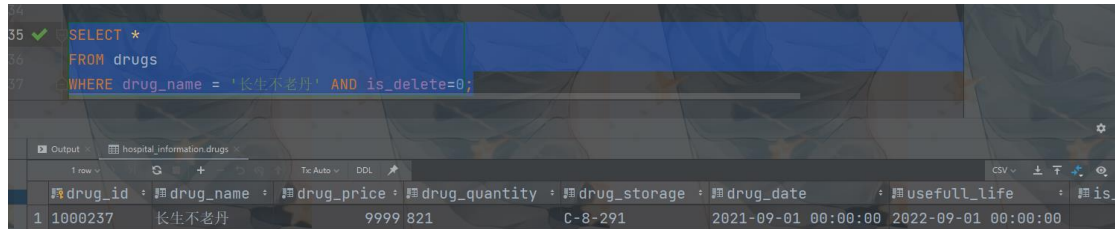
```

patient_id	t_id	price	is_delete	create_time	update_time
1 251381xxxxxxxx5553	003	440	0	2021-11-26 18:49:23	2021-11-26 18:49:23
2 315213xxxxxxxx5552	004	1080	0	2021-11-26 18:49:23	2021-11-26 18:49:23
3 411282xxxxxxxx5555	001	80	0	2021-11-26 18:49:23	2021-11-26 18:49:23
4 421282xxxxxxxx5554	002	20	0	2021-11-26 18:49:23	2021-11-26 18:49:23

6.1.2 实现药品的入库、出库管理；

1. **INSERT INTO** drugs(drug_id, drug_name, drug_price, drug_quantity, drug_storage, drug_date, usefull_life)

2. `VALUES ('1000237', '长生不老丹', 9999.00, '821', 'C-8-291', '2021-09-01', '2022-09-01');`
- 3.
4. `SELECT * FROM drugs WHERE drug_name = '长生不老丹' AND is_delete=0;`



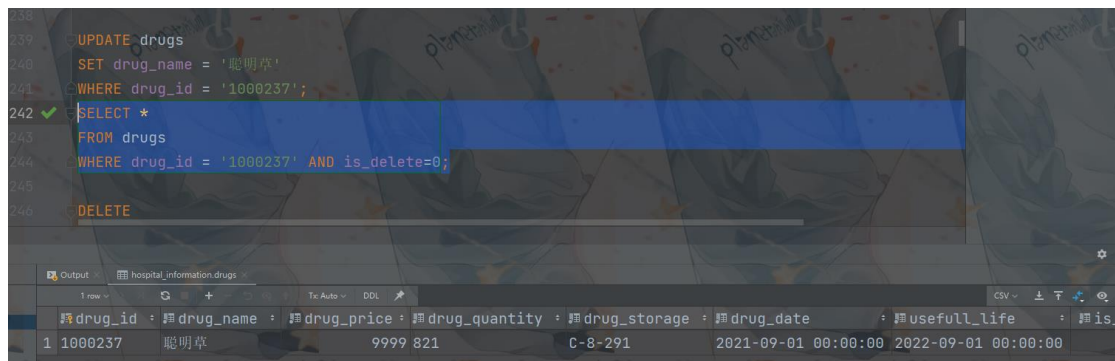
The screenshot shows a SQL IDE with a query editor and an output window. The query editor contains the following SQL code:

```
SELECT *
FROM drugs
WHERE drug_name = '长生不老丹' AND is_delete=0;
```

The output window displays the results of the query in a table with the following columns: drug_id, drug_name, drug_price, drug_quantity, drug_storage, drug_date, usefull_life, and is_delete. The table contains one row of data.

drug_id	drug_name	drug_price	drug_quantity	drug_storage	drug_date	usefull_life	is_delete
1000237	长生不老丹	9999.00	821	C-8-291	2021-09-01 00:00:00	2022-09-01 00:00:00	0

1. `UPDATE drugs SET drug_name = '聪明草' WHERE drug_id = '1000237';`
- 2.
3. `UPDATE drugs SET IS_DELETE=1 WHERE drug_name='聪明草';`



The screenshot shows a SQL IDE with a query editor and an output window. The query editor contains the following SQL code:

```
UPDATE drugs
SET drug_name = '聪明草'
WHERE drug_id = '1000237';
```

The output window displays the results of the query in a table with the following columns: drug_id, drug_name, drug_price, drug_quantity, drug_storage, drug_date, usefull_life, and is_delete. The table contains one row of data.

drug_id	drug_name	drug_price	drug_quantity	drug_storage	drug_date	usefull_life	is_delete
1000237	聪明草	9999.00	821	C-8-291	2021-09-01 00:00:00	2022-09-01 00:00:00	0



The screenshot shows a SQL IDE with a query editor and an output window. The query editor contains the following SQL code:

```
INSERT INTO drugs(drug_id, drug_name, drug_price, drug_quantity, drug_storage, drug_date, usefull_life)
VALUES ('1000237', '长生不老丹', 9999.00, '821', 'C-8-291', '2021-09-01', '2022-09-01');

UPDATE drugs
SET drug_name = '聪明草'
WHERE drug_id = '1000237';

UPDATE drugs SET IS_DELETE=1 WHERE drug_name='聪明草';

SELECT drug_name, sum(drug_quantity)
FROM drug_view
GROUP BY drug_name;

commit;
```

The output window displays the results of the queries in a table with the following columns: drug_id, drug_name, drug_price, drug_quantity, drug_storage, drug_date, usefull_life, and is_delete. The table contains one row of data.

drug_id	drug_name	drug_price	drug_quantity	drug_storage	drug_date	usefull_life	is_delete
1000237	聪明草	9999.00	821	C-8-291	2021-09-01 00:00:00	2022-09-01 00:00:00	1

对应的 java 后端实现

1. 增删改操作
2. `package com.vector.hospital_information;`
- 3.
4. `import com.vector.config.SpringConfiguration;`
5. `import org.junit.Test;`
- 6.
7. `import org.junit.runner.RunWith;`
8. `import org.springframework.context.ApplicationContext;`

```
9. import org.springframework.context.annotation.AnnotationConfigApplicationCont
    ext;
10. import org.springframework.stereotype.Component;
11. import org.springframework.test.context.ContextConfiguration;
12. import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
13.
14. import javax.annotation.Resource;
15. import javax.sql.DataSource;
16. import java.sql.Connection;
17. import java.sql.PreparedStatement;
18. import java.sql.SQLException;
19.
20. @RunWith(SpringJUnit4ClassRunner.class)
21. @ContextConfiguration(classes = {SpringConfiguration.class})
22. @Component("Update_test")
23. public class Update_test {
24.     @Resource(name = "dataSource")
25.     DataSource dataSource;
26.     @Resource(name = "Update_test")
27.     Update_test modify;
28.     @Test
29.     /**
30.      * 测试增删改
31.      */
32.     public void TestUpdate() throws SQLException {
33.
34.         /**
35.          * 测试增添数据
36.          */
37.         String sql1 = "INSERT INTO drugs(drug_id,drug_name,drug_price,drug_qu
            antity,drug_storage,drug_date,usefull_life) VALUES (?, ?, ?, ?, ?, ?, ?)";
38.         modify.update(sql1,"1000237", " 长生不老丹 ", 9999.00, "821", "A-8-
            291", "2021-09-01", "2022-09-01");
39.
40.         /**
41.          * 测试修改数据
42.          */
43.         // String sql2 = "UPDATE drugs SET drug_name = ? WHERE drug_id = ?";
44.         // modify.update(sql2,"聪明草","1000237");
45.         // /**
46.         //      * 测试删除数据
47.         //      */
48.         // String sql3 = "DELETE FROM drugs WHERE drug_name=?";
49.         // modify.update(sql3,"聪明草");
```



```
50.     }
51.     /**
52.      * 王佳慧
53.      * 通用增删改
54.      * @param sql
55.      * @param args
56.      * @throws SQLException
57.      */
58.
59.     //通用的增删改操作
60.     public void update(String sql, Object... args) throws SQLException { //sql
        当中占位符个数与可变形参的长度一致
61.
62.         Connection conn = null;
63.         PreparedStatement ps = null;
64.         //ApplicationContext app = null;
65.         try {
66.             //app = new AnnotationConfigApplicationContext(SpringConfiguratio
                n.class)
67.             //1.获取数据库连接
68.             conn = dataSource.getConnection();
69.             //2.预编译 sql 语句, 返回 PreparedStatement 实例
70.             ps = conn.prepareStatement(sql);
71.             //3.填充占位符
72.             for (int i = 0; i < args.length; i++) {
73.                 ps.setObject(i + 1, args[i]);
74.             }
75.             //4.执行 sql 语句
76.             ps.execute();
77.             System.out.println("添加记录成功");
78.         } catch (Exception e) {
79.             e.printStackTrace();
80.         } finally {
81.             //5.资源的关闭
82.             conn.close();
83.         }
84.     }
85.
86. }
```

```
30      */
31      public void TestUpdate() throws SQLException {
32
33          /**
34           * 测试增添数据
35           */
36          String sql1 = "INSERT INTO drugs(drug_id,drug_name,drug_price,drug_quantity,drug_s
37          modify.update(sql1, new Object[] { "1000237", "长生不老丹", 9999.00, "821", "A-8-291", "2021-0
38
39          /**
40           * 测试修改数据
41           */
42      }
43  }
```

Tests passed: 1 of 1 test - 342 ms

11月 21, 2021 8:10:06 下午 com.alibaba.druid.support.logging.JakartaCommonsLoggingImpl info
信息: {dataSource-1} initied
添加记录成功

1. Sql 查询

```
2. package com.vector.hospital_information;
3.
4. import com.vector.config.SpringConfiguration;
5. import com.vector.test.DataSourceTest;
6. import org.junit.Test;
7. import org.junit.runner.RunWith;
8. import org.springframework.stereotype.Component;
9. import org.springframework.test.context.ContextConfiguration;
10. import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
11.
12. import javax.annotation.Resource;
13. import javax.sql.DataSource;
14. import java.lang.reflect.Field;
15. import java.sql.*;
16.
17. @RunWith(SpringJUnit4ClassRunner.class)
18. @ContextConfiguration(classes = {SpringConfiguration.class})
19. @Component("PreparedStatementQueryTest")
20. public class PreparedStatementQueryTest {
21.
22.     @Resource(name = "dataSource")
23.     DataSource dataSource;
24.     @Resource(name = "PreparedStatementQueryTest")
25.     PreparedStatementQueryTest queryTest ;
26.
27.
28.     @Test
29.     /**
30.      * 测试查询
```

```
31.     */
32.     public void TestQuery() throws SQLException {
33.
34.
35.         /**
36.          * 测试查询一条记录
37.          */
38.         String sql = "SELECT * from drugs where drug_name=?";
39.         Drugs drugs = queryTest.getInstance(Drugs.class,sql,"长生不老丹");
40.         System.out.println(drugs);
41.
42.
43.     }
44.     /**
45.      * 王佳慧
46.      * 针对于不同的表的通用查询操作,返回表中的一条记录
47.      * @param clazz
48.      * @param sql
49.      * @param args
50.      * @param <T>
51.      * @return
52.      */
53.     public <T>T getInstance(Class<T> clazz,String sql,Object ...args) throws
        SQLException {
54.         Connection conn = null;
55.         PreparedStatement ps = null;
56.         ResultSet rs = null;
57.         try {
58.             conn = dataSource.getConnection();//加载数据库
59.             ps = conn.prepareStatement(sql);
60.             for (int i = 0; i < args.length; i++) {
61.                 ps.setObject(i + 1, args[i]);
62.             }
63.             //执行,获取结果集
64.             rs = ps.executeQuery();
65.             //获取结果集的元数据
66.             ResultSetMetaData rsmd = rs.getMetaData();
67.             //获取列数
68.             int columuCount = rsmd.getColumnCount();
69.             if (rs.next()) {
70.                 T t = clazz.newInstance();
71.                 for (int i = 0; i < columuCount; i++) {
72.                     //获取每个列的列值,通过 ResultSet
73.                     Object columnValue = rs.getObject(i + 1);
```

```

74.          //获取每个列的列名,通过 ResultSetMetaData
75.          //获取列的列名:getColumnName() ---不推荐使用
76.          //获取列的别名:getColumnLabel()
77.          String columnLabel = rsmd.getColumnLabel(i+1);
78.          // 通过反射,将对象指定名 columnName 的属性值赋值给
          columnValue
79.          Field field = clazz.getDeclaredField(columnLabel);
80.          field.setAccessible(true);
81.          field.set(t, columnValue);
82.      }
83.      return t;
84.  }
85.  } catch (Exception e) {
86.      e.printStackTrace();
87.  } finally {
88.      conn.close();
89.  }
90.
91.      return null;
92.  }
93.
94.
95.}

```



```

/**
 * 测试查询一条记录
 */
String sql = "SELECT * from drugs where drug_name=?";
Drugs drugs = queryTest.getInstance(Drugs.class,sql, new Object[] {"长生不老丹"});
System.out.println(drugs);

```

11月 21, 2021 8:25:31 下午 com.alibaba.druid.support.logging.JakartaCommonsLoggingImpl info
 信息: {dataSource-1} inited
 Drugs{drug_id='1000237', drug_name='长生不老丹', drug_price=9999, drug_quantity='821', drug_storage='A-8-291',
 drug_date=2021-09-01T00:00, usefull_life=2022-09-01T00:00}

6.1.3 实现科室、医生、病人的管理；

(1) 逻辑增删改

```

1. INSERT INTO register(r_num, r_patient_id, r_sex, r_dept, r_name)
2. VALUES ('222', '411282xxxxxx1182', '女', '肛肠科', '尘思宇');
3.
4. SELECT * from register where r_patient_id='41128220230304554X WHERE IS_DELETE=0'
   ;
5.

```

```

6. START TRANSACTION;
7. BEGIN;
8. UPDATE patient SET p_name = '病人
   1' WHERE p_atient_id = '41128220230304554X AND IS_DELETE=0;
9. UPDATE register SET r_name = '病人
   1' WHERE r_patient_id = '41128220230304554X' AND;
10. IS_DELETE=0;
11.
12. UPDATE register SET IS_DELETE=1 WHERE r_num='222';

```

Java 相关事务提交核心代码

```

1. try {
2.         //app = new AnnotationConfigApplicationContext(SpringC
   onfiguration.class)
3.         //1. 获取数据库连接
4.         conn = dataSource.getConnection();
5.         conn.setAutoCommit(false);
6.         //2. 预编译 sql 语句, 返回 PreparedStatement 实例
7.         ps = conn.prepareStatement(sql);
8.         //3. 填充占位符
9.         for (int i = 0; i < args.length; i++) {
10.             ps.setObject(i + 1, args[i]);
11.         }
12.         //4. 执行 sql 语句
13.         ps.execute();
14.         conn.commit();
15.         System.out.println("添加记录成功");
16.     } catch (Exception e) {
17.         conn.rollback();
18.         e.printStackTrace();
19.     } finally {
20.         //5. 资源的关闭
21.         conn.close();
22.     }
23. }

```

(2) 级联操作

```

1. -- 级联操作
2. alter table patient add
3.     constraint patient_register_dept

```


4. `foreign key(p_patient_id) references register(r_patient_id) on delete cascade;`
5. `DELETE FROM patient WHERE p_patient_id='41128220230304554X';`

6.1.4 实现处方的登记管理；

```

7. create view recipel_view as select * from recipel;
8. commit ;
9. select *
10. from recipel;
11. INSERT INTO recipel
12. VALUES (1,'001', '100023', '2盒', '病人1');
13. INSERT INTO recipel
14. VALUES (2,'002', '1000233', '1盒', '病人2');
15. INSERT INTO recipel
16. VALUES (3,'003', '1000229', '1盒', '病人3');
17. INSERT INTO recipel
18. VALUES (4,'004', '1000230', '2盒', '病人4');

```



id	doctor_id	drug_id	count	patient_name
1	001	100023	2盒	病人1
2	002	1000233	1盒	病人2
3	003	1000229	1盒	病人3
4	004	1000230	2盒	病人4

1. /**
2. * 测试增添数据
3. */
4. String sql1 = "INSERT INTO recipel(id,doctor_id,drug_id,count,patient_name) VALUES (?,?,,?,?);";
5. modify.update(sql1,1,"001", "100023", "2 盒","病人 1");

6.1.5 实现收费管理；

```
271 commit;
272
273 # 支付凭据
274 create table pay -- 中间表 patient_charge
275 (
276     patient_id varchar(20) comment '病人编号',
277     t_id        varchar(10) comment '收费编号',
278     price       decimal not null comment '价格',
279     primary key (patient_id, t_id)
280 );
281 create view pay_view as select * from pay;
282 commit;
```

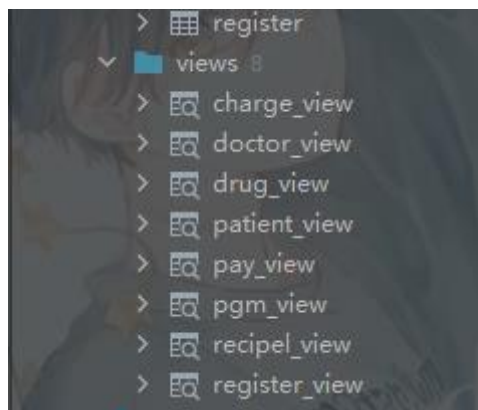
Output × hospital_information.pay ×

4 rows

	patient_id	t_id	price
1	251381xxxxxxxx5553	003	440
2	315213xxxxxxxx5552	004	1080
3	411282xxxxxxxx5555	001	80
4	421282xxxxxxxx5554	002	20

6.2 视图

创建视图查询各种药品的库存总数：




```

194 DELETE FROM drugs WHERE drug_name='聪明草';
195 ✓ SELECT drug_name,sum(drug_quantity) FROM drug_view GROUP BY drug_name ;
196 commit;
197

```

	drug_name	sum(drug_quantity)
1	黄连羊肝丸	821
2	黄连解毒丸	821
3	黄金波药酒	821
4	黄连上清片	821
5	维C银翘片	821
6	骨筋丸胶囊	821
7	除障泽海甫片	821
8	除脂生发片	821
9	远志糖浆	821
10	还少丹	821
11	当归补血丸	821
12	胃康灵胶囊	821

1. `SELECT drug_name,sum(drug_quantity) FROM drug_view GROUP BY drug_name ;`

6.3 触发器

药品出库操作

1. -- 创建触发器，当药品入库、出库时自动修改库存；
2. # 药品出库操作
3. delimiter \$\$ -- 自定义结束符号
4. create trigger recipel_update
5. before insert
6. on recipel
7. for EACH ROW
8. BEGIN
9. SELECT @quantity=drug_quantity into @str
10. FROM drugs WHERE NEW.drug_id = drugs.drug_id;
11. IF @quantity <= 0 || NEW.count > @quantity THEN
12. SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning: 药品数量为零!';
13. ELSE
14. UPDATE drugs SET drug_quantity = drug_quantity - NEW.count WHERE drug_id =NEW.drug_id AND IS_DELETE=0;
15. end if;
16. end
17. \$\$ -- 自定义触发器结束
18. delimiter ;


```

375 create trigger recipel_update
376 after insert
377 on recipel
378 for EACH ROW
379 BEGIN
380 SELECT @quantity=drug_quantity into @str
381 FROM drugs WHERE NEW.drug_id = drugs.drug_id;
382 IF @quantity <= 0 || NEW.count > @quantity THEN
383 SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning: 药品数量为零!';
384 ELSE
385 UPDATE drugs SET drug_quantity = drug_quantity - NEW.count WHERE drug_id=NEW.drug_id AND is
386 end if;
387 end
388 $$ -- 自定义触发器结束
389 delimiter ;
390 drop trigger recipel_update;
391
392 INSERT INTO recipel(DOCTOR_ID, COUNT, DRUG_ID, PATIENT_NAME)
393 VALUES ('005', '2', '1000226', '病人6');
394

```

插入前

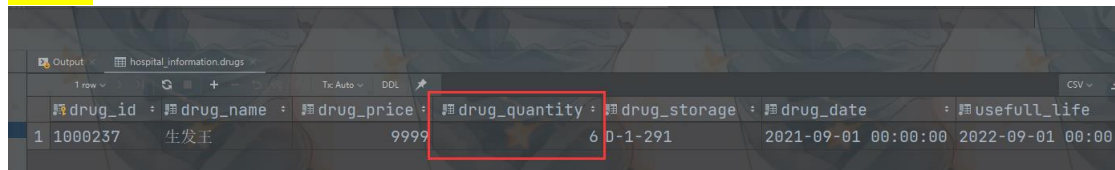
16	1000225	多糖止血修复生物胶液（生物多糖冲洗胶	140	817 C-2-011	2021
17	1000226	D-二聚体试剂盒	40	817 C-5-190	2021
18	1000227	Pholcodine	340	817 C-4-302	2021

插入后

WHERE	ORDER BY	drug_id	drug_name	drug_price	drug_quantity	drug_storage
		3	1000212	黄金波药酒	43	817 A-1-110
		4	1000213	黄连上清片	12	817 A-1-111
		5	1000214	维C银翘片	8	817 B-1-102
		6	1000215	筋骨丸胶囊	37	817 C-2-302
		7	1000216	除障泽海甫片	14	817 D-1-102
		8	1000217	除脂生发片	36	817 C-1-102
		9	1000218	远志糖浆	43	817 B-2-100
		10	1000219	还少丹	40	817 C-3-001
		11	1000220	当归补血丸	20	817 A-3-291
		12	1000221	胃康灵胶囊	50	817 B-2-231
		13	1000222	维生素B2注射液	64	817 C-1-213
		14	1000223	磺胺甲噁唑	56	817 B-1-221
		15	1000224	乙型肝炎病毒表面抗原检测试剂盒(化学	240	817 B-2-312
		16	1000225	多糖止血修复生物胶液（生物多糖冲洗胶	140	817 C-2-011
		17	1000226	D-二聚体试剂盒	40	815 C-5-190
		18	1000227	Pholcodine	340	817 C-4-302
		19	1000228	尼可待因	221	817 C-6-302
		20	1000229	Ethylmorphine	440	817 C-4-202
		21	100023	感冒灵颗粒	40	817 A-2-302
		22	1000230	Thiofentanyl	560	817 C-3-071

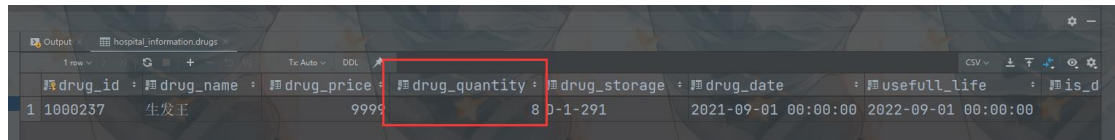
#药品入库操作

插入前



	drug_id	drug_name	drug_price	drug_quantity	drug_storage	drug_date	usefull_life
1 row	1	1000237	生发王	9999	6	0-1-291	2021-09-01 00:00:00 2022-09-01 00:00:00

插入后



	drug_id	drug_name	drug_price	drug_quantity	drug_storage	drug_date	usefull_life	is_d
1 row	1	1000237	生发王	9999	8	0-1-291	2021-09-01 00:00:00 2022-09-01 00:00:00	

6.4 存储过程

创建存储过程统计某段时间内，各科室的就诊人数和输入情况；

存储过程

```
drop procedure count_people_date;
```

```
DELIMITER $$
```

```
CREATE PROCEDURE count_people_date(
```

```
#      IN @begin_date datetime, 这样写是错误的
```

```
      IN begin_date datetime,
```

```
      IN end_date datetime
```

```
)
```

```
BEGIN
```

```
    SELECT r_dept '科室',count(*) '问诊人数'
```

```
    FROM register
```

```
    WHERE update_time BETWEEN begin_date AND end_date AND is_delete=0
```

```
    GROUP BY r_dept ;
```

```
end $$
```

```
DELIMITER ;
```

```
CALL count_people_date('2021-12-04','2021-12-05');
```

4 rows

WHERE ORDER BY r_dept

ent_id	r_p_name	r_sex	r_dept	r_name	is_delete	create_time	update_time
1 (xxxxxx5552	病人4	女	呼吸道科	杰瑞哲	0	2021-12-04 16:22:46	2021-12-04 16:22:46
2 (xxxxxx5554	病人2	女	妇产科	莫家里昂	0	2021-12-04 16:22:46	2021-12-04 16:22:46
3 (xxxxxx5553	病人3	男	肛肠科	尘思语	0	2021-12-04 16:22:46	2021-12-04 16:22:46
4 (xxxxxx1182	病人1	女	肛肠科	尘思宇	0	2021-12-04 16:21:05	2021-12-04 16:21:05

```
# 存储过程
drop procedure count_people_date;
DELIMITER $$
CREATE PROCEDURE count_people_date(
#
# IN @begin_date datetime, 这样写是错误的
IN begin_date datetime,
IN end_date datetime
)
BEGIN
SELECT r_dept '科室', count(*) '问诊人数'
FROM register
WHERE update_time BETWEEN begin_date AND end_date AND is_delete=0
GROUP BY r_dept ;
end $$
DELIMITER ;

CALL count_people_date( begin_date: '2021-12-04', end_date: '2021-12-05');
```

Output Result 27

科室	问诊人数
1 妇产科	1
2 肛肠科	2
3 呼吸道科	1

7.基准测试

压力测试：

所有数据写入一个文件
文件名 C:\Users\78235\Desktop\hospital_information\压力测试.xml

显示日志内容: ☐ 仅错误日志 ☐ 仅成功日志

Label	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/sec
JDBC Request	3669588	58	19	28	33	988	0	32084	58.58%	7050.4/sec	3278.28	0.00
总体	3669588	58	19	28	33	988	0	32084	58.58%	7050.4/sec	3278.28	0.00

时间延迟测试：

```
(no such file or directory)
[root@iz8vbetpyzn77hrq77e601Z shell_dir]# sh sql_analyze.sh 5-sec-status-2021-12-05_08-status
#ts date time load QPS
1638708235 2021-12-05 20:43:55 0.02
1638708240 2021-12-05 20:44:00 0.02
1638708245 2021-12-05 20:44:05 0.02
1638708390 2021-12-05 20:46:30 0.08
1638708395 2021-12-05 20:46:35 0.07[root@iz8vbetpyzn77hrq77e601Z shell_dir]#
```

```
TS 1638708235.011112177 2021-12-05 20:43:55 20:43:55 up 33 days, 2:29, 1 user, load average: 0.02, 0.25, 0.35
TS 1638708240.003433445 2021-12-05 20:44:00 20:44:00 up 33 days, 2:29, 1 user, load average: 0.02, 0.25, 0.35
TS 1638708245.003713191 2021-12-05 20:44:05 20:44:05 up 33 days, 2:29, 1 user, load average: 0.02, 0.24, 0.35
TS 1638708390.003236595 2021-12-05 20:46:30 20:46:30 up 33 days, 2:31, 1 user, load average: 0.08, 0.23, 0.33
TS 1638708395.004275820 2021-12-05 20:46:35 20:46:35 up 33 days, 2:32, 1 user, load average: 0.07, 0.22, 0.32
```

8 . 总结

对于本次课程设计,考虑到做事就要做完美,做了一次了,那就给他开源,设计,优化,调试,测试,对比都做一遍.当然在进行数据库操作的时候,难度最大的有

1. **触发器** 由于我使用了 mysql,这与 sql service 有着高度的隔离.这部分内容是完全不一样的.做迁移时耗费了很大力气.有很多问题摸索了很久,有些问题是 stackoverflow, 百度, csdn 都解决不了的.比如我使用了 update 触发器,但是在触发器中执行插入操作,总是插不进去!困扰了我很久,寻访各个 dba 群中大佬,都无能为力.总之,有些是耗费很长时间都无法解决的,很挫败.但虽然挫败,但我也收获了很多,对于基本的 sql 操作,以及 mysql, sql service 都有了深刻认识.

2. **新思想** 同时在设计数据库的过程中,也触发了很多新思想,有了新思想,眼前豁然开朗,逻辑删除思想解决了一对多,牵一发而动全身的难题.也保证了数据的永久存储.

3. **Linux 下的数据时延测试.** 本以为靠两个 shell 自动化脚本很容易做.但是错误也是频繁发生.但在解决问题的过程中,我了解到了 sh -x 脚本调试.这让我捕获了错误.完成了数据采集,以及 qbs 在时间维度的测试.

9.参考文献

[1]高性能 MySQL:第 3 版/(美)Schwartz, B. (美)Zaitsev, P., (美)Tkachenko, V.

著;宁海元等译.-北京:电子工业出版社,2013.5 书名原文:High Performance MySQL, Third Edition.

[2]数据库系统概论/王珊,萨师煊编著.—5 版.—北京:高等教育出版社,2014.9
ISBN 978-7-04-040664-1

[3]java 开发手册社区开发者集体智慧的结晶-(华山版)v1.5.0.-杭州:阿里巴巴,2019.06