电子科技大学计算机学院

标 准 实 验 报 告

（实验）课程名称 数据库及其应用

电子科技大学教务处制表

电 子 科 技 大 学

实 验 报 告

学生姓名：刘文晨　　学号：2018080901006　　指导教师：孙明

实验地点：主楼 A2-412　　实验时间：2020-12-5 下午

一、实验室名称：主楼 A2-412

二、实验项目名称：酒店预订系统实验

三、实验学时：4

四、实验原理：

本实验利用 SQL 语句对存储了酒店信息的数据库进行操作。结构化查询语言(Structured Query Language)简称 SQL，是一种特殊目的的编程语言，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统。

本次实验主要使用到的是查询语句，包括选择列表语句（SELECT）、FROM 子句和 WHERE 子句。它们分别说明所查询列、查询的表或视图、以及搜索条件等。对查询出的结果排序后输出。

选择列表语句（SELECT）指出所查询列，它可以是一组列名列表、星号、表达式、变量（包括局部变量和全局变量）等构成。SELECT 语句可实现的操作包括：

1. 选择所有列
2. 选择部分列并指定它们的显示次序
3. 更改列标题
4. 删除重复行
5. 限制返回的行数等等

FROM 子句指定 SELECT 语句查询及与查询相关的表或视图。在 FROM 子

句中最多可指定 256 个表或视图，它们之间用逗号分隔。

WHERE 子句设置查询条件，过滤掉不需要的数据行。WHEREE 子句可包括各种条件运算符。

使用 ORDER BY 子句对查询返回的结果按一列或多列排序。ORDER BY 子句的语法格式为：ORDER BY {column_name [ASC|DESC]} [，…n] 其中 ASC 表示升序，为默认值，DESC 为降序。

查询操作涉及简单查询、连接查询、嵌套查询以及集合查询等，需要根据具体的问题与要求，选用其中的查询方式查找目标数据。

五、实验目的：

通过本次实验，了解系统的设计方法，掌握分析问题和运用数据库相关知识解决问题的能力，熟练掌握 SQL 语句的编写与测试。

六、实验内容：

本次实验基于 MySQL 实现。MySQL 是一个关系型数据库管理系统，能够很好的对数据进行存储与读取。所使用的酒店信息数据存储在 hotel_data.sql 中，酒店预订系统的数据库表结构如图 1 所示。

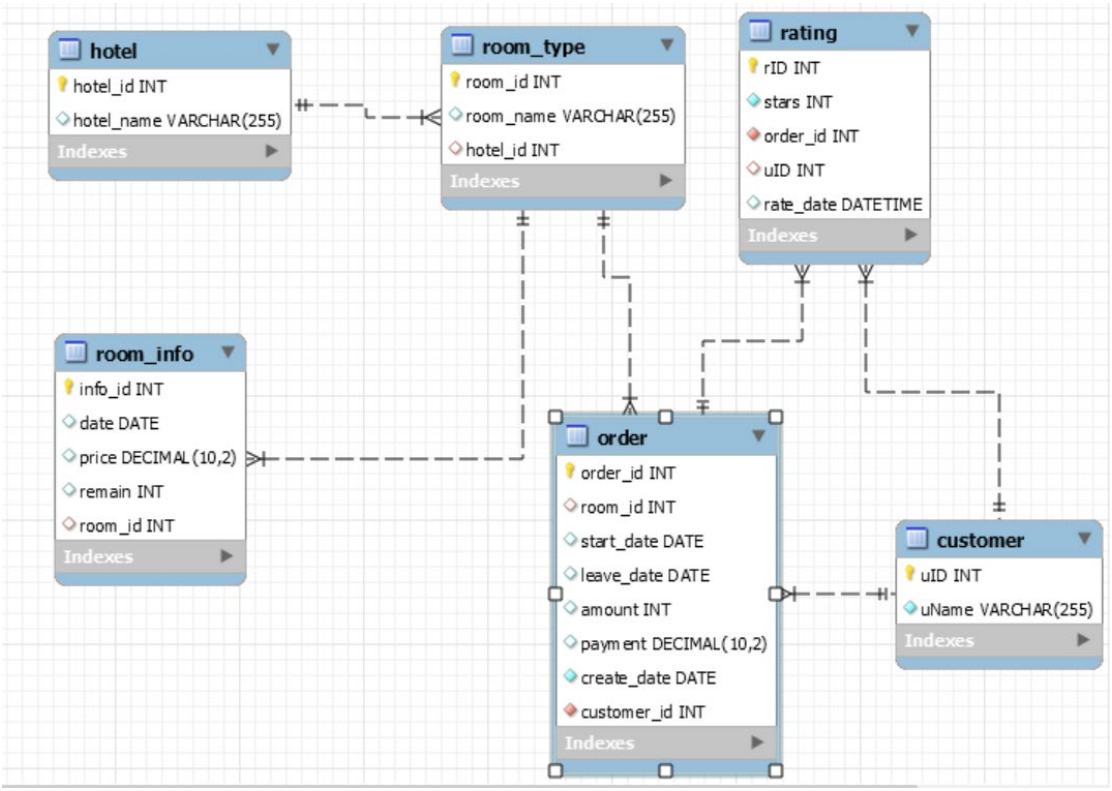


图 1 酒店预订系统的数据库表结构图

- customer(uID,uName)
- hotel(hotel_id,hotel_name)
- order(order_id,room_id,start_date,leave_date,amount,payment,create_date,custom_id)
- rating(rID,stars,order_id,uID,rate_date)
- room_info(info_id,date,price,remain,room_id)
- room_type(room_id,room_name,hotel_id)

实验利用 SQL 语句对存储了酒店信息的数据库进行操作，完成以下操作：

1. 查询所有房型的具体信息，包括 room_id, room_name, hotel_id。
2. 查询所有酒店名称中包含“希尔顿”的酒店，返回酒店名称和酒店 id。
3. 查询订单总价在 10000 元及以上的所有订单详情，包括订单编号、酒店编号、房型编号及居住时长。
4. 查询所有房型的订单情况，包括房型编号，房型名称，订单编号、价格。
5. 创建启悦酒店的订单视图。
6. 在订单表的总价字段上 创建降序的普通索引。索引名为 orderpayment。
7. 查询所有酒店 2020-11-14 所有房型的平均价格并从低到高排序。
8. 从订单表中统计一个酒店在指定日期的各种房型的预订情况。例如统计希尔顿大酒店 2020-11-14 当天各个房型预定情况，返回酒店名，房型，预定数量。
9. 查找同时评价了 2 次及以上的用户信息。
10. 查询评价过所有总统套房的顾客姓名。
11. 指定时间区间和每天要预定的房间数量，查询满足条件（时间区间，将预定房间数）的房型及其平均价格，并按平均价格从低到高进行排序。查询结果应包含酒店，房型及平均价格信息。例如预定 11.14-16 日每天房间数量 4 间。
12. 完成预订房间，包括更新房型信息和创建订单。例如订单为预订 11 月 14 号-15 号 4 号房型 4 间。

七、实验器材（设备、元器件）：

处理器：Intel® Core™ i5-8300H CPU @ 2.30GHz　2.30GHz
已安装的内存(RAM)：8GB

系统类型：64 位操作系统，基于 x64 的处理器

DBMS：MySQL Server 8.0

数据库管理工具：Navicat Premium 15

八、实验步骤：

以下为每个问题的实现思路。

1. 查询所有房型的具体信息，包括 room_id, room_name, hotel_id。

   分析：需要查询的 room_id, room_name, hotel_id 都在 room_type 中，直接查询即可。

2. 查询所有酒店名称中包含“希尔顿”的酒店，返回酒店名称和酒店 id。

   分析：需要查询的 hotel_name, hotel_id 都在 hotel 中，查询条件为 hotel_name LIKE '%希尔顿%'。

3. 查询订单总价在 10000 元及以上的所有订单详情，包括订单编号、酒店编号、房型编号及居住时长。

   分析：需要查询的 order_id,hotel_id,room_id 在 order、hotel、room_type 中，居住时长为 leave_date-start_date+1，查询条件为 payment>=10000。

4. 查询所有房型的订单情况，包括房型编号，房型名称，订单编号、价格。

   分析：需要查询的 room_id, room_name, order_id, payment 在 order、hotel、room_type 中，使用右连接。

5. 创建启悦酒店的订单视图。

   分析：语法为 CREATE VIEW qiyue_hotel AS …，后面为查询语句。

6. 在订单表的总价字段上 创建降序的普通索引。索引名为 orderpayment。

   分析：语法为 CREATE INDEX orderpayment ON table_name(…)。

7. 查询所有酒店 2020-11-14 所有房型的平均价格并从低到高排序。

   分析：需要查询的 hotel_id,hotel_name,AVG(price)在 room_info、hotel、room_type 中，以 hotel_id 分组，以 AVG(price)排序。

8. 从订单表中统计一个酒店在指定日期的各种房型的预订情况。例如统计希尔顿大酒店 2020-11-14 当天各个房型预定情况，返回酒店名，房型，预定数量。

   分析：需要查询的 hotel_name,room_name,amount 在 order、hotel、room_type 中，查询条件中，start_date<='2020-11-14' AND start_date>='2020-11-14'。

9. 查找同时评价了 2 次及以上的用户信息。

分析：用 NOT IN 排除没有评价或只评价 1 次的用户信息。

10. 查询评价过所有总统套房的顾客姓名。

    分析：注意使用 DISTINCT。

11. 指定时间区间和每天要预定的房间数量，查询满足条件（时间区间，将预定房间数）的房型及其平均价格，并按平均价格从低到高进行排序。查询结果应包含酒店，房型及平均价格信息。例如预定 11.14-16 日每天房间数量 4 间。

    分析：需要查询的 hotel_name,room_name,AVG(price)在 room_info、hotel、room_type 中，以 room_id 分组，以 AVG(price)排序。

12. 完成预订房间，包括更新房型信息和创建订单。例如订单为预订 11 月 14 号-15 号 4 号房型 4 间。

    分析：order_id 应为 null，custom_id 要在 customer 的 uID 中存在，payment 需要再使用 1 个子查询求出。

九、实验数据及结果分析：

1. 查询所有房型的具体信息，包括 room_id, room_name, hotel_id

```
SELECT * FROM room_type;
```

room_id	room_name	hotel_id
1	商务双床房	1
2	行政大床房	1
3	豪华套房	1
4	海景房	2
5	园景房	2
6	山景房	2
7	总统套房	3
8	豪华套房	3
9	普通套房	3
10	行政双床房	1
11	亲子房	1
12	总统套房	4
13	豪华海景房	4
14	标准双床房	4
15	情侣大床房	1
16	总统套房	5
17	行政套房	5
18	商务双床房	5
19	豪华海景房	6
20	别墅海景房	6

图 2 问题 1 运行截图

2. 查询所有酒店名称中包含“希尔顿”的酒店，返回酒店名称和酒店 id

```
SELECT hotel_name,hotel_id FROM hotel WHERE hotel_name LIKE '%希尔顿%';
```

hotel_name	hotel_id
希尔顿大酒店	5
希尔顿度假酒店	6

图 3 问题 2 运行截图

3. 查询订单总价在 10000 元及以上的所有订单详情，包括订单编号、酒店编号、房型编号及居住时长

```
SELECT order_id,hotel_id,room_id,leave_date-start_date+1 AS time
FROM hotel NATURAL JOIN `order` NATURAL JOIN room_type WHERE payment>=1
0000;
```

order_id	hotel_id	room_id	time
24	5	16	4
13	5	16	3
16	5	17	3
5	1	2	3
8	3	7	3
10	4	12	3
22	3	7	4
23	4	12	10
18	3	9	2

图 4 问题 3 运行截图

4. 查询所有房型的订单情况，包括房型编号，房型名称，订单编号、价格。

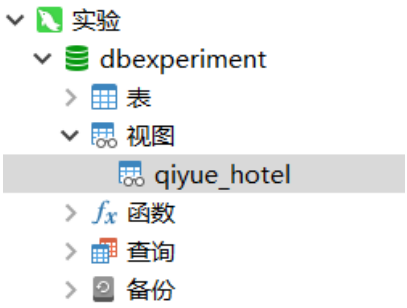
```
SELECT room_type.room_id,room_type.room_name,`order`.order_id,`order
`.payment
FROM `order` RIGHT JOIN room_type ON `order`.room_id=room_type.room_
id;
```

room_id	room_name	order_id	payment
1	商务双床房	2	7295.00
2	行政大床房	5	16000.00
3	豪华套房	7	6000.00
4	海景房	4	8400.00
5	园景房	1	5300.00
5	园景房	20	5600.00
6	山景房	6	5200.00
7	总统套房	8	15400.00
7	总统套房	22	12000.00
7	总统套房	25	7500.00
8	豪华套房	3	7300.00
9	普通套房	18	11750.00
9	普通套房	19	3450.00
10	行政双床房	9	6654.00
11	亲子房	11	9200.00
12	总统套房	10	12900.00
12	总统套房	23	12000.00
12	总统套房	26	6000.00
13	豪华海景房	17	4950.00
13	豪华海景房	21	6300.00
14	标准双床房	12	4650.00
15	情侣大床房	15	8800.00
16	总统套房	13	17800.00
17	行政套房	16	17000.00
18	商务双床房	14	8250.00
19	豪华海景房	(Null)	(Null)
20	别墅海景房	(Null)	(Null)

图 5 问题 4 运行截图

5. 创建启悦酒店的订单视图。

```
CREATE VIEW qiyue_hotel
AS SELECT `order`.* FROM `order` NATURAL JOIN hotel NATURAL JOIN room_t
ype WHERE hotel_name LIKE '%启悦%';
```



order_id	room_id	start_date	leave_date	amount	payment	create_date	customer_id
4	4	2020-11-14	2020-11-16	2	8400.00	2020-11-01	201905
1	5	2020-11-14	2020-11-15	2	5300.00	2020-11-01	201904
20	5	2020-11-16	2020-11-16	4	5600.00	2020-11-01	201904
6	6	2020-11-14	2020-11-16	2	5200.00	2020-11-01	201907

图 6 问题 5 运行截图

6. 在订单表的总价字段上 创建降序的普通索引。索引名为 orderpayment。

```
CREATE UNIQUE INDEX orderpayment ON `order`(payment DESC);
```

```
1  show INDEX FROM `order`
```

信息 结果 1 剖析 状态

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
order	0	PRIMARY	1	order_id	A	27	(Null)	(Null)		BTREE			YES	(Null)
order	1	room_order_id	1	room_id	A	18	(Null)	(Null)	YES	BTREE			YES	(Null)
order	1	customer_id	1	customer_id	A	18	(Null)	(Null)		BTREE			YES	(Null)
order	1	orderpayment	1	payment	D	25	(Null)	(Null)	YES	BTREE			YES	(Null)

图 7 问题 6 运行截图

7. 查询所有酒店 2020-11-14 所有房型的平均价格并从低到高排序

```
SELECT hotel_id,hotel_name,AVG(price) AS average_price FROM hotel NATUR
AL JOIN room_type NATURAL JOIN room_info WHERE date='2020-11-14' GROUP
BY hotel_id ORDER BY average_price;
```

hotel_id	hotel_name	average_price
3	来住大酒店	1250.000000
1	丽呈东谷酒店	1375.000000
2	启悦酒店	1383.333333
5	希尔顿大酒店	1450.000000
4	悦杏温泉酒店	1550.000000

图 8 问题 7 运行截图

8. 从订单表中统计一个酒店在指定日期的各种房型的预订情况。例如统计希尔顿大酒店 2020-11-14 当天各个房型预定情况，返回酒店名，房型，预定数

```
SELECT hotel_name,room_name,amount
FROM hotel NATURAL JOIN room_type NATURAL JOIN `order`
WHERE hotel_name LIKE '希尔顿大酒店' AND start_date='2020-11-14';
```

hotel_name	room_name	amount
希尔顿大酒店	总统套房	4
希尔顿大酒店	行政套房	4
希尔顿大酒店	商务双床房	3

图 9 问题 8 运行截图

9. 查找同时评价了 2 次及以上的用户信息。

```
SELECT uID,uName FROM customer
WHERE uID NOT IN (SELECT uID FROM rating GROUP BY uID HAVING COUNT(uID)
=1);
```

uID	uName
201903	梅野石
201904	李琦
201907	李佳奇
2019018	罗翔

图 10 问题 9 运行截图

10. 查询评价过所有总统套房的顾客姓名。

```
SELECT DISTINCT uName
FROM customer,rating,`order`,room_type
WHERE customer.uID=rating.uID AND rating.order_id=order.order_id AND ro
om_type.room_id=order.room_id
AND room_name="总统套房" GROUP BY rating.uID
HAVING COUNT(DISTINCT room_type.room_id)>=(SELECT COUNT(*) FROM room_ty
pe WHERE room_name="总统套房");
```

uName
李佳奇

图 11 问题 10 运行截图

11. 指定时间区间和每天要预定的房间数量，查询满足条件（时间区间，将预定房间数）的房型及其平均价格，并按平均价格从低到高进行排序。查询结果应包含酒店，房型及平均价格信息。例如预定 11.14-16 日每天房间数量 4 间。

```
SELECT hotel_name,room_name,AVG(price) AS average_price FROM hotel NATU
RAL JOIN room_type NATURAL JOIN room_info WHERE date<='2020-11-16' AND
date>='2020-11-14' GROUP BY room_id HAVING min(remain)>=4 ORDER BY aver
age_price;
```

hotel_name	room_name	average_price
启悦酒店	海景房	1400.000000
悦杏温泉酒店	总统套房	1433.333333

图 12 问题 11 运行截图

12. 完成预订房间，包括更新房型信息和创建订单。例如订单为预订 11 月 14 号-15 号 4 号房型 4 间。

```
INSERT INTO `order` VALUES (null,4,'2020-11-14','2020-11-15',4,4*(SELECT SUM(price) FROM room_info WHERE room_id=4 AND date BETWEEN '2020-11-14' AND '2020-11-15'),'2020-12-07',201901);
UPDATE room_info SET remain=remain-4 WHERE room_id=4 AND date BETWEEN '2020-11-14' AND '2020-11-15';
```

order_id	room_id	start_date	leave_date	amount	payment	create_date	customer_id
1	5	2020-11-14	2020-11-15	2	5300.00	2020-11-01	201904
2	1	2020-11-28	2020-11-28	5	7295.00	2020-11-01	201901
3	8	2020-11-14	2020-11-16	2	7300.00	2020-11-01	201902
4	4	2020-11-14	2020-11-16	2	8400.00	2020-11-01	201905
5	2	2020-11-14	2020-11-16	4	16000.00	2020-11-01	201906
6	6	2020-11-14	2020-11-16	2	5200.00	2020-11-01	201907
7	3	2020-11-14	2020-11-14	5	6000.00	2020-11-01	201908
8	7	2020-11-14	2020-11-16	4	15400.00	2020-11-01	201909
9	10	2020-11-28	2020-11-29	2	6654.00	2020-11-01	2019010
10	12	2020-11-14	2020-11-16	3	12900.00	2020-11-01	2019011
11	11	2020-11-14	2020-11-16	2	9200.00	2020-11-01	2019012
12	14	2020-11-14	2020-11-16	1	4650.00	2020-11-01	2019013
13	16	2020-11-14	2020-11-16	4	17800.00	2020-11-01	2019014
14	18	2020-11-14	2020-11-15	3	8250.00	2020-11-01	2019015
15	15	2020-11-14	2020-11-16	2	8800.00	2020-11-01	2019016
16	17	2020-11-14	2020-11-16	4	17000.00	2020-11-01	2019017
17	13	2020-11-14	2020-11-14	3	4950.00	2020-11-01	2019018
18	9	2020-11-14	2020-11-15	5	11750.00	2020-11-01	201903
19	9	2020-11-16	2020-11-16	3	3450.00	2020-11-01	201903
20	5	2020-11-16	2020-11-16	4	5600.00	2020-11-01	201904
21	13	2020-11-15	2020-11-16	2	6300.00	2020-11-01	2019018
22	7	2020-09-01	2020-09-04	4	12000.00	2020-09-01	201907
23	12	2020-07-01	2020-07-10	3	12000.00	2020-06-01	201907
24	16	2020-10-01	2020-10-04	3	180000.00	2020-08-01	201907
25	7	2020-11-01	2020-11-01	3	7500.00	2020-09-01	201908
26	12	2020-11-01	2020-11-01	2	6000.00	2020-08-01	201908
27	16	2020-11-01	2020-11-03	1	8000.00	2020-09-01	201908
28	4	2020-11-14	2020-11-15	4	11000.00	2020-12-07	201901

info_id	date	price	remain	room_id
1	2020-11-28	1459.00	6	1
2	2020-11-29	1468.00	4	1
3	2020-11-30	1488.00	6	1
4	2020-11-14	1300.00	2	2
5	2020-11-15	1300.00	1	2
6	2020-11-16	1400.00	1	2
7	2020-11-14	1200.00	4	3
8	2020-11-15	1200.00	3	3
9	2020-11-16	1300.00	4	3
10	2020-11-14	1450.00	1	4
11	2020-11-15	1300.00	1	4
12	2020-11-16	1450.00	5	4

图 13 问题 12 运行截图

十、实验结论：

通过本次实验，基本了解一个简单的系统的结构与设计方法，初步掌握了分析问题和运用数据库相关知识解决问题的能力。

十一、总结及心得体会：

通过本次实验，锻炼了 SQL 语句的编写和测试的能力。因为把“=”错写成“LIKE”，错了一题，所以继续完成了 4 个附加题，加深了对于存储过程和触发器的理解，加强了 JDBC 和 JFrame 等 Java 代码编写的能力。

十二、对本实验过程及方法、手段的改进建议：

本实验基本在 Navicat 上完成，可以换用华为云数据库实现。附加题的 3、4 题用 Java 实现，可以换成 Web 或者微信小程序作为前端。

报告评分：

指导教师签字：