



西安交通大学

软件学院

数据库实验：报刊订阅管理系统数据库设计

选课编号：

姓 名：

学 号：

班 级：

指导老师：

完成日期：

2025. 6. 8

目 录

1. 需求分析	1
2. 概念结构设计	3
3. 逻辑结构设计	3
3.1 建立关系模式	4
3.2 关系模式规范化处理	4
3.3 关系模式逻辑结构定义	5
4. 数据库物理设计	7
5. 数据库实施与测试	11
5.1 数据入库	11
5.2 建立视图	12
5.3 增删改查测试	15
5.3.1 插入测试	15
5.3.2 修改测试	15
5.3.3 删除测试	16
5.3.4 查询测试	17
5.4 建立并测试触发器	18
6. 数据库建模实验总结	20
7. 样本数据库的安装和数据加载	20
8. 样本数据库的分析	20
9. 样本数据库上的查询	21
9.1 查询一	21
9.2 查询二	22
9.3 查询三	23
9.4 查询四	24
9.5 查询五	25
9.6 查询六	26
9.7 查询七	27
9.8 查询八	28
9.9 查询九	29
9.10 查询十	30
9.11 查询十一	31
9.12 查询十二	32

9.13 查询十三	33
9.14 插入十四	34
10. 运用 SQL EXPLAIN 进行查询分析	36
10.1 对于以上查询的分析	36
10.1.1 查询一分析	36
10.1.2 查询二分析	37
10.1.3 查询三分析	37
10.1.4 查询四分析	37
10.1.5 查询五分析	38
10.1.6 查询六分析	38
10.1.7 查询七分析	38
10.1.8 查询八分析	39
10.1.9 查询九分析	39
10.1.10 查询十分析	40
10.1.11 查询十一分析	40
10.1.12 查询十二分析	40
10.1.13 查询十三分析	41
10.2 索引与查询优化	41
11. SQL 查询实验总结	42

在当今信息快速传播的时代，报刊仍作为重要的文化传播与信息载体，服务于广大读者与各类企事业单位。随着订阅数量和用户类型的不断增长，传统的人工管理方式已难以满足现代报刊发行机构的运营需求。为了提高订阅服务效率，规范订单流程，降低人力成本，构建一个功能完善、操作便捷的报刊订阅管理系统已成为行业发展的必然选择。

1. 需求分析

报刊订阅管理系统用于集中管理报刊目录、订户、订单、投递信息等数据，实现订阅登记、订单管理、投递安排和历史追溯等功能。系统需确保数据一致性、业务合规性以及投递过程的可追踪性。具体需求如下：

1. 报刊目录

系统需维护当前可订阅的报刊目录，包含以下信息：

报刊编号

报刊名称

出版周期（如日报、周刊、月刊）

单价

2. 订户信息

系统需存储报刊订户的详细信息，包括：

订户编号

姓名

联系电话

通讯地址

3. 订单信息

用于记录订户的订阅行为，每条订单对应一次订阅操作，需包括：

订单编号

订户编号

报刊编号

下单日期

订阅起止日期

数量（份数）

4. 投递卡信息

投递卡是组织实际投递工作的核心单位，每张投递卡可关联多个订单的投递信息，包括：

投递卡编号

发卡日期

5. 投递记录

投递记录用于保存每次实际投递行为，支持一次订单的多次投递，记录完整的投递历史，包括：

投递记录编号

投递卡编号

订单编号

投递日期

投递份数

2. 概念结构设计

根据以上需求建立 E-R 图。订户与订单是 1:n 的关系；报刊与订单是 1:n 的关系；订单和投递记录是 1:n 的关系；投递卡和投递记录是 1:n 的关系。

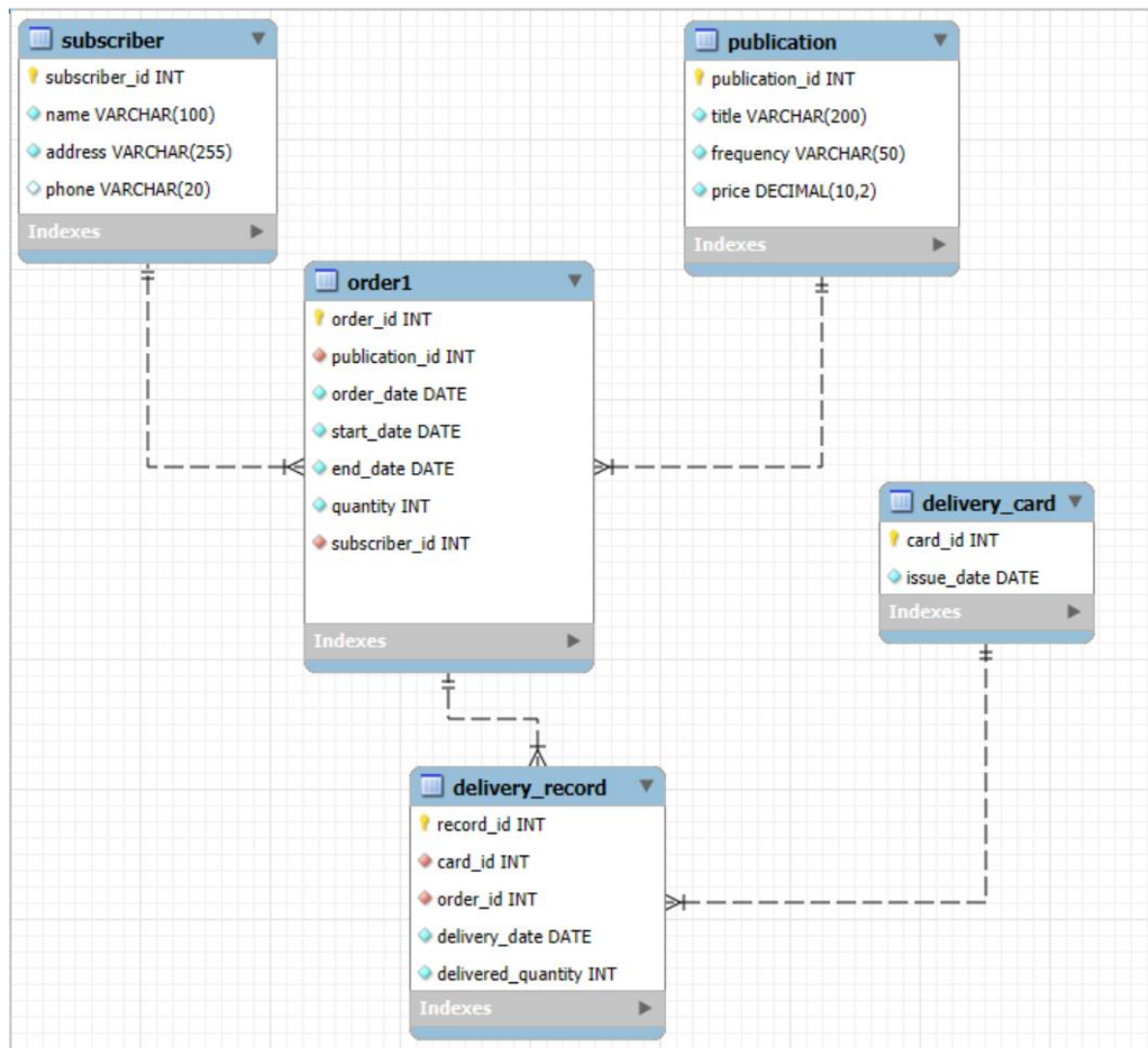


图 2-1 报刊订阅管理系统 ER 图

3. 逻辑结构设计

逻辑设计的任务是在概念设计的基础上给出与 DBMS 相关的数据库逻辑模式。逻辑结构设计主要包括关系模式的建立、关系模式的规范化处理和关系模式的逻辑结构定义。

3.1 建立关系模式

对于 E1 与 E2 的 1:n 关系,当 E2 是部分参与时,转换为关系模式 R1(K1, A1), R2(K2, A2), R3(K1, K2, AR), 其中 K2 是 R3 的外键。当 E2 是全参与时,转换为关系模式 R1(K1, A1), R2(K2, A2, K1, AR), 其中 K1 是 R2 的外键。

对于 E1 与 E2 的 m:n 关系,转换为关系模式 R1(K1, A1), R2(K2, A2), R3(K1, K2, AR), 其中 K1, K2 联合构成 R3 的键, 且 K1, K2 又同时都是外键。

对于报刊订阅管理系统, 根据其 E-R 图可建立如下几个关系模式:

报刊目录 (报刊编号, 报刊名称, 发行频率, 报价)

订户 (订户编号, 姓名, 地址, 电话)

订单 (订单编号, 订户编号, 报刊编号, 订购日期, 开始日期, 结束日期, 订阅份数)

投递卡 (投递卡编号, 发放日期)

投递记录 (投递记录编号, 投递卡编号, 订单编号, 投递日期, 投递份数)

其中, 订单关系模式中的外键是订户编号、报刊编号, 投递记录关系模式中的外键是投递卡编号、订单编号。

3.2 关系模式规范化处理

报刊目录关系模式中主属性为报刊编号, 非主属性为报刊名称、发行频率、报价。报刊名称、发行频率、报价均完全函数依赖于报刊编号 (主键), 无部分依赖或传递依赖。不存在非主属性对主键的传递依赖 (如报价不依赖于发行频率等中间属性)。所以该关系模式是 3NF。

订户关系模式中主属性为订户编号, 非主属性为姓名、地址、电话。姓名、地址、电话均完全函数依赖于订户编号, 无部分依赖或传递依赖。非主属性之间无依赖关系 (如地址不依赖于姓名)。所以该关系模式是 3NF。

订单关系模式中主属性为订单编号, 非主属性为订户编号、报刊编号、订购日期、开始日期、结束日期、订阅份数。订户编号、报刊编号、订购日期、开始日期、结束日期、订阅份数均完全函数依赖于订单编号。订户编号和报刊编号作为外键, 仅用于关联其他表, 自身无额外依赖; 非主属性间无传递依赖 (如结束日期不依赖于订阅份数)。所以该关系模式是 3NF。

投递卡关系模式中主属性为投递卡编号，非主属性为发放日期。发放日期完全函数依赖于投递卡编号（主键），无部分依赖或传递依赖。所以该关系模式是 3NF。

投递记录关系模式中主属性为投递记录编号（主键），非主属性为投递卡编号、订单编号、投递日期、投递份数。投递卡编号、订单编号、投递日期、投递份数均完全函数依赖于投递记录编号（主键）。投递卡编号和订单编号作为外键，仅用于关联其他表，自身无额外依赖；非主属性间无传递依赖（如投递份数不依赖于投递日期）。所以该关系模式是 3NF。

综上所述，这几个关系模式都是 3NF，无需进行关系模式的分解。

3.3 关系模式逻辑结构定义

对于以上建立的关系模式，可以列出如下表格表示其逻辑结构：

表 3-1 订户关系模式

属性名	含义	数据类型	长度	是否为主属性	是否为外键	约束条件
subscriber_id	订户编号	INT	10	是		Not null Unsigned
name	姓名	VARCHAR	100			Not null
address	地址	VARCHAR	255			Not null
phone	电话	VARCHAR	20			null

表 3-2 报刊目录关系模式

属性名	含义	数据类型	长度	是否为主属性	是否为外键	约束条件
publication_id	报刊编号	INT	10	是		Not null Unsigned

title	报刊名称	VARCHAR	200		Not null
frequency	发行频率	VARCHAR	50		Not null
price	单价（保留两位小数）	DECIMAL	10,2		Not null

表 3-3 订单表关系模式

属性名	含义	数据类型	长度	是否为主属性	是否为外键	约束条件
order_id	订单编号	INT	10	是		Not null Unsigned
subscriber_id	订户编号	INT	10		是	Not null
publication_id	报刊编号	INT	10		是	Not null
order_date	下单日期	DATE				Not null
start_date	订阅起始日期	DATE				Not null
end_date	订阅截止日期	DATE				Not null
quantity	订阅份数	INT	10			Not null

表 3-4 投递卡关系模式

属性名	含义	数据类型	长度	是否为主属性	是否为外键	约束条件
card_id	投递卡编号	INT	10	是		Not null Unsigned
issue_date	发卡日期	DATE		是		Not null

表 3-5 投递记录关系模式

属性名	含义	数据类型	长度	是否为主属性	是否为外键	约束条件
record_id	记录编号	INT	10	是		Not null

					Unsigned
card_id	投递卡编号	INT	10	是	Not null
order_id	订单编号	INT	10	是	Not null
delivery_date	投递日期	DATE			Not null
delivered_quantity	次投递份数	INT	10		Not null

4. 数据库物理设计

根据逻辑结构设计的结果，在 MySQL Workbench 中建立数据库。代码如下：

```
SET @@OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0;
```

```
SET @@OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
```

```
SET @@OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_I
N_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_S
UBSTITUTION';
```

```
-- -----
-- Schema mydb
-- -----
```

```
-- -----
-- Schema mydb
-- -----
```

```
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET
utf8 ;
USE `mydb` ;
```

```
-- -----  
-- Table `mydb`.`subscriber`  
-- -----  
  
CREATE TABLE IF NOT EXISTS `mydb`.`subscriber` (  
  `subscriber_id` INT UNSIGNED NOT NULL,  
  `name` VARCHAR(100) NOT NULL,  
  `address` VARCHAR(255) NOT NULL,  
  `phone` VARCHAR(20) NULL,  
  PRIMARY KEY (`subscriber_id`),  
  UNIQUE INDEX `subscriber_id_UNIQUE` (`subscriber_id` ASC) VISIBLE)  
ENGINE = InnoDB;  
  
  
-- -----  
-- Table `mydb`.`publication`  
-- -----  
  
CREATE TABLE IF NOT EXISTS `mydb`.`publication` (  
  `publication_id` INT UNSIGNED NOT NULL,  
  `title` VARCHAR(200) NOT NULL,  
  `frequency` VARCHAR(50) NOT NULL,  
  `price` DECIMAL(10,2) NOT NULL,  
  PRIMARY KEY (`publication_id`),  
  UNIQUE INDEX `publication_id_UNIQUE` (`publication_id` ASC) VISIBLE)  
ENGINE = InnoDB;  
  
  
-- -----  
-- Table `mydb`.`order1`  
-- -----  
  
CREATE TABLE IF NOT EXISTS `mydb`.`order1` (  
  `order_id` INT UNSIGNED NOT NULL,
```

```
`publication_id` INT NOT NULL,  
`order_date` DATE NOT NULL,  
`start_date` DATE NOT NULL,  
`end_date` DATE NOT NULL,  
`quantity` INT UNSIGNED NOT NULL,  
`subscriber_id` INT NOT NULL,  
PRIMARY KEY (`order_id`),  
UNIQUE INDEX `order_id_UNIQUE` (`order_id` ASC) VISIBLE,  
INDEX `fk_order1_subscriber_idx` (`subscriber_id` ASC) VISIBLE,  
INDEX `fk_order1_publication1_idx` (`publication_id` ASC) VISIBLE,  
CONSTRAINT `fk_order1_subscriber`  
    FOREIGN KEY (`subscriber_id`)  
    REFERENCES `mydb`.`subscriber` (`subscriber_id`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
CONSTRAINT `fk_order1_publication1`  
    FOREIGN KEY (`publication_id`)  
    REFERENCES `mydb`.`publication` (`publication_id`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `mydb`.`delivery_card`  
-- -----  
  
CREATE TABLE IF NOT EXISTS `mydb`.`delivery_card` (  
    `card_id` INT UNSIGNED NOT NULL,  
    `issue_date` DATE NOT NULL,  
    PRIMARY KEY (`card_id`),  
    UNIQUE INDEX `card_id_UNIQUE` (`card_id` ASC) VISIBLE)
```

ENGINE = InnoDB;

```
-- -----
-- Table `mydb`.`delivery_record`
-- -----
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`delivery_record` (
  `record_id` INT UNSIGNED NOT NULL,
  `card_id` INT NOT NULL,
  `order_id` INT NOT NULL,
  `delivery_date` DATE NOT NULL,
  `delivered_quantity` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`record_id`),
  UNIQUE INDEX `record_id_UNIQUE` (`record_id` ASC) VISIBLE,
  INDEX `fk_delivery_record_delivery_card1_idx` (`card_id` ASC) VISIBLE,
  INDEX `fk_delivery_record_order11_idx` (`order_id` ASC) VISIBLE,
  CONSTRAINT `fk_delivery_record_delivery_card1`
    FOREIGN KEY (`card_id`)
      REFERENCES `mydb`.`delivery_card` (`card_id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_delivery_record_order11`
    FOREIGN KEY (`order_id`)
      REFERENCES `mydb`.`order1` (`order_id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

```
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

运行结果如下：



图 4-1 数据库建立结果

5. 数据库实施与测试

对建立的数据库进行数据入库、视图建立、增删查改和触发器等相关测试。

5.1 数据入库

对建立的数据库插入一些测试数据，插入代码如下：

```
INSERT INTO Subscriber (subscriber_id, name, address, phone) VALUES
```

```
(1, '张三', '北京市海淀区', '13800000001'),
```

```
(2, '李四', '上海市浦东新区', '13800000002'),
```

```
(3, '王五', '广州市天河区', '13800000003'),
```

```
(4, '赵六', '深圳市南山区', '13800000004'),
```

```
(5, '孙七', '成都市武侯区', '13800000005');
```

```
INSERT INTO Publication (publication_id, title, frequency, price) VALUES
```

```
(101, '人民日报', '每日', 2.00),
```

```
(102, '参考消息', '每日', 1.50),
```

```
(103, '南方周末', '每周', 4.00),
```

```
(104, '中国青年报', '每日', 1.80),
```

```
(105, '经济观察报', '每周', 3.50);
```

```
INSERT INTO order1 (order_id, subscriber_id, publication_id, order_date,
start_date, end_date, quantity) VALUES
```

```
(1001, 1, 101, '2025-01-01', '2025-01-05', '2025-06-30', 1),
```

```
(1002, 1, 102, '2025-01-02', '2025-01-10', '2025-06-30', 2),
```

```
(1003, 2, 103, '2025-01-05', '2025-01-15', '2025-12-31', 1),
```

```

(1004, 3, 104, '2025-01-10', '2025-01-20', '2025-07-31', 1),
(1005, 4, 105, '2025-01-15', '2025-01-25', '2025-12-31', 2);
INSERT INTO delivery_card (card_id, issue_date) VALUES
(201, '2025-01-10'),
(202, '2025-01-20'),
(203, '2025-02-01'),
(204, '2025-02-15'),
(205, '2025-03-01');
INSERT INTO delivery_record (record_id, card_id, order_id, delivery_date,
delivered_quantity) VALUES
(301, 201, 1001, '2025-01-10', 1),
(302, 201, 1002, '2025-01-10', 1),
(303, 202, 1002, '2025-01-20', 1),
(304, 203, 1003, '2025-02-01', 1),
(305, 204, 1004, '2025-02-15', 1);

```

插入结果如下：

#	Time	Action	Message	Duration / Fetch
1	08:33:21	INSERT INTO Subscriber (subscriber_id, name, address, phone) VALUES (1, '张三', '北京市海淀区', '13800000001'), (2, '李四', '...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
2	08:33:58	SELECT * FROM news_manage.subscriber LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
3	08:34:17	INSERT INTO Publication (publication_id, title, frequency, price) VALUES (101, '人民日报', '每日', 2.00), (102, '参考消息', '每日', ...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.031 sec
4	08:34:26	INSERT INTO Order (order_id, subscriber_id, publication_id, order_date, start_date, end_date, quantity) VALUES (1001, 1, 101, '...	Error Code: 1146. Table 'news_manage.order' doesn't exist	0.031 sec
5	08:34:59	INSERT INTO Order1 (order_id, subscriber_id, publication_id, order_date, start_date, end_date, quantity) VALUES (1001, 1, 101, '...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.031 sec
6	08:35:07	INSERT INTO DeliveryCard (card_id, issue_date) VALUES (201, '2025-01-10'), (202, '2025-01-20'), (203, '2025-02-01'), (204, '2025...	Error Code: 1146. Table 'news_manage.deliverycard' doesn't exist	0.000 sec
7	08:35:19	INSERT INTO delivery_card (card_id, issue_date) VALUES (201, '2025-01-10'), (202, '2025-01-20'), (203, '2025-02-01'), (204, '202...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.031 sec
8	08:35:41	INSERT INTO delivery_record (record_id, card_id, order_id, delivery_date, delivered_quantity) VALUES (301, 201, 1001, '2025-01-...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.032 sec

5.2 建立视图

1. 建立视图 v_orders，显示每个订户订阅的报刊名称及订阅份数。

```

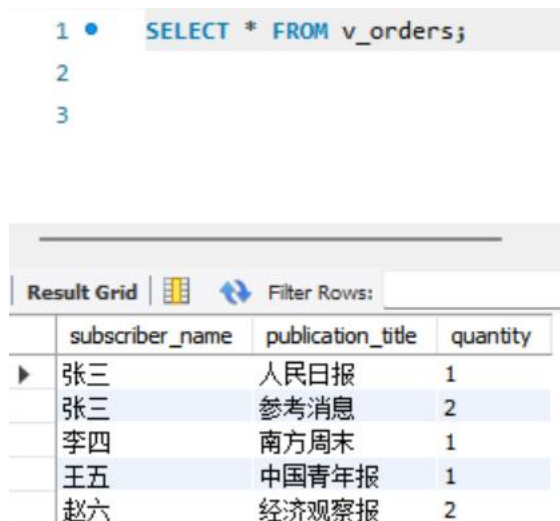
CREATE VIEW v_orders AS
SELECT
    s.name AS subscriber_name,
    p.title AS publication_title,
    o.quantity
FROM
    Subscriber s
JOIN order1 o ON s.subscriber_id = o.subscriber_id

```

```
JOIN Publication p ON o.publication_id = p.publication_id;
```

在视图 v_simple_orders 上执行查询：

```
SELECT * FROM v_orders;
```



The screenshot shows a SQL query editor with the query `SELECT * FROM v_orders;` entered. Below the editor, the 'Result Grid' displays the following data:

	subscriber_name	publication_title	quantity
▶	张三	人民日报	1
	张三	参考消息	2
	李四	南方周末	1
	王五	中国青年报	1
	赵六	经济观察报	2

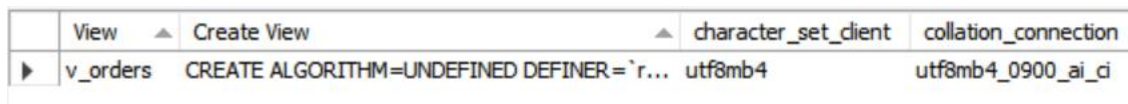
图 5-1 视图一的查询测试

在视图 v_orders 上修改用户权限：

```
GRANT SELECT ON v_orders TO 'abc'@'localhost';
```

查看视图的定义信息：

```
show create view v_orders
```




The screenshot shows the output of the `show create view v_orders` command. It displays the view name, the SQL statement used to create it, and the character set and collation settings.

View	Create View	character_set_client	collation_connection
▶ v_orders	CREATE ALGORITHM=UNDEFINED DEFINER=`r...`	utf8mb4	utf8mb4_0900_ai_ci

图 5-2 查看视图一的定义信息

删除视图：

```
drop view if exists v_orders
```



The screenshot shows a green checkmark icon and the text '20 08:54:49 drop view if exists v_orders', indicating the command was executed successfully.

2. 建立视图 v_delivery, 显示每次投递的投递卡编号、报刊名称和投递数量。

```
CREATE VIEW v_delivery AS
```

```
SELECT
```

```
    dc.card_id,
```

```
    p.title AS publication_title,
```

```
    dr.delivered_quantity
```

```
FROM
```


Delivery_record dr

JOIN delivery_card dc ON dr.card_id = dc.card_id

JOIN order1 o ON dr.order_id = o.order_id

JOIN publication p ON o.publication_id = p.publication_id;

在视图 v_delivery 上执行查询：

```
SELECT * FROM v_delivery
WHERE publication_title = '南方周末'
AND delivered_quantity = 1
AND card_id IN (202, 203);
```

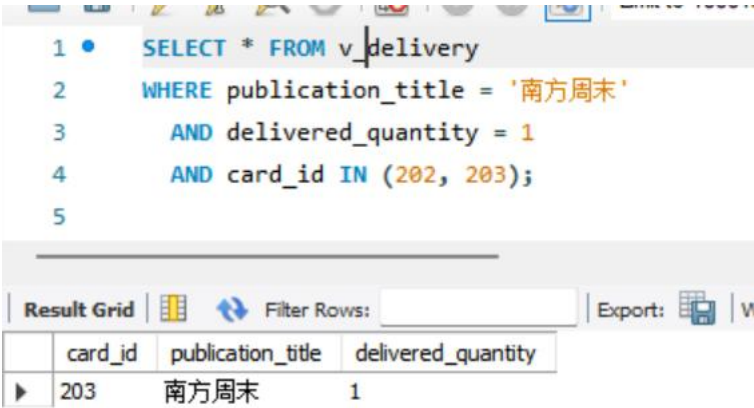


图 5-3 视图二上的查询一

在视图上筛选投递数量<2 的记录：

```
SELECT * FROM v_delivery
WHERE delivered_quantity < 2;
```

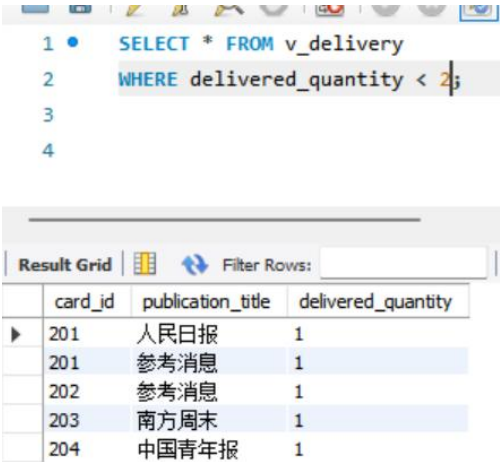


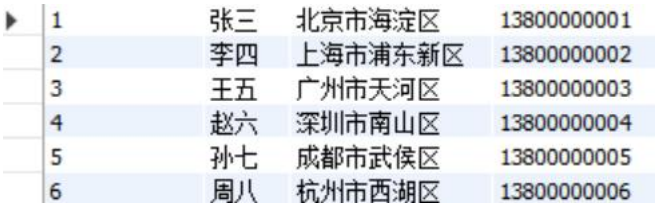
图 5-4 视图二上的查询二

5.3 增删改查测试

5.3.1 插入测试

1. 新增一个订户。

```
INSERT INTO subscriber (subscriber_id, name, address, phone)
VALUES (6, '周八', '杭州市西湖区', '13800000006');
```

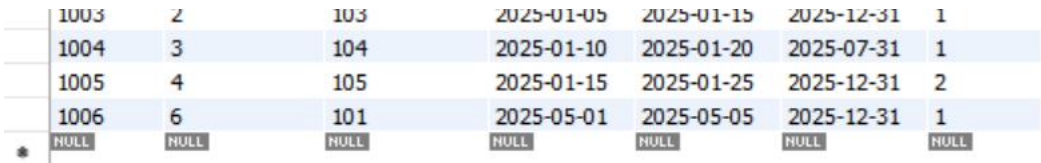


1	张三	北京市海淀区	13800000001
2	李四	上海市浦东新区	13800000002
3	王五	广州市天河区	13800000003
4	赵六	深圳市南山区	13800000004
5	孙七	成都市武侯区	13800000005
6	周八	杭州市西湖区	13800000006

图 5-5 插入测试一

2. 新增一份订单。

```
INSERT INTO order1 (order_id, subscriber_id, publication_id, order_date, start_date,
end_date, quantity)
VALUES (1006, 6, 101, '2025-05-01', '2025-05-05', '2025-12-31', 1);
```



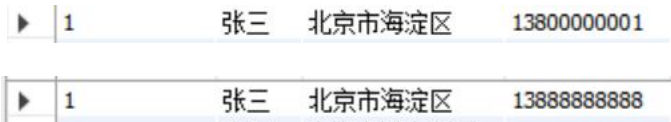
1003	2	103	2025-01-05	2025-01-15	2025-12-31	1
1004	3	104	2025-01-10	2025-01-20	2025-07-31	1
1005	4	105	2025-01-15	2025-01-25	2025-12-31	2
1006	6	101	2025-05-01	2025-05-05	2025-12-31	1
*	NULL	NULL	NULL	NULL	NULL	NULL

图 5-6 插入测试二

5.3.2 修改测试

1. 修改用户张三的电话。

```
UPDATE subscriber
SET phone = '13888888888'
WHERE name = '张三';
```



1	张三	北京市海淀区	13800000001
1	张三	北京市海淀区	13888888888

图 5-7 修改测试一

2. 修改订单的订阅数量。

```
UPDATE order1
SET quantity = 3
WHERE order_id = 1002;
```

	order_id	subscriber_id	publication_id	order_date	start_date	end_date	quantity
▶	1001	1	101	2025-01-01	2025-01-05	2025-06-30	1
	1002	1	102	2025-01-02	2025-01-10	2025-06-30	2

	order_id	subscriber_id	publication_id	order_date	start_date	end_date	quantity
▶	1001	1	101	2025-01-01	2025-01-05	2025-06-30	1
	1002	1	102	2025-01-02	2025-01-10	2025-06-30	3

图 5-8 修改测试二

5.3.3 删除测试

1. 删除特定电话号码的用户。

	subscriber_id	name	address	phone
▶	1	张三	北京市海淀区	13888888888
	2	李四	上海市浦东新区	13800000002
	3	王五	广州市天河区	13800000003
	4	赵六	深圳市南山区	13800000004
	5	孙七	成都市武侯区	13800000005
	6	周八	杭州市西湖区	13800000006

图 5-9 删除测试一删除前

```
DELETE FROM subscriber
WHERE phone = 13800000006;
```

	subscriber_id	name	address	phone
▶	1	张三	北京市海淀区	13888888888
	2	李四	上海市浦东新区	13800000002
	3	王五	广州市天河区	13800000003
	4	赵六	深圳市南山区	13800000004
	5	孙七	成都市武侯区	13800000005
*	NULL	NULL	NULL	NULL

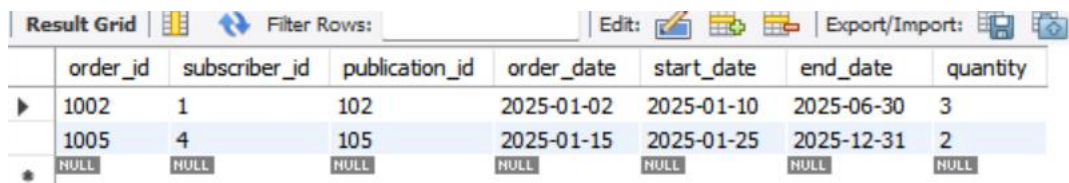
图 5-10 删除测试一删除后

2. 删除订阅份数为 1 的订单。

	order_id	subscriber_id	publication_id	order_date	start_date	end_date	quantity
▶	1001	1	101	2025-01-01	2025-01-05	2025-06-30	1
	1003	2	103	2025-01-05	2025-01-15	2025-12-31	1
	1004	3	104	2025-01-10	2025-01-20	2025-07-31	1
	1006	6	101	2025-05-01	2025-05-05	2025-12-31	1
	1005	4	105	2025-01-15	2025-01-25	2025-12-31	2
	1002	1	102	2025-01-02	2025-01-10	2025-06-30	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 5-11 删除测试二删除前

```
DELETE FROM order1
WHERE quantity = 1;
```



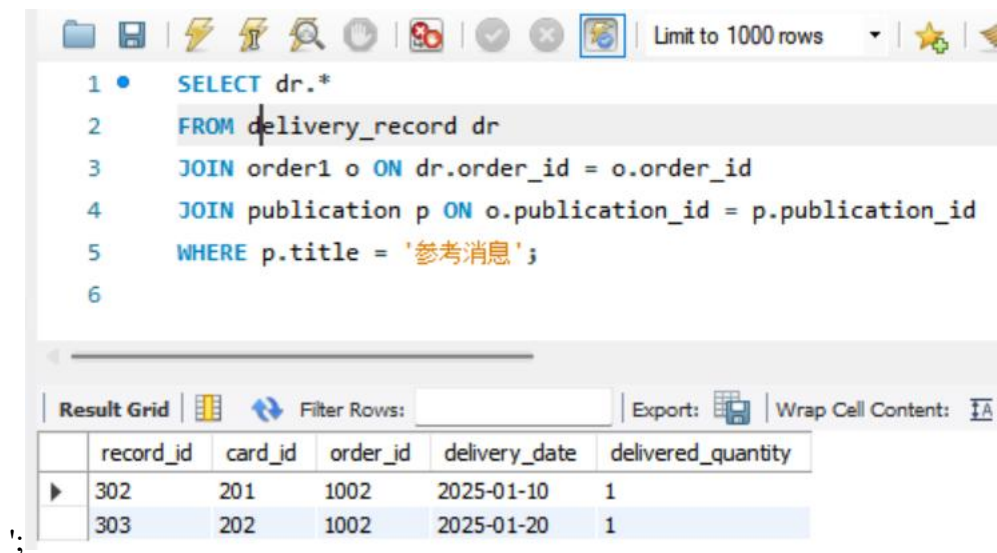
	order_id	subscriber_id	publication_id	order_date	start_date	end_date	quantity
▶	1002	1	102	2025-01-02	2025-01-10	2025-06-30	3
	1005	4	105	2025-01-15	2025-01-25	2025-12-31	2
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 5-12 删除测试二删除后

5.3.4 查询测试

1. 查询所有投递记录中投递了《参考消息》的记录。

```
SELECT dr.*
FROM delivery_record dr
JOIN order1 o ON dr.order_id = o.order_id
JOIN publication p ON o.publication_id = p.publication_id
WHERE p.title = '参考消息'
```



```
1 • SELECT dr.*
2 FROM delivery_record dr
3 JOIN order1 o ON dr.order_id = o.order_id
4 JOIN publication p ON o.publication_id = p.publication_id
5 WHERE p.title = '参考消息';
6
```

	record_id	card_id	order_id	delivery_date	delivered_quantity
▶	302	201	1002	2025-01-10	1
	303	202	1002	2025-01-20	1

图 5-13 查询测试一

2. 查询每位订户的订单数量。

```
SELECT s.name, COUNT(o.order_id) AS total_orders
FROM subscriber s
JOIN order1 o ON s.subscriber_id = o.subscriber_id
GROUP BY s.name;
```

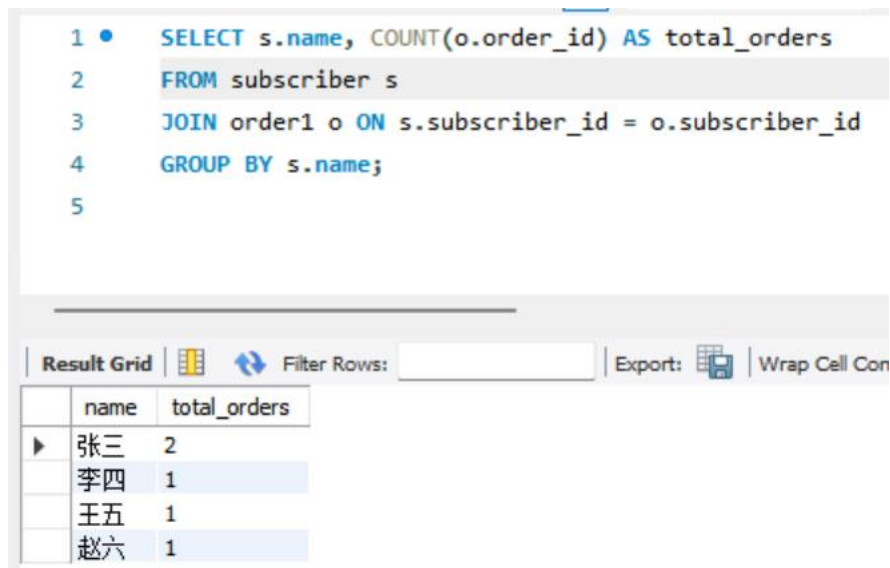


图 5-14 查询测试二

5.4 建立并测试触发器

1. 建立一个触发器，禁止对报刊目录（publication）进行删除操作。

```
DELIMITER //
```

```
CREATE TRIGGER trg_prevent_publication_delete
BEFORE DELETE ON publication
FOR EACH ROW
BEGIN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = '禁止删除 Publication 表中的数据!';
END;
//
```

```
DELIMITER ;
```

建立成功后，尝试删除：

```
DELETE FROM Publication WHERE publication_id = 101;
```

86 09:50:02 DELETE FROM Publication WHERE publication_id = 101	Error Code: 1644. 禁止删除 Publication 表中的数据!	0.000 sec
----------------------------------------------------------------	-------------------------------------------	-----------

图 5-15 触发器一效果

2. 建立一个触发器，当 Order 表中新插入的数据如果未填写 order_date，则自动设置为当前时间。

```
DELIMITER //
```

```
CREATE TRIGGER trg_order_set_date
BEFORE INSERT ON order1
FOR EACH ROW
BEGIN
    IF NEW.order_date IS NULL THEN
        SET NEW.order_date = CURRENT_DATE();
    END IF;
END;
//
```

```
DELIMITER ;
```

插入前的数据表如图 5-19 所示。插入一条订单信息后：

```
INSERT INTO order1 (order_id, subscriber_id, publication_id, start_date, end_date,
                    quantity)
VALUES (1007, 1, 101, '2025-06-01', '2025-12-31', 1);
```

	order_id	subscriber_id	publication_id	order_date	start_date	end_date	quantity
▶	1001	1	101	2025-01-01	2025-01-05	2025-06-30	1
	1002	1	102	2025-01-02	2025-01-10	2025-06-30	3
	1003	2	103	2025-01-05	2025-01-15	2025-12-31	1
	1004	3	104	2025-01-10	2025-01-20	2025-07-31	1
	1005	4	105	2025-01-15	2025-01-25	2025-12-31	2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 5-19 触发器测试二插入前

	order_id	subscriber_id	publication_id	order_date	start_date	end_date	quantity
▶	1001	1	101	2025-01-01	2025-01-05	2025-06-30	1
	1002	1	102	2025-01-02	2025-01-10	2025-06-30	3
	1003	2	103	2025-01-05	2025-01-15	2025-12-31	1
	1004	3	104	2025-01-10	2025-01-20	2025-07-31	1
	1005	4	105	2025-01-15	2025-01-25	2025-12-31	2
	1007	1	101	2025-05-23	2025-06-01	2025-12-31	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 5-20 触发器测试二插入后

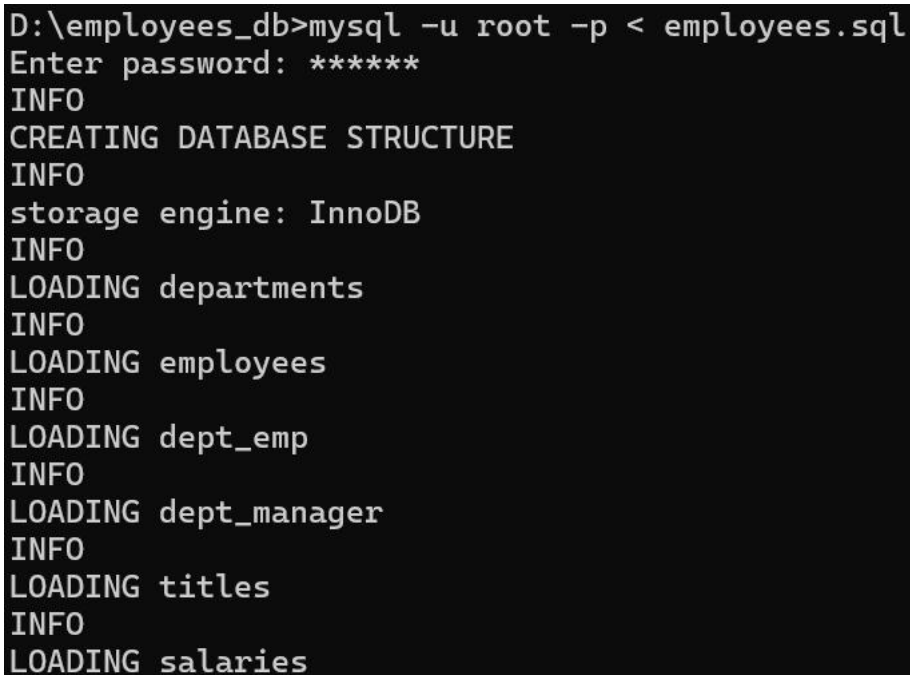
可以看到 1007 号订单的 order_date 为 2025-5-23 即为当前时间。

6. 数据库建模实验总结

实验中所要求的各项内容均已顺利完成，所建立的数据库在增、删、改、查操作及触发器测试过程中均运行正常，无异常情况。此次实验全面锻炼了我数据库设计方面的实践能力，包括从需求分析到 E-R 图绘制、逻辑结构设计以及数据库的实际创建等全过程。同时，也加深了我对 SQL 语言的理解和掌握，特别是在数据操作（DML）和触发器等方面的应用能力得到了有效提升。通过本次实验，我对数据库系统的整体构建过程以及 SQL 语言在数据管理中的关键作用有了更深入的理解。

7. 样本数据库的安装和数据加载

下载样本数据库 employees_db，依据相关说明完成样本数据库的安装与加载。



```
D:\employees_db>mysql -u root -p < employees.sql
Enter password: *****
INFO
CREATING DATABASE STRUCTURE
INFO
storage engine: InnoDB
INFO
LOADING departments
INFO
LOADING employees
INFO
LOADING dept_emp
INFO
LOADING dept_manager
INFO
LOADING titles
INFO
LOADING salaries
```

图 7-1 样本数据库的加载

8. 样本数据库的分析

根据样本数据库的结构绘制 E-R 图。

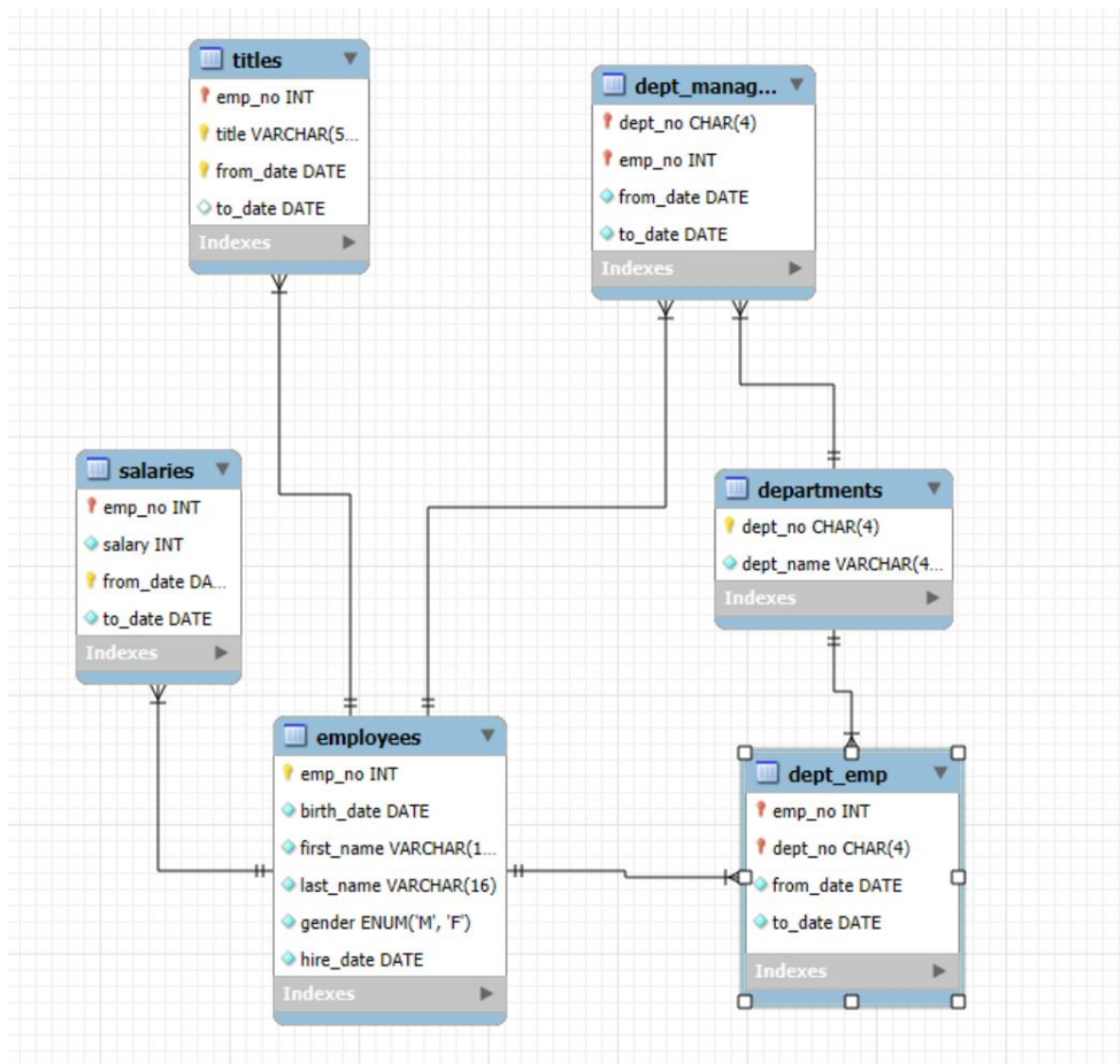


图 8-1 样本数据库的 E-R 图

9. 样本数据库上的查询

9.1 查询一

查询每个部门(departments)的编号(dept_no),名称(dept_name),在该部门工作过的雇员(employees)人数,最低工资(salary),平均工资,最高工资及工资总额;

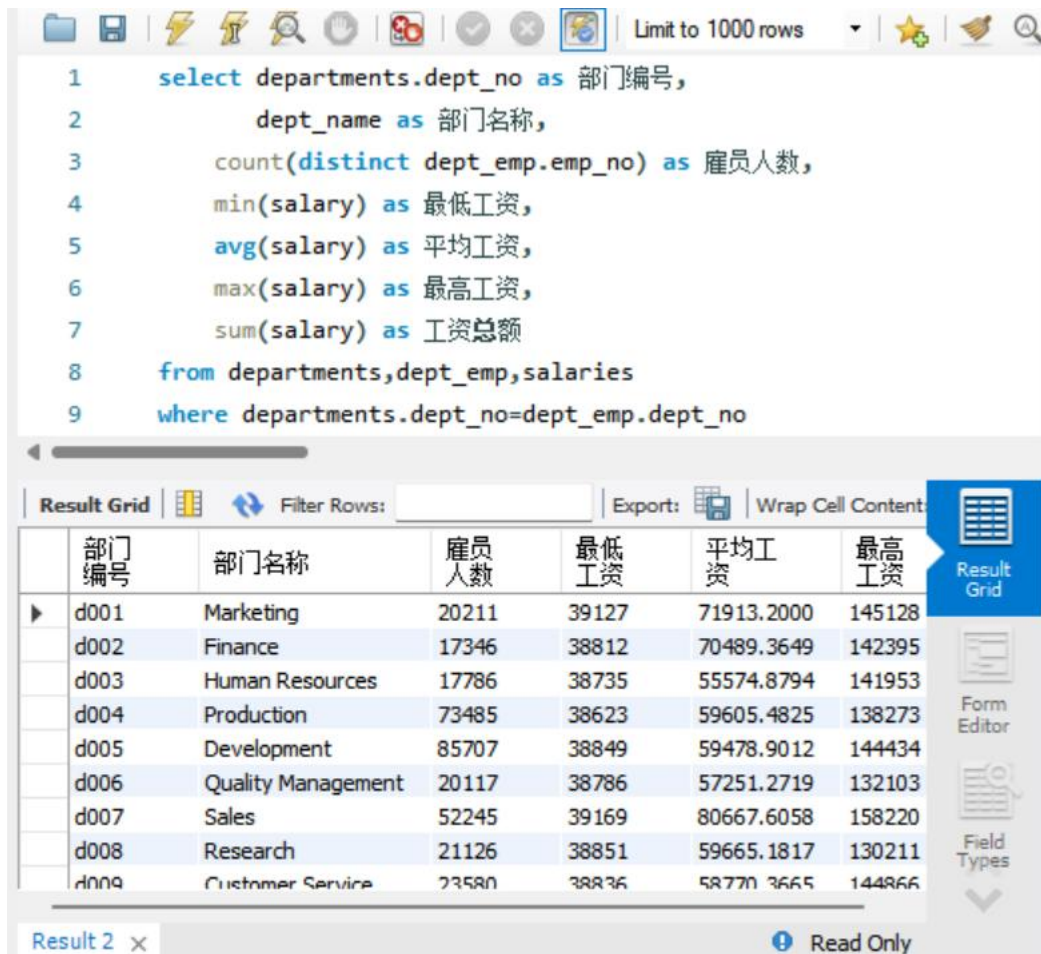
```
select departments.dept_no as 部门编号,
       dept_name as 部门名称,
       count(distinct dept_emp.emp_no) as 雇员人数,
       min(salary) as 最低工资,
```



```

avg(salary) as 平均工资,
max(salary) as 最高工资,
sum(salary) as 工资总额
from departments,dept_emp,salaries
where departments.dept_no=dept_emp.dept_no
and dept_emp.emp_no=salaries.emp_no
group by departments.dept_no

```



The screenshot shows a database query tool interface. The top part displays the SQL query code, and the bottom part shows the results in a table format. The query is a grouped aggregate query that calculates various statistics for each department.

SQL Query:

```

1 select departments.dept_no as 部门编号,
2    dept_name as 部门名称,
3    count(distinct dept_emp.emp_no) as 雇员人数,
4    min(salary) as 最低工资,
5    avg(salary) as 平均工资,
6    max(salary) as 最高工资,
7    sum(salary) as 工资总额
8 from departments,dept_emp,salaries
9 where departments.dept_no=dept_emp.dept_no

```

Result Grid:

部门编号	部门名称	雇员人数	最低工资	平均工资	最高工资
d001	Marketing	20211	39127	71913.2000	145128
d002	Finance	17346	38812	70489.3649	142395
d003	Human Resources	17786	38735	55574.8794	141953
d004	Production	73485	38623	59605.4825	138273
d005	Development	85707	38849	59478.9012	144434
d006	Quality Management	20117	38786	57251.2719	132103
d007	Sales	52245	39169	80667.6058	158220
d008	Research	21126	38851	59665.1817	130211
d009	Customer Service	73580	38836	58770.3665	144866

图 9-1 查询一的查询代码及结果

9.2 查询二

查询每个部门(departments)的编号(dept_no),名称(dept_name),及各个时间段(from_date,to_date)担任该部门经理(dept_manager)的雇员的编号(emp_no)和姓名(first_name+last_name),并按时间段先后显式;

```

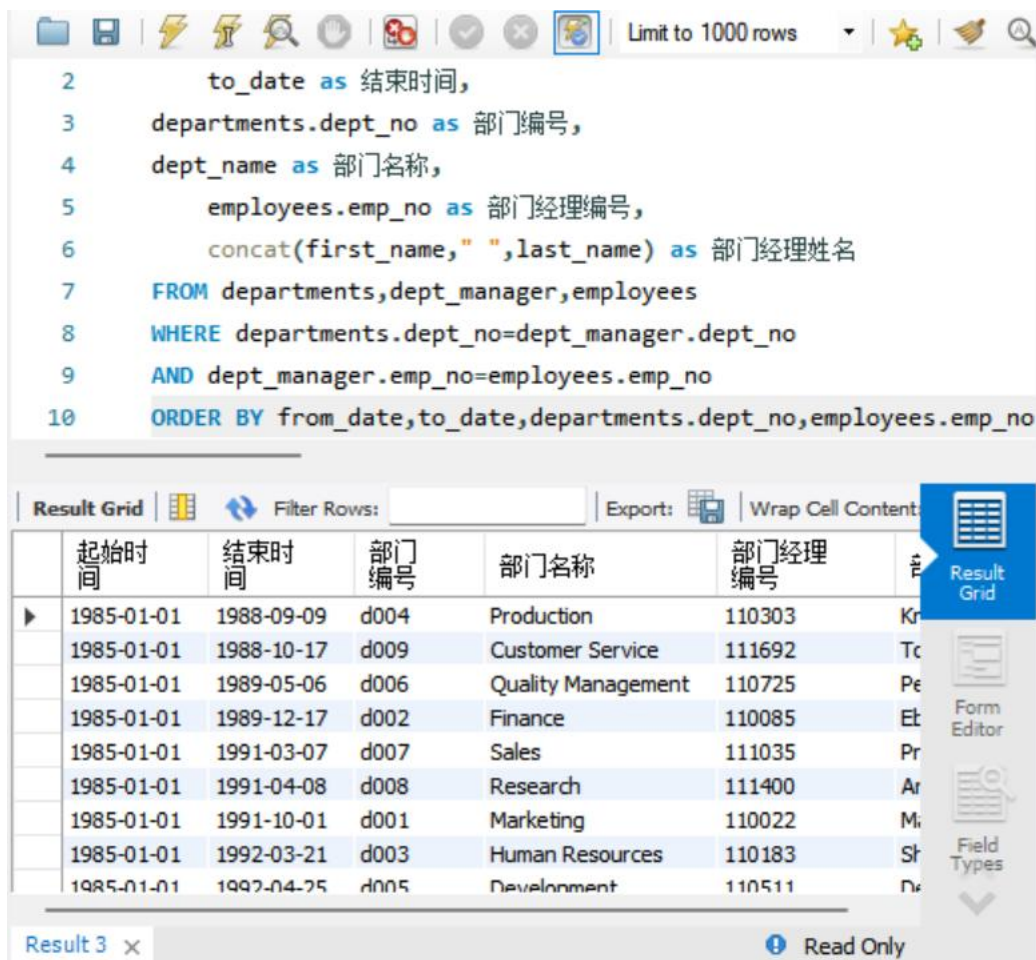
select from_date as 起始时间,
       to_date as 结束时间,

```

```

departments.dept_no as 部门编号,
dept_name as 部门名称,
employees.emp_no as 部门经理编号,
concat(first_name," ",last_name) as 部门经理姓名
from departments,dept_manager,employees
where departments.dept_no=dept_manager.dept_no
and dept_manager.emp_no=employees.emp_no
order by from_date,to_date,departments.dept_no,employees.emp_no

```



The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

```

2      to_date as 结束时间,
3      departments.dept_no as 部门编号,
4      dept_name as 部门名称,
5      employees.emp_no as 部门经理编号,
6      concat(first_name," ",last_name) as 部门经理姓名
7  FROM departments,dept_manager,employees
8  WHERE departments.dept_no=dept_manager.dept_no
9  AND dept_manager.emp_no=employees.emp_no
10 ORDER BY from_date,to_date,departments.dept_no,employees.emp_no

```

Below the query, the results are displayed in a grid. The grid has 7 columns: 起始时间 (Start Date), 结束时间 (End Date), 部门编号 (Department ID), 部门名称 (Department Name), 部门经理编号 (Manager ID), 部门经理姓名 (Manager Name), and 部门 (Department). The results are as follows:

起始时间	结束时间	部门编号	部门名称	部门经理编号	部门经理姓名	部门
1985-01-01	1988-09-09	d004	Production	110303	Kr	
1985-01-01	1988-10-17	d009	Customer Service	111692	To	
1985-01-01	1989-05-06	d006	Quality Management	110725	Pe	
1985-01-01	1989-12-17	d002	Finance	110085	El	
1985-01-01	1991-03-07	d007	Sales	111035	Pr	
1985-01-01	1991-04-08	d008	Research	111400	Ar	
1985-01-01	1991-10-01	d001	Marketing	110022	Mi	
1985-01-01	1992-03-21	d003	Human Resources	110183	St	
1985-01-01	1992-04-25	d005	Development	110511	De	

图 9-2 查询二的查询代码及结果

9.3 查询三

查询每位雇员的编号(emp_no),姓名(first_name+last_name),及各个时间段(from_date,end_date)的工资额(salary),并按时间段先后显式;

```

select employees.emp_no as 雇员编号,
concat(first_name," ",last_name) as 雇员姓名,

```

```

from_date as 起始时间,
to_date as 结束时间,
salary as 工资额
from employees,salaries
where employees.emp_no=salaries.emp_no
order by from_date,to_date,employees.emp_no

```

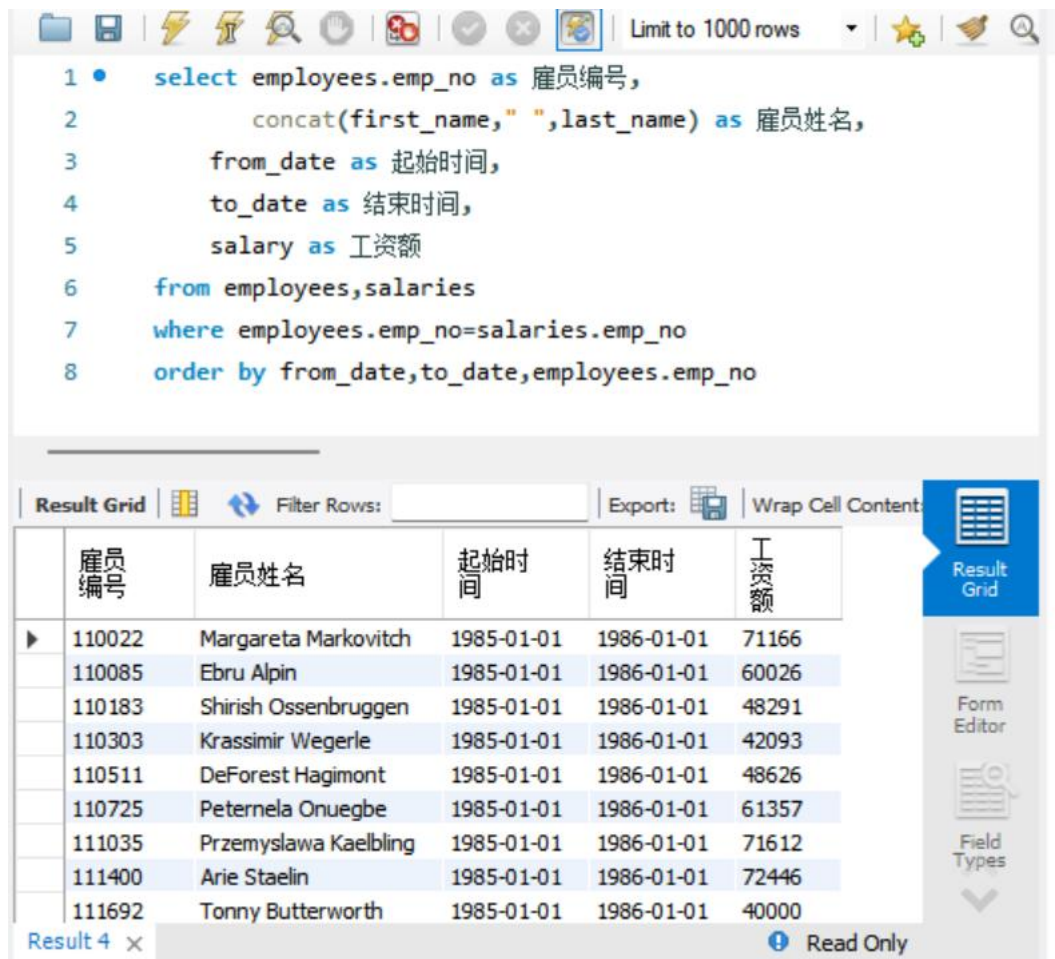


图 9-3 查询三的查询代码及结果

9.4 查询四

查询每位雇员的编号(emp_no),姓名(first_name+last_name),及各个时间段(from_date,end_date)的工作部门名称(dept_name),并按时间段先后显式;

```

select employees.emp_no as 雇员编号,
concat(first_name," ",last_name) as 雇员姓名,
from_date as 起始时间,
to_date as 结束时间,

```

```

dept_name as 工作部门名称
from employees,dept_emp,departments
where employees.emp_no=dept_emp.emp_no
and departments.dept_no=dept_emp.dept_no
order by from_date,to_date,employees.emp_no

```

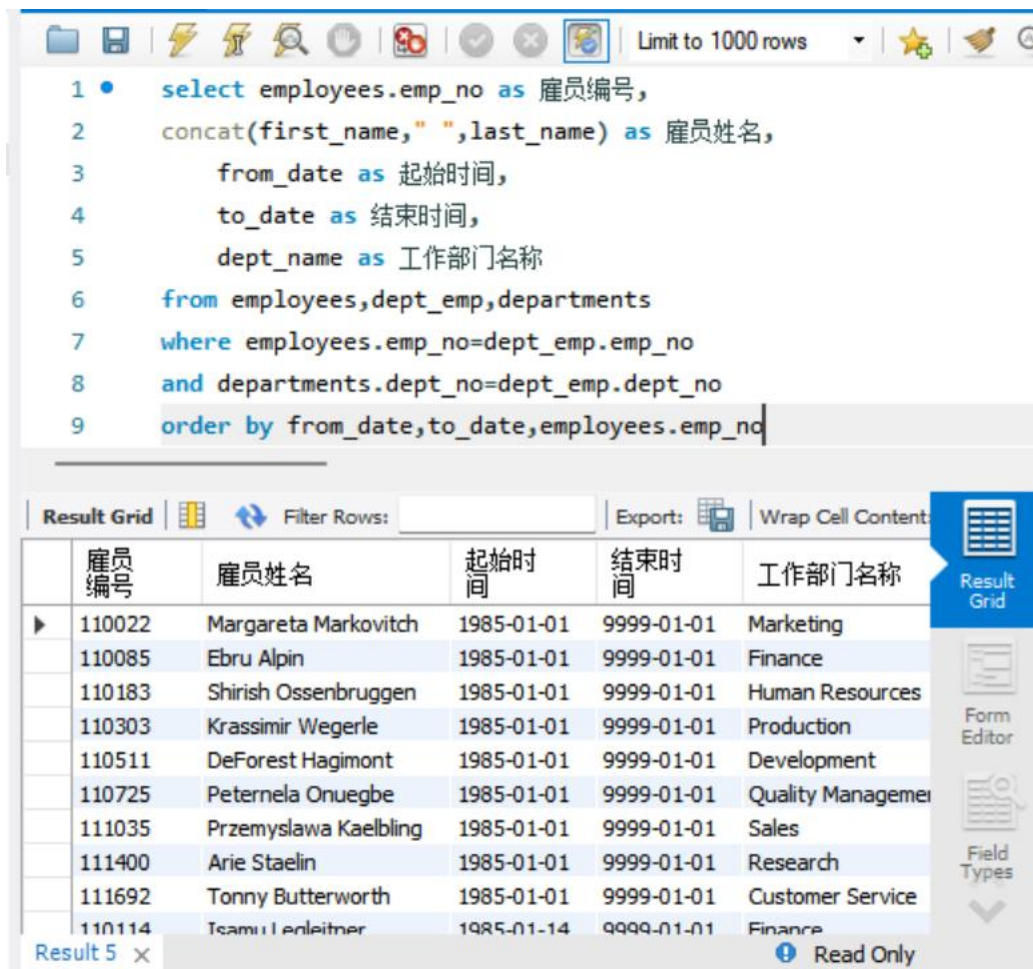


图 9-4 查询四的查询代码及结果

9.5 查询五

查询每位雇员的编号(emp_no),姓名(first_name+last_name),及任职过的部门数;

```

select employees.emp_no as 雇员编号,
       concat(first_name," ",last_name) as 雇员姓名,
       count(distinct dept_no) as 曾任职部门数
from employees,dept_emp
where employees.emp_no=dept_emp.emp_no

```


group by employees.emp_no

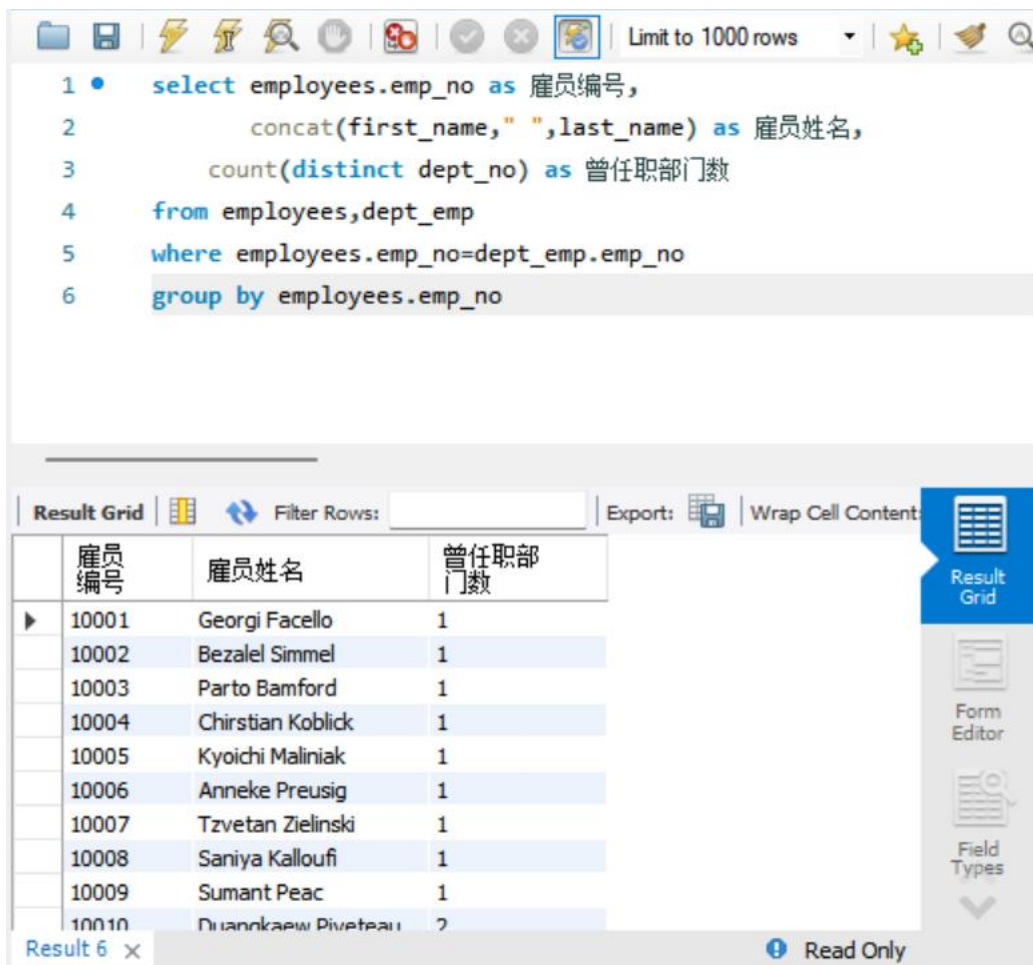


图 9-5 查询五的查询代码及结果

9.6 查询六

查询每位雇员的编号(emp_no),姓名(first_name+last_name),及各个时间段(from_date,end_date)担任的职务(title),并按时间段先后显式;

```

select employees.emp_no as 雇员编号,
       concat(first_name," ",last_name) as 雇员姓名,
       from_date as 起始时间,
       to_date as 结束时间,
       title as 职位
from employees,titles
where employees.emp_no=titles.emp_no
order by from_date,to_date,employees.emp_no

```

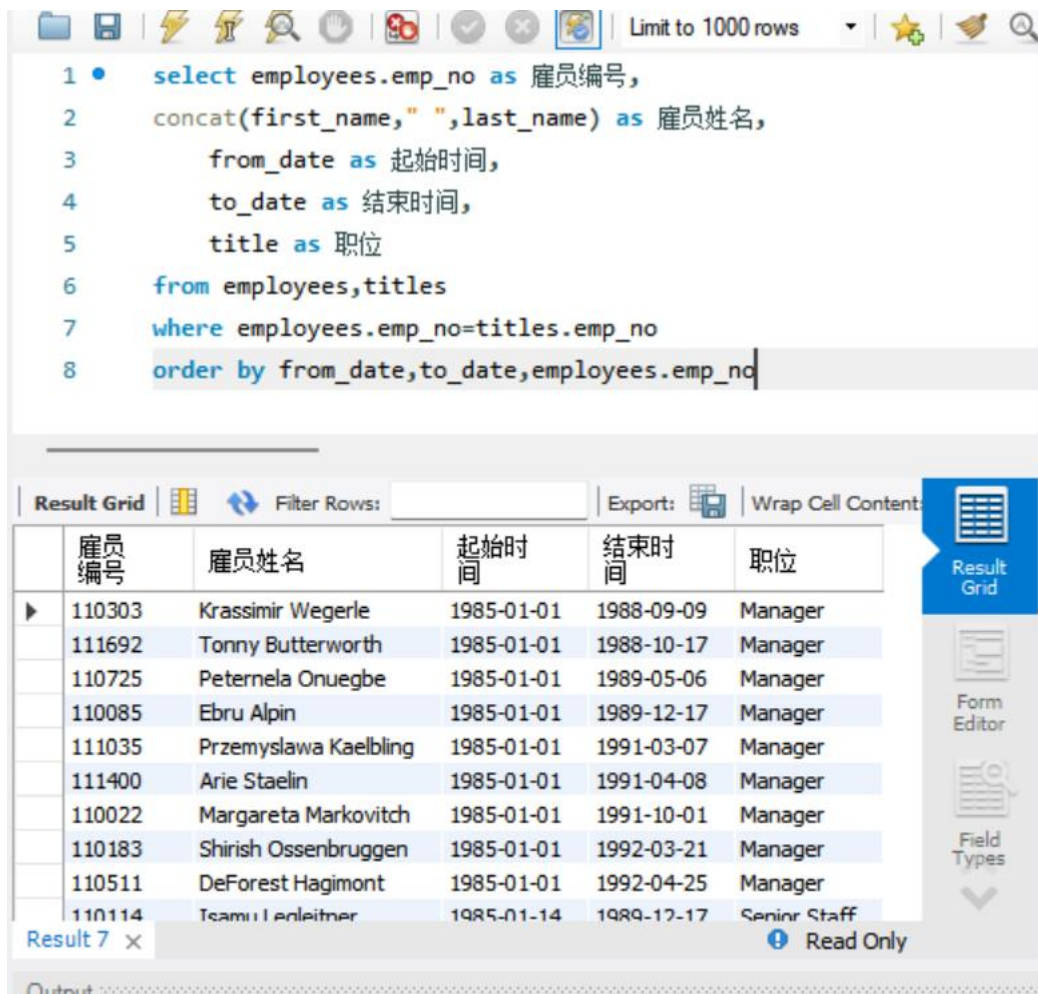


图 9-6 查询六的查询代码及结果

9.7 查询七

查询担任每种职务(title)的雇员人数;

select title as 职位,

count(distinct emp_no) as 人数

from titles

group by title

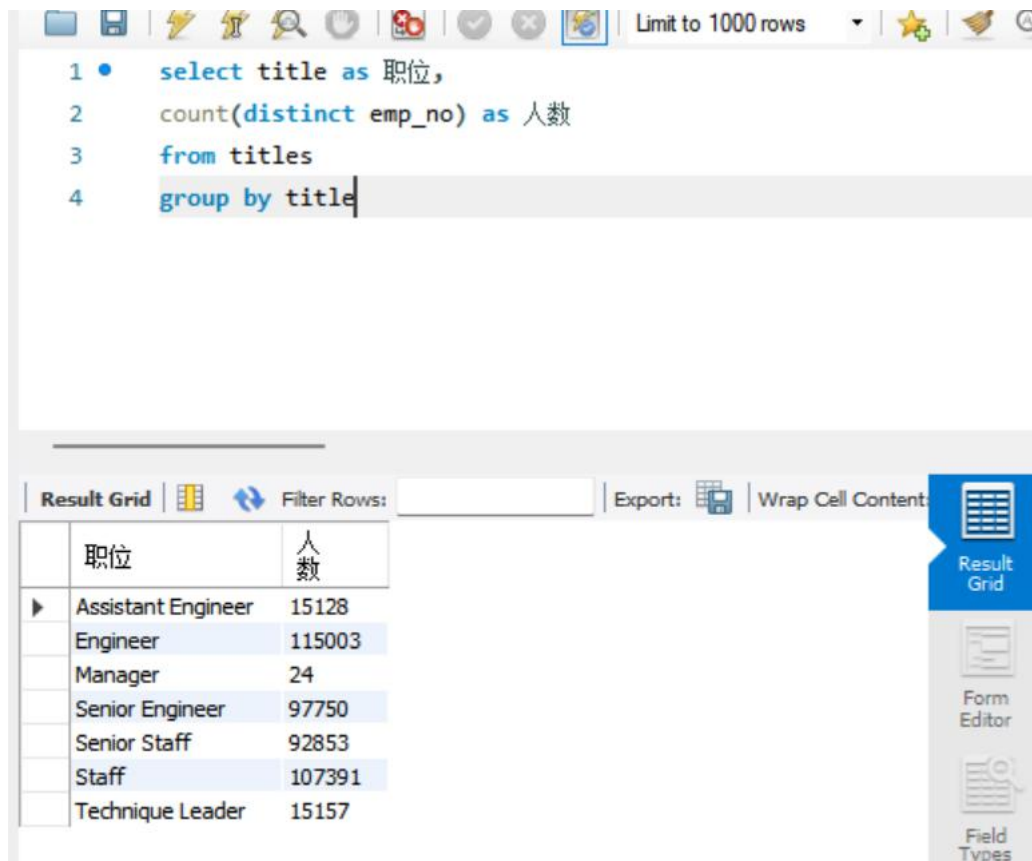


图 9-7 查询七的查询代码及结果

9.8 查询八

查询每个部门中担任每种职务(title)的雇员人数:

```

select dept_emp.dept_no as 部门编号,
dept_name as 部门名称,
    title as 职位,
count(distinct titles.emp_no) as 人数
from titles,dept_emp,departments
where titles.emp_no=dept_emp.emp_no
and dept_emp.dept_no=departments.dept_no
group by dept_emp.dept_no,title

```

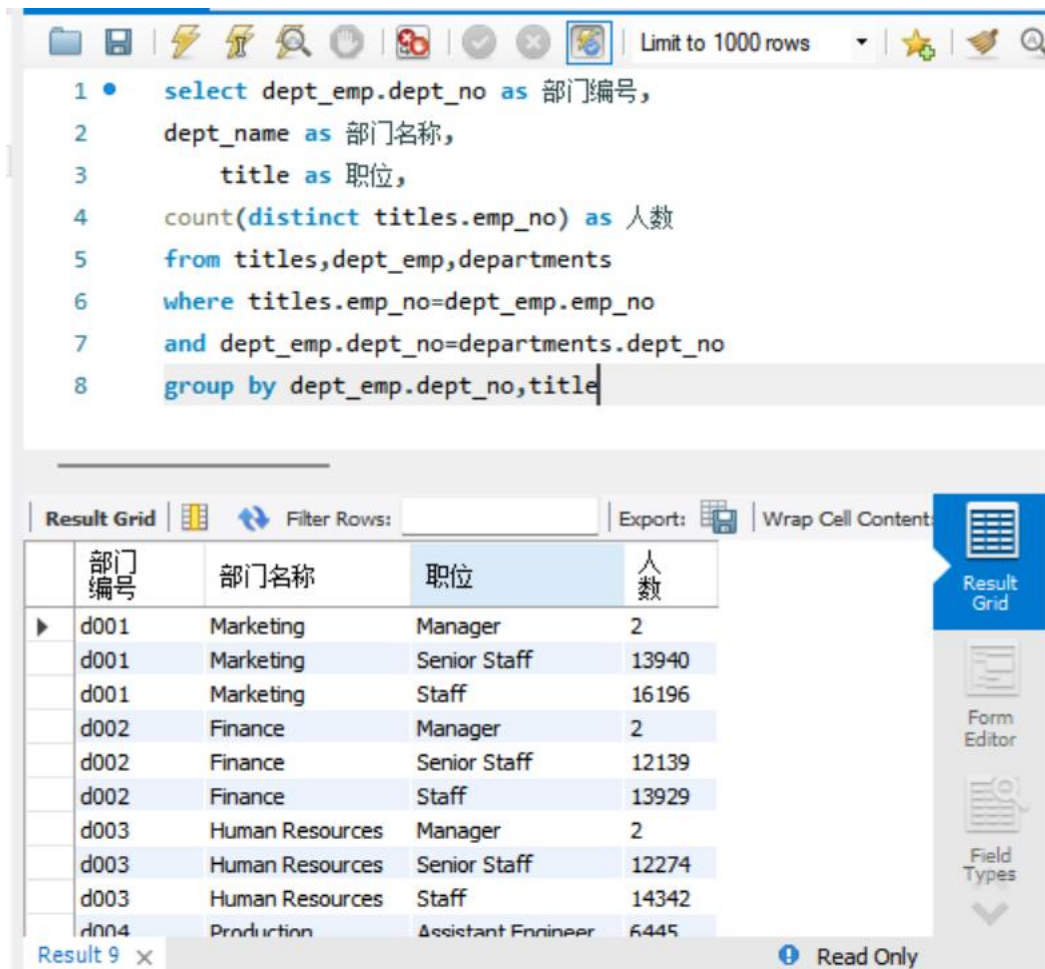


图 9-8 查询八的查询代码及结果

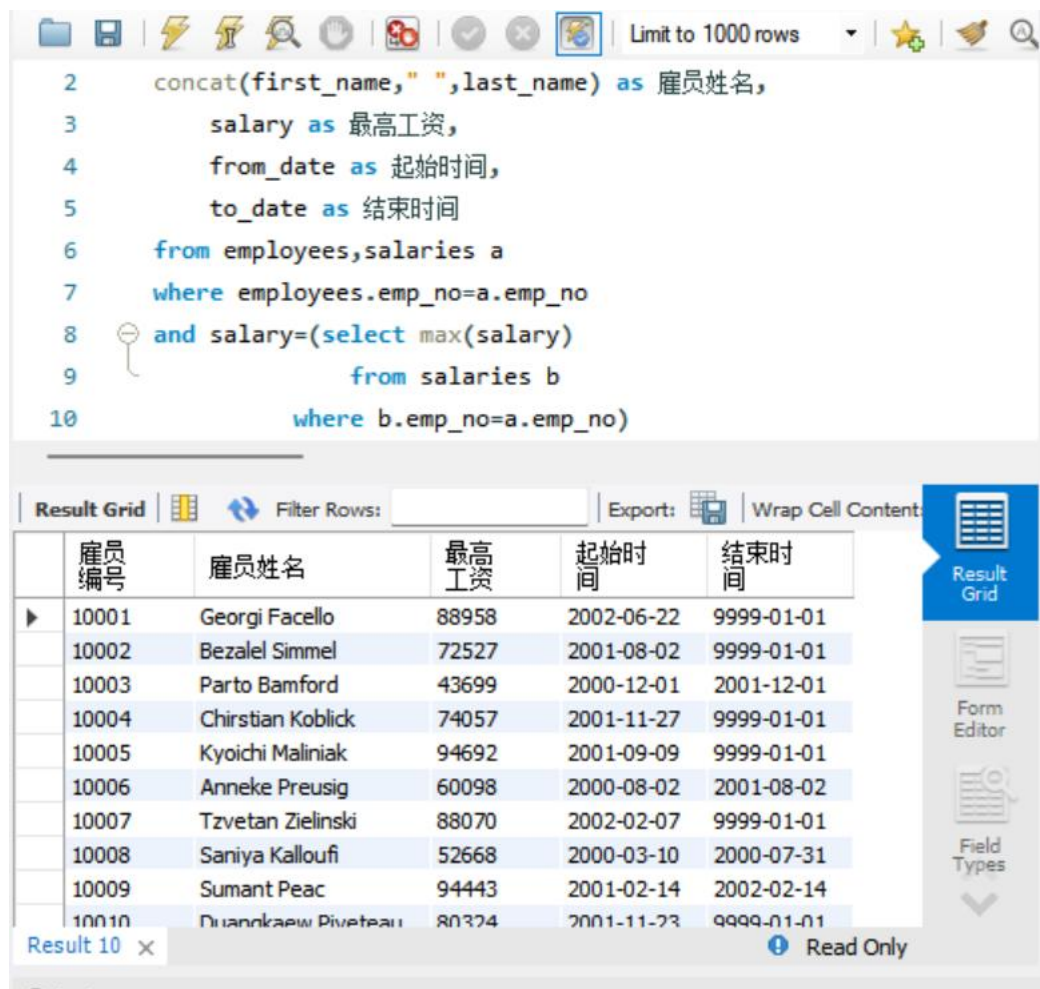
9.9 查询九

查询每位雇员的编号(emp_no),姓名(first_name+last_name),及工资额最高的时间段(from_date,end_data)及其工资额;

```

select employees.emp_no as 雇员编号,
concat(first_name," ",last_name) as 雇员姓名,
salary as 最高工资,
from_date as 起始时间,
to_date as 结束时间
from employees,salaries a
where employees.emp_no=a.emp_no
and salary=(select max(salary)
            from salaries b
            where b.emp_no=a.emp_no)

```

The screenshot shows a database query editor with the following SQL code:

```

2  concat(first_name," ",last_name) as 雇员姓名,
3  salary as 最高工资,
4  from_date as 起始时间,
5  to_date as 结束时间
6  from employees,salaries a
7  where employees.emp_no=a.emp_no
8  and salary=(select max(salary)
9               from salaries b
10              where b.emp_no=a.emp_no)

```

The results are displayed in a table with the following columns: 雇员编号, 雇员姓名, 最高工资, 起始时间, 结束时间. The table contains 10 rows of data.

雇员编号	雇员姓名	最高工资	起始时间	结束时间
10001	Georgi Facello	88958	2002-06-22	9999-01-01
10002	Bezalel Simmel	72527	2001-08-02	9999-01-01
10003	Parto Bamford	43699	2000-12-01	2001-12-01
10004	Chirstian Koblick	74057	2001-11-27	9999-01-01
10005	Kyoichi Maliniak	94692	2001-09-09	9999-01-01
10006	Anneke Preusig	60098	2000-08-02	2001-08-02
10007	Tzvetan Zielinski	88070	2002-02-07	9999-01-01
10008	Saniya Kalloufi	52668	2000-03-10	2000-07-31
10009	Sumant Peac	94443	2001-02-14	2002-02-14
10010	Diana Kravetz	80374	2001-11-23	9999-01-01

图 9-9 查询九的查询代码及结果

9.10 查询十

查询所有曾经担任过部门经理(dept_manager)的雇员的编号(emp_no)和姓名(first_name+last_name);

```

select employees.emp_no as 雇员编号,
concat(first_name," ",last_name) as 雇员姓名
from employees,dept_manager
where employees.emp_no=dept_manager.emp_no

```

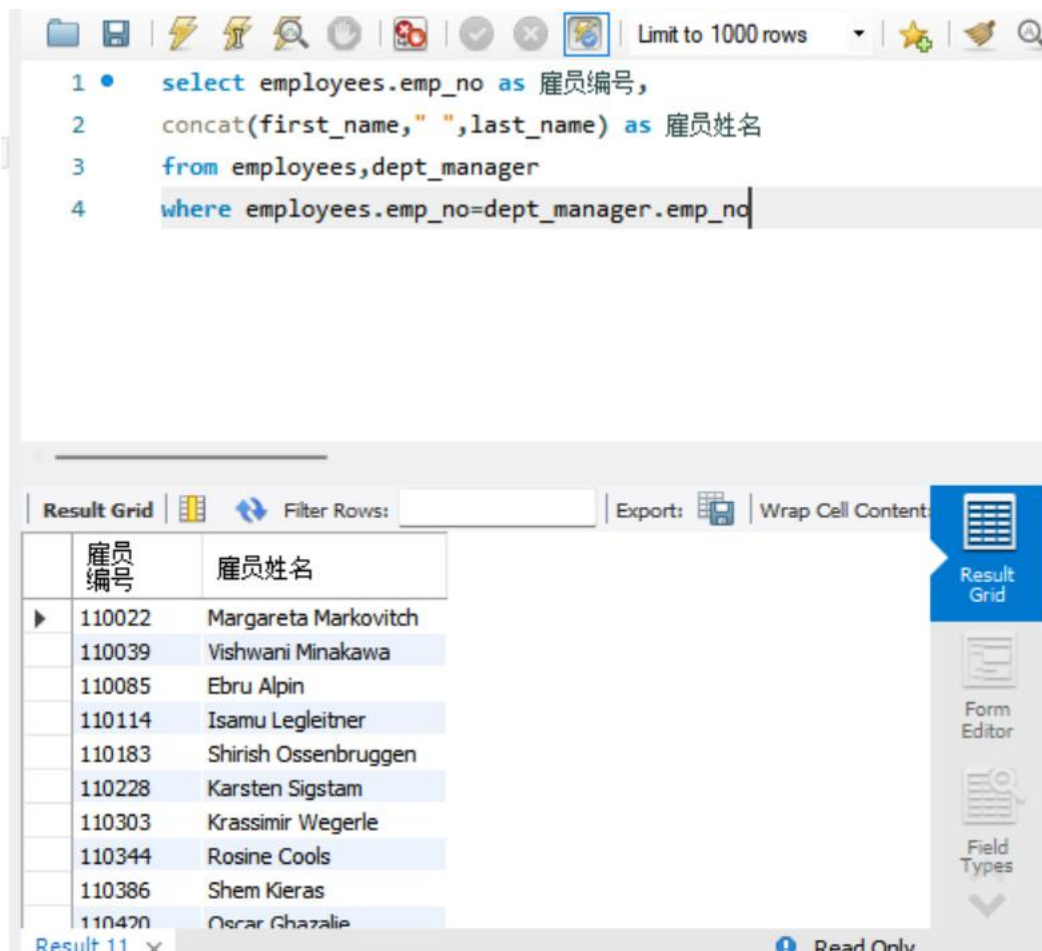


图 9-10 查询十的查询代码及结果

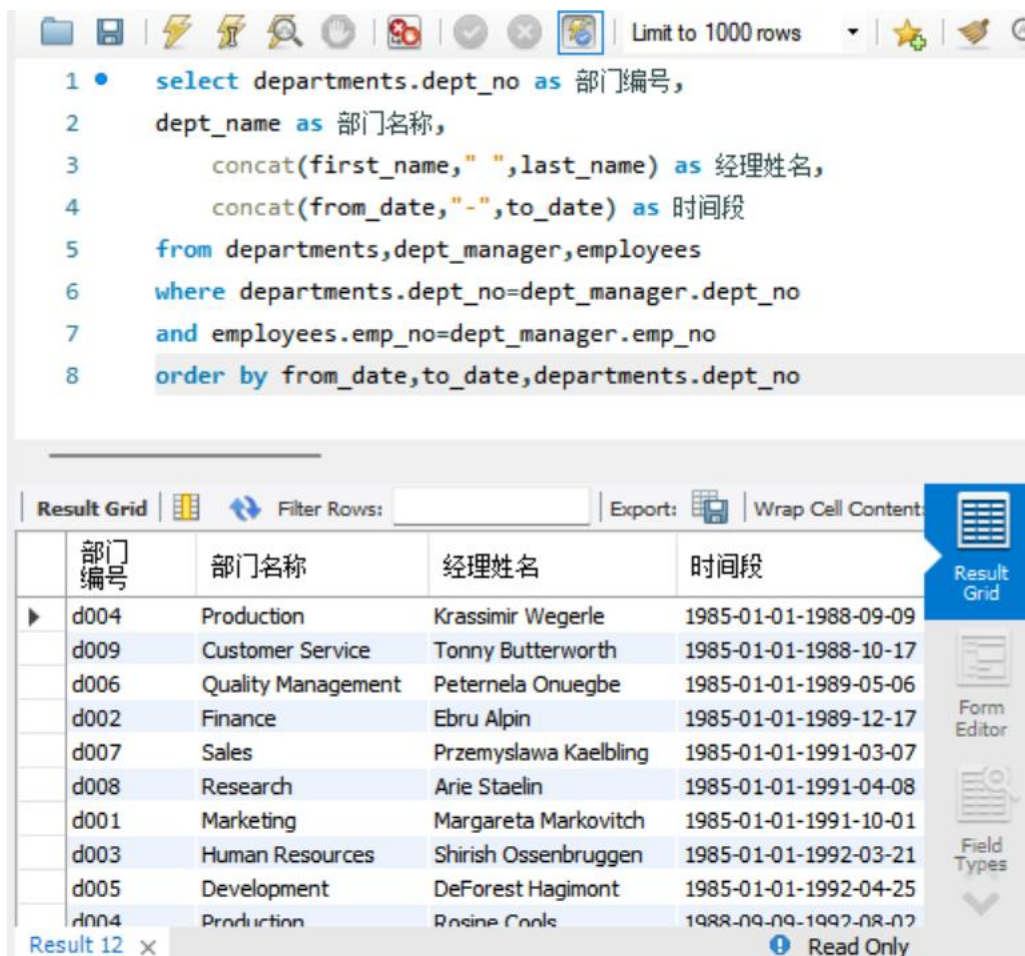
9.11 查询十一

按时间段 (from_date,end_date) 先后列出每个部门 (departments) 的编号 (dept_no), 名称 (dept_name), 及其经理的姓名 (first_name+last_name);

```

select departments.dept_no as 部门编号,
       dept_name as 部门名称,
       concat(first_name, " ", last_name) as 经理姓名,
       concat(from_date, "-", to_date) as 时间段
from departments, dept_manager, employees
where departments.dept_no=dept_manager.dept_no
and employees.emp_no=dept_manager.emp_no
order by from_date, to_date, departments.dept_no

```



The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

```

1 select departments.dept_no as 部门编号,
2   dept_name as 部门名称,
3   concat(first_name," ",last_name) as 经理姓名,
4   concat(from_date,"-",to_date) as 时间段
5 from departments,dept_manager,employees
6 where departments.dept_no=dept_manager.dept_no
7    and employees.emp_no=dept_manager.emp_no
8 order by from_date,to_date,departments.dept_no

```

Below the query editor, the results are displayed in a grid. The grid has five columns: 部门编号, 部门名称, 经理姓名, and 时间段. The results are as follows:

部门编号	部门名称	经理姓名	时间段
d004	Production	Krassimir Wegerle	1985-01-01-1988-09-09
d009	Customer Service	Tonny Butterworth	1985-01-01-1988-10-17
d006	Quality Management	Peternela Onuegbie	1985-01-01-1989-05-06
d002	Finance	Ebru Alpin	1985-01-01-1989-12-17
d007	Sales	Przemyslaw Kaelbling	1985-01-01-1991-03-07
d008	Research	Arie Staelin	1985-01-01-1991-04-08
d001	Marketing	Margareta Markovitch	1985-01-01-1991-10-01
d003	Human Resources	Shirish Ossenbruggen	1985-01-01-1992-03-21
d005	Development	DeForest Hagimont	1985-01-01-1992-04-25
d004	Production	Rosine Coles	1988-09-09-1997-08-07

图 9-11 查询十一的查询代码及结果

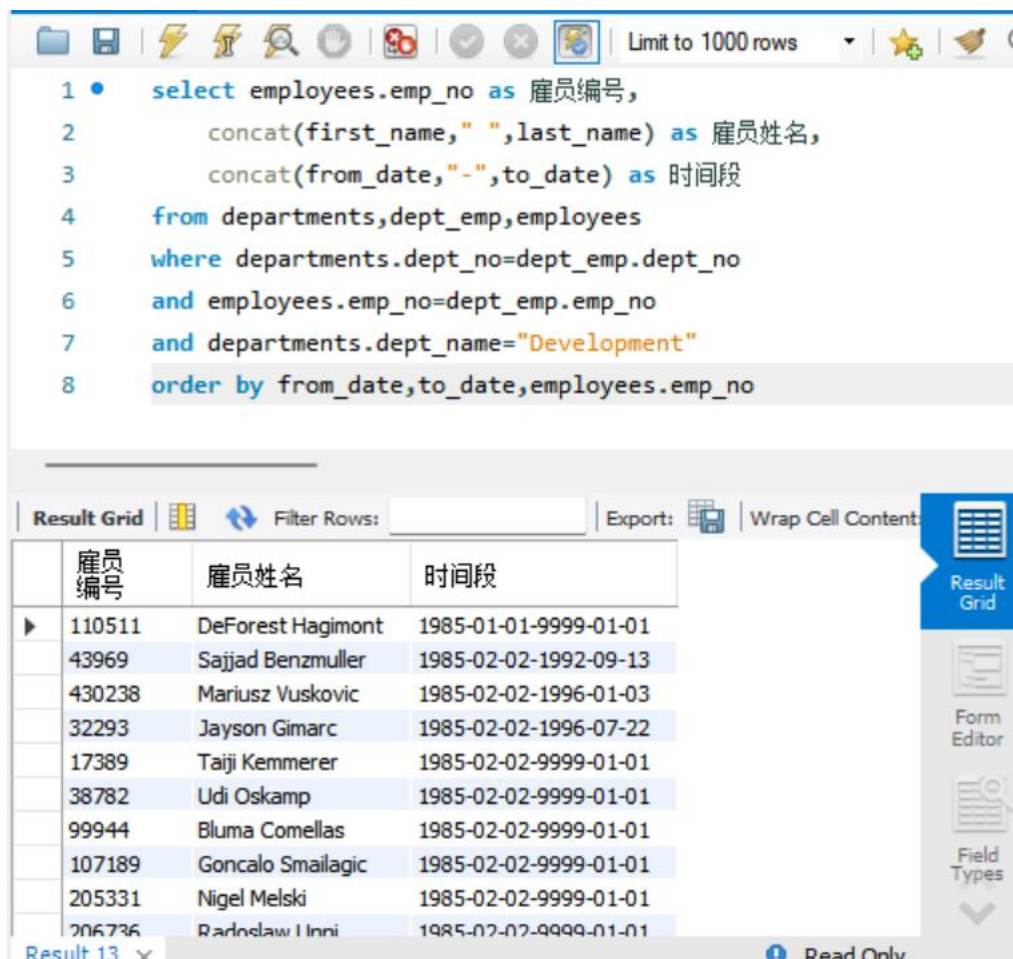
9.12 查询十二

查询所有曾经在 ‘Development’ 工作过雇员的编号 (emp_no), 姓名 (first_name+last_name), 及时间段 (from_date, end_date);

```

select employees.emp_no as 雇员编号,
       concat(first_name," ",last_name) as 雇员姓名,
       concat(from_date,"-",to_date) as 时间段
from departments,dept_emp,employees
where departments.dept_no=dept_emp.dept_no
and employees.emp_no=dept_emp.emp_no
and departments.dept_name="Development"
order by from_date,to_date,employees.emp_no

```



The screenshot shows a database query tool interface. The top part displays the SQL query code, and the bottom part shows the results in a table format.

SQL Query:

```

1 • select employees.emp_no as 雇员编号,
2     concat(first_name," ",last_name) as 雇员姓名,
3     concat(from_date,"-",to_date) as 时间段
4 from departments,dept_emp,employees
5 where departments.dept_no=dept_emp.dept_no
6 and employees.emp_no=dept_emp.emp_no
7 and departments.dept_name="Development"
8 order by from_date,to_date,employees.emp_no

```

Result Grid:

雇员编号	雇员姓名	时间段
110511	DeForest Hagimont	1985-01-01-9999-01-01
43969	Sajjad Benzmulder	1985-02-02-1992-09-13
430238	Mariusz Vuskovic	1985-02-02-1996-01-03
32293	Jayson Gimarc	1985-02-02-1996-07-22
17389	Taiji Kemmerer	1985-02-02-9999-01-01
38782	Udi Oskamp	1985-02-02-9999-01-01
99944	Bluma Comellas	1985-02-02-9999-01-01
107189	Goncalo Smailagic	1985-02-02-9999-01-01
205331	Nigel Melski	1985-02-02-9999-01-01
206736	Radoslaw Inni	1985-02-02-9999-01-01

图 9-12 查询十二的查询代码及结果

9.13 查询十三

查询曾经在所有部门都工作过的雇员的编号 (emp_no), 姓名 (first_name+last_name);

```

select employees.emp_no as 雇员编号,
concat(first_name," ",last_name) as 雇员姓名
from employees
where not exists(
select *
from departments
where not exists(
select dept_no
from dept_emp as de2
where de2.dept_no=departments.dept_no

```

```
and de2.emp_no=employees.emp_no
```

```
)
```

```
)
```

本题实际上实现了关系代数中的除法。在未进行 14 题的插入之前，执行该查询代码输出为空。

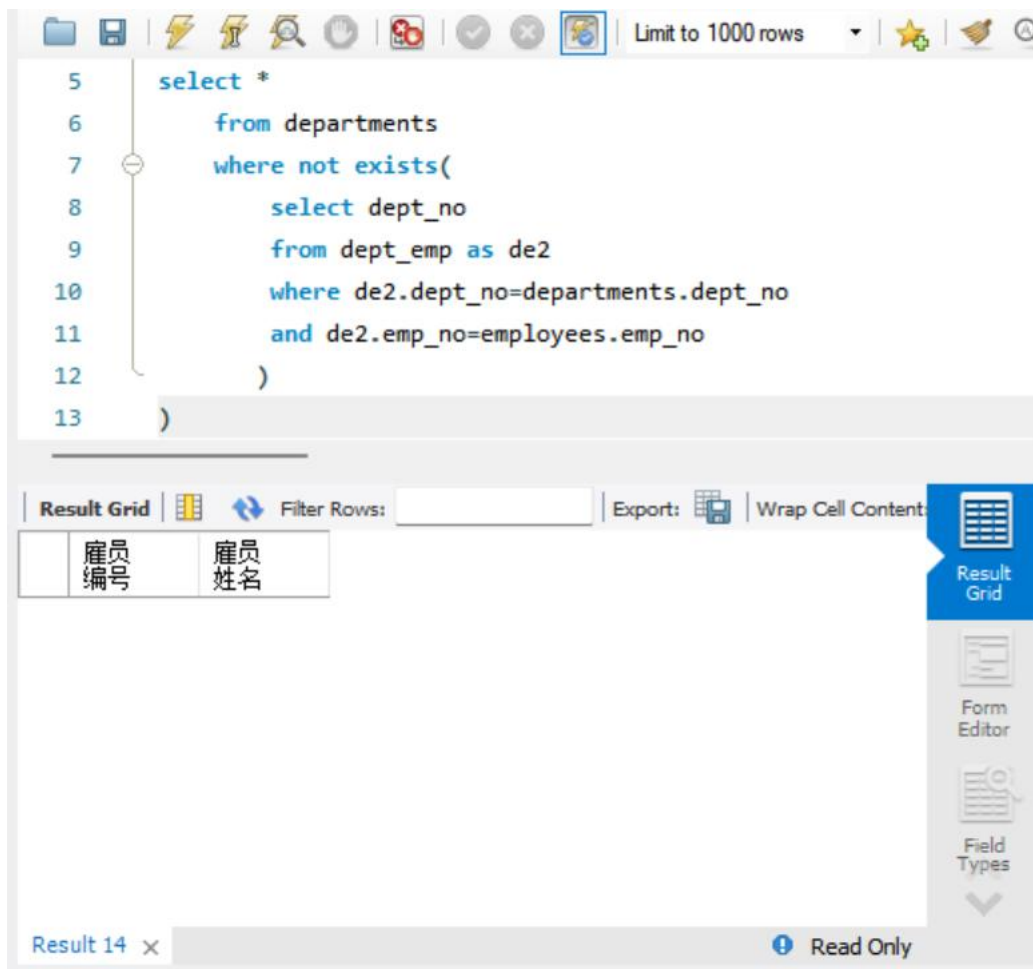


图 9-13 查询十三的查询代码及结果

9.14 插入十四

在 dept_emp 表中插入适当数据使得至少 3 个以上雇员满足上一题的查询要求。

```
insert into employees
```

```
values
```

```
(1000000,'2023-01-01',"TEST1","TEST1",'M','2023-01-01'),
```

```
(1000001,'2022-01-01',"TEST2","TEST2",'F','2022-01-01'),
```

```
(1000002,'2021-01-01',"TEST3","TEST3",'M','2021-01-01');
```

```
insert into dept_emp
values
(1000000,"d001",'2023-01-01','2023-01-01'),
(1000000,"d002",'2023-01-01','2023-01-01'),
(1000000,"d003",'2023-01-01','2023-01-01'),
(1000000,"d004",'2023-01-01','2023-01-01'),
(1000000,"d005",'2023-01-01','2023-01-01'),
(1000000,"d006",'2023-01-01','2023-01-01'),
(1000000,"d007",'2023-01-01','2023-01-01'),
(1000000,"d008",'2023-01-01','2023-01-01'),
(1000000,"d009",'2023-01-01','2023-01-01'),
(1000001,"d001",'2023-01-01','2023-01-01'),
(1000001,"d002",'2023-01-01','2023-01-01'),
(1000001,"d003",'2023-01-01','2023-01-01'),
(1000001,"d004",'2023-01-01','2023-01-01'),
(1000001,"d005",'2023-01-01','2023-01-01'),
(1000001,"d006",'2023-01-01','2023-01-01'),
(1000001,"d007",'2023-01-01','2023-01-01'),
(1000001,"d008",'2023-01-01','2023-01-01'),
(1000001,"d009",'2023-01-01','2023-01-01'),
(1000002,"d001",'2023-01-01','2023-01-01'),
(1000002,"d002",'2023-01-01','2023-01-01'),
(1000002,"d003",'2023-01-01','2023-01-01'),
(1000002,"d004",'2023-01-01','2023-01-01'),
(1000002,"d005",'2023-01-01','2023-01-01'),
(1000002,"d006",'2023-01-01','2023-01-01'),
(1000002,"d007",'2023-01-01','2023-01-01'),
(1000002,"d008",'2023-01-01','2023-01-01'),
(1000002,"d009",'2023-01-01','2023-01-01')
```

在 `employees` 表中插入了三个新雇员，并在 `dept_emp` 表中插入这三个新雇员在所有 9 个部门的任职记录。执行插入后运行 13 题的查询，结果如图 9-14。

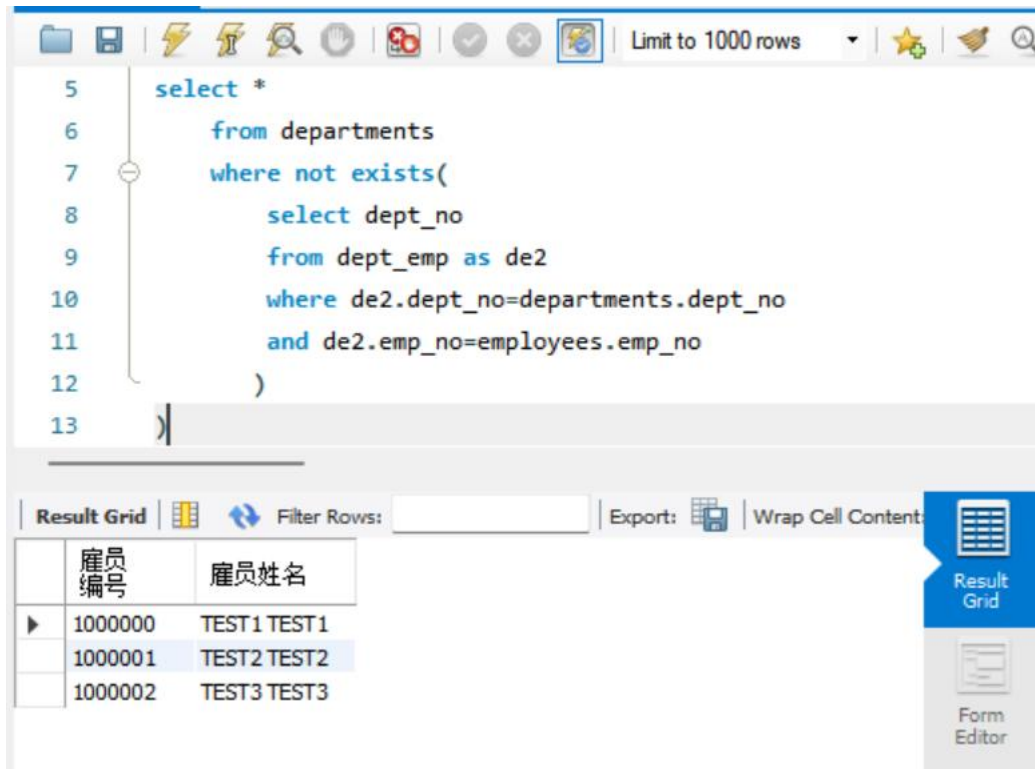


图 9-14 执行插入后查询十三的查询代码及结果

10. 运用 SQL EXPLAIN 进行查询分析

10.1 对于以上查询的分析

10.1.1 查询一分析

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	dept_emp	NULL	index	PRIMARY,emp_no,dept_no	emp_no	4	NULL	331143	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	departments	NULL	eq_ref	PRIMARY,dept_name	PRIMARY	16	employees.dept_emp.dept_no	1	100.00	NULL
1	SIMPLE	salaries	NULL	ref	PRIMARY,emp_no	PRIMARY	4	employees.dept_emp.emp_no	9	100.00	NULL

图 10-1 查询一分析

读取表的顺序：dept_emp, departments, salaries

哪些索引能够被使用：dept_emp 的主键索引、emp_no、dept_no, departments 的主键索引、dept_name, salaries 的主键索引、emp_no

数据读取操作的操作类型：index, eq_ref, ref

哪些索引能够被实际使用：dept_emp.emp_no, departments 的主键索引, salaries 的主键索引

表之间的引用：departments 引用 dept_emp.dept_no, salaries 引用 dept_emp.emp_no

每张表有多少行被物理查询：331170, 1, 9

10.1.2 查询二分析

Result Grid												
Filter Rows:												
Export: Wrap Cell Content:												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	departments	NULL	index	PRIMARY	dept_name	162	NULL	9	100.00	Using index; Using temporary; Using filesort
	1	SIMPLE	dept_manager	NULL	ref	PRIMARY, emp_no, dept_no	dept_no	16	employees.departments.dept_no	2	100.00	NULL
	1	SIMPLE	employees	NULL	eq_ref	PRIMARY	PRIMARY	4	employees.dept_manager.emp_no	1	100.00	NULL

图 10-2 查询二分析

读取表的顺序：departments, dept_manager, employees

哪些索引能够被使用：dept_manager 的主键索引、emp_no、dept_no，departments 的主键索引，employees 的主键索引

数据读取操作的操作类型：index, ref, eq_ref

哪些索引能够被实际使用：departments.dept_name、dept_manager.dept_no 和 employees 的主键索引

表之间的引用：dept_manager 引用 departments.dept_no，employees 引用 dept_manager.emp_no

每张表有多少行被物理查询：9，2，1

10.1.3 查询三分析

Result Grid												
Filter Rows:												
Export: Wrap Cell Content:												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	employees	NULL	ALL	PRIMARY	NULL	NULL	NULL	299243	100.00	Using temporary; Using filesort
	1	SIMPLE	salaries	NULL	ref	PRIMARY, emp_no	PRIMARY	4	employees.employees.emp_no	9	100.00	NULL

图 10-3 查询三分析

读取表的顺序：employees, salaries

哪些索引能够被使用：employees 的主键索引，salaries 的主键索引、emp_no

数据读取操作的操作类型：ALL, ref

哪些索引能够被实际使用：salaries 的主键索引

表之间的引用：salaries 引用 employees.emp_no

每张表有多少行被物理查询：299243，9

10.1.4 查询四分析

Result Grid												
Filter Rows:												
Export: Wrap Cell Content:												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	departments	NULL	index	PRIMARY	dept_name	162	NULL	9	100.00	Using index; Using temporary; Using filesort
	1	SIMPLE	dept_emp	NULL	ref	PRIMARY, emp_no, dept_no	dept_no	16	employees.departments.dept_no	41392	100.00	NULL
	1	SIMPLE	employees	NULL	eq_ref	PRIMARY	PRIMARY	4	employees.dept_emp.emp_no	1	100.00	NULL

图 10-4 查询四分析

读取表的顺序：departments, dept_emp, employees

哪些索引能够被使用：departments 的主键索引，dept_emp 的主键索引、emp_no、dept_no，employees 的主键索引

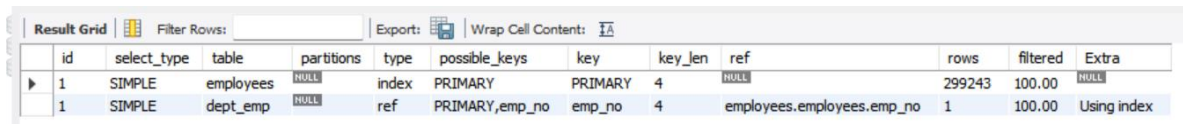
数据读取操作的操作类型：index，ref，eq_ref

哪些索引能够被实际使用：departments.dept_name，dept_emp.dept_no，employees 的主键索引

表之间的引用：dept_emp 引用 departments.dept_no，employees 引用 dept_emp.emp_no

每张表有多少行被物理查询：9，41392，1

10.1.5 查询五分析



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	employees	NULL	index	PRIMARY	PRIMARY	4	NULL	299243	100.00	NULL
1	SIMPLE	dept_emp	NULL	ref	PRIMARY,emp_no	emp_no	4	employees.employees.emp_no	1	100.00	Using index

图 10-5 查询五分析

读取表的顺序：employees，dept_emp

哪些索引能够被使用：dept_emp 的主键索引、emp_no，employees 的主键索引

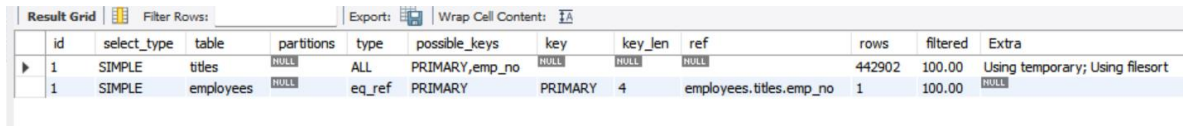
数据读取操作的操作类型：index，ref

哪些索引能够被实际使用：dept_emp 的主键索引，employees 的主键索引

表之间的引用：dept_emp 引用 employees.emp_no

每张表有多少行被物理查询：299243，1

10.1.6 查询六分析



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	titles	NULL	ALL	PRIMARY,emp_no	NULL	NULL	NULL	442902	100.00	Using temporary; Using filesort
1	SIMPLE	employees	NULL	eq_ref	PRIMARY	PRIMARY	4	employees.titles.emp_no	1	100.00	NULL

图 10-6 查询六分析

读取表的顺序：titles，employees

哪些索引能够被使用：titles 的主键索引、emp_no，employees 的主键索引

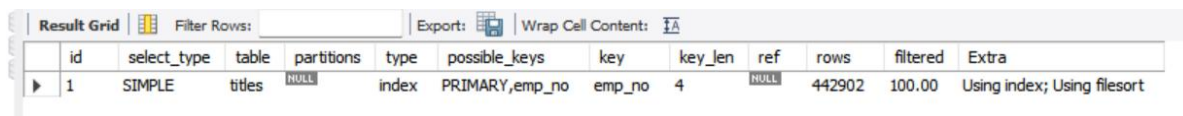
数据读取操作的操作类型：ALL，eq_ref

哪些索引能够被实际使用：employees 的主键索引

表之间的引用：employees 引用 titles.emp_no

每张表有多少行被物理查询：442902，1

10.1.7 查询七分析



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	titles	NULL	index	PRIMARY,emp_no	emp_no	4	NULL	442902	100.00	Using index; Using filesort

图 10-7 查询七分析

读取表的顺序： titles

哪些索引能够被使用： titles 的主键索引、emp_no

数据读取操作的操作类型： index

哪些索引能够被实际使用： titles.emp_no

表之间的引用： 无

每张表有多少行被物理查询： 442902

10.1.8 查询八分析

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	titles	NULL	index	PRIMARY,emp_no	emp_no	4	NULL	441654	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	dept_emp	NULL	ref	PRIMARY,emp_no,dept_no	PRIMARY	4	employees.titles.emp_no	1	100.00	Using index
1	SIMPLE	departments	NULL	eq_ref	PRIMARY	PRIMARY	16	employees.dept_emp.dept_no	1	100.00	NULL

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	departments	NULL	index	PRIMARY	dept_name	162	NULL	9	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	dept_emp	NULL	ref	PRIMARY,emp_no,dept_no	dept_no	16	employees.departments.dept_no	41392	100.00	Using index
1	SIMPLE	titles	NULL	ref	PRIMARY,emp_no	emp_no	4	employees.dept_emp.emp_no	1	100.00	Using index

图 10-8 查询八分析

读取表的顺序： departments, dept_emp, titles

哪些索引能够被使用： departments 的主键索引， dept_emp 的主键索引、emp_no、dept_no， titles 的主键索引、emp_no

数据读取操作的操作类型： index, ref, ref

哪些索引能够被实际使用： departments.dept_name， dept_emp.dept_no， employees.dept_no

表之间的引用： dept_emp 引用 departments.emp_no， titles 引用 dept_emp.dept_no

每张表有多少行被物理查询： 9， 41392， 1

10.1.9 查询九分析

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	employees	NULL	ALL	PRIMARY	NULL	NULL	NULL	299243	100.00	NULL
1	PRIMARY	a	NULL	ref	PRIMARY,emp_no	PRIMARY	4	employees.employees.emp_no	9	100.00	Using where
2	DEPENDENT SUBQUERY	b	NULL	ref	PRIMARY,emp_no	PRIMARY	4	employees.a.emp_no	9	100.00	NULL

图 10-9 查询九分析

读取表的顺序： employees, salaries a, salaries b

哪些索引能够被使用： employees 的主键索引， a、b 的主键索引、emp_no

数据读取操作的操作类型： ALL, ref, ref

哪些索引能够被实际使用： a、b 的主键索引

表之间的引用： a 引用 employees.emp_no， b 引用 a..emp_no

每张表有多少行被物理查询：299243，9，9

10.1.10 查询十分析

Result Grid											
Filter Rows:											
Export:											
Wrap Cell Content:											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	dept_manager	NULL	index	PRIMARY,emp_no	emp_no	4	NULL	24	100.00	Using index
1	SIMPLE	employees	NULL	eq_ref	PRIMARY	PRIMARY	4	employees.dept_manager.emp_no	1	100.00	Using index

图 10-10 查询十分析

读取表的顺序：dept_manager，employees

哪些索引能够被使用：dept_manager 的主键索引、emp_no，employees 的主键索引

数据读取操作的操作类型：index，eq_ref

哪些索引能够被实际使用：dept_manager.emp_no 和 employees 的主键索引

表之间的引用：employees 引用 dept_manager.emp_no

每张表有多少行被物理查询：24，1

10.1.11 查询十一分析

Result Grid											
Filter Rows:											
Export:											
Wrap Cell Content:											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	departments	NULL	index	PRIMARY	dept_name	162	NULL	9	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	dept_manager	NULL	ref	PRIMARY,emp_no,dept_no	dept_no	16	employees.departments.dept_no	2	100.00	Using index
1	SIMPLE	employees	NULL	eq_ref	PRIMARY	PRIMARY	4	employees.dept_manager.emp_no	1	100.00	Using index

图 10-11 查询十一分析

读取表的顺序： departments，dept_manager，employees

哪些索引能够被使用： departments 的主键索引，dept_manager 的主键索引、emp_no、dept_no，employees 的主键索引

数据读取操作的操作类型：index，ref，eq_ref

哪些索引能够被实际使用： departments.dept_name，dept_manager.dept_no，employees 的主键索引

表之间的引用： dept_emp 引用 departments.dept_no，employees 引用 dept_manager.emp_no

每张表有多少行被物理查询：9，2，1

10.1.12 查询十二分析

Result Grid											
Filter Rows:											
Export:											
Wrap Cell Content:											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	departments	NULL	const	PRIMARY,dept_name	dept_name	162	const	1	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	dept_emp	NULL	ref	PRIMARY,emp_no,dept_no	dept_no	16	const	148054	100.00	Using index
1	SIMPLE	employees	NULL	eq_ref	PRIMARY	PRIMARY	4	employees.dept_emp.emp_no	1	100.00	Using index

图 10-12 查询十二分析

读取表的顺序： departments，dept_emp，employees

哪些索引能够被使用： departments 的主键索引、dept_name, dept_emp 的主键索引、emp_no、dept_no, employees 的主键索引

数据读取操作的操作类型：const, ref, eq_ref

哪些索引能够被实际使用： departments.dept_name, dept_emp.dept_no, employees 的主键索引

表之间的引用： employees 引用 dept_emp.emp_no

每张表有多少行被物理查询： 1, 148054, 1

10.1.13 查询十三分析

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	employees	NULL	ALL	NULL	NULL	NULL	NULL	299243	100.00	Using where
2	DEPENDENT SUBQUERY	departments	NULL	index	NULL	dept_name	162	NULL	9	100.00	Using index
3	DEPENDENT SUBQUERY	<subquery3>	NULL	eq_ref	<auto_distinct_key>	<auto_distinct_key>	22	employees.departments.dept_no,employees.e...	1	100.00	Using where; Not exists
3	MATERIALIZED	de2	NULL	ref	PRIMARY,emp_no,dept_no	emp_no	4	employees.employees.emp_no	1	100.00	Using index

图 10-13 查询十三分析

读取表的顺序： employees, departments, <subquery3>, dept_emp de2

哪些索引能够被使用： <subquery3>.<auto_distinct_key>, de2 的主键索引、emp_no、dept_no

数据读取操作的操作类型：ALL, index, eq_ref, ref

哪些索引能够被实际使用： departments.dept_name, <subquery3>.<auto_distinct_key>, employees 的主键索引

表之间的引用： <subquery3>引用 departments.dept_no 和 employees. emp_no, de2 引用 employees. emp_no

每张表有多少行被物理查询： 299243, 9, 1, 1

10.2 索引与查询优化

使用 SQL EXPLAIN 工具对下列 SQL 查询语句进行分析，判断哪些索引将对下列查询产生加速作用。

```
SELECT TITLE,DEPT_NAME,GENDER, COUNT(DISTINCT E.EMP_NO)
FROM DEPARTMENTS D,DEPT_EMP DE,EMPLOYEES E,SALARIES
S,TITLES T
WHERE D.DEPT_NO=DE.DEPT_NO AND DE.EMP_NO=E.EMP_NO AND
E.EMP_NO=S.EMP_NO AND E.EMP_NO=T.EMP_NO
GROUP BY TITLE,DEPT_NAME,GENDER
ORDER BY DEPT_NAME;
```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
	1	SIMPLE	T	NULL	index	PRIMARY,emp_no	emp_no	4	NULL	441654	100.00	Using index; Using temporary; Using filesort
	1	SIMPLE	E	NULL	eq_ref	PRIMARY	PRIMARY	4	employees.T.emp_no	1	100.00	NULL
	1	SIMPLE	DE	NULL	ref	PRIMARY,emp_no,dept_no	PRIMARY	4	employees.T.emp_no	1	100.00	Using index
	1	SIMPLE	D	NULL	eq_ref	PRIMARY	PRIMARY	16	employees.DE.dept_no	1	100.00	NULL
	1	SIMPLE	S	NULL	ref	PRIMARY,emp_no	PRIMARY	4	employees.T.emp_no	9	100.00	Using index

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
	1	SIMPLE	D	NULL	index	PRIMARY	dept_name	162	NULL	9	100.00	Using index; Using temporary; Using filesort
	1	SIMPLE	DE	NULL	ref	PRIMARY,emp_no,dept_no	dept_no	16	employees.D.dept_no	41392	100.00	Using index
	1	SIMPLE	E	NULL	eq_ref	PRIMARY	PRIMARY	4	employees.DE.emp_no	1	100.00	NULL
	1	SIMPLE	T	NULL	ref	PRIMARY,emp_no	emp_no	4	employees.DE.emp_no	1	100.00	Using index
	1	SIMPLE	S	NULL	ref	PRIMARY,emp_no	PRIMARY	4	employees.DE.emp_no	9	100.00	Using index

图 10-14 explain 分析结果

能够被实际使用的索引有：T.emp_no,E 的主键索引,DE.dept_no,D.dept_name,S 的主键索引。

对于 dept_name, 删去其索引：drop index dept_name on departments

执行查询，耗时 20.969sec。

31	09:26:44	drop index dept_name on departments	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
32	09:26:58	SELECT TITLE,DEPT_NAME,GENDER, COUNT(DISTINCT E.EMP_NO) FROM DEPARTMENTS D,DEPT_EMP DE,EMPLOYEES E,SALARIES... 87 row(s) returned	20.969 sec / 0.000 sec	

图 10-15 无索引查询时间

添加索引：create index dept_name on departments(dept_name)

执行查询，耗时 14.438sec。

33	09:28:02	create index dept_name on departments(dept_name)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
34	09:28:05	SELECT TITLE,DEPT_NAME,GENDER, COUNT(DISTINCT E.EMP_NO) FROM DEPARTMENTS D,DEPT_EMP DE,EMPLOYEES E,SALARIES... 87 row(s) returned	14.438 sec / 0.000 sec	

图 10-16 有索引查询时间

可见索引对查询具有优化作用。

11. SQL 查询实验总结

通过本次实验，我深入实践了 SQL 查询语句的编写与运用，掌握了如何利用‘EXPLAIN’工具输出查询语句的执行计划，并对其进行了细致分析。在此基础上，我进一步借助‘EXPLAIN’语句等工具，对索引及查询进行了优化分析。整个实验过程极大地提升了我运用 SQL 进行高效数据查询与分析的能力，使我能够更加熟练地处理各类数据查询任务。