

软件学院

数据库实验：公交线路查询系统数据库设计

|  |  |
| --- | --- |
| 选课编号： |  |
| 姓 名： |  |
| 学 号： |  |
| 班 级： |  |
| 指导老师： |  |
| 完成日期： |  |

**目 录**

[1. 需求分析 1](#_Toc137141660)

[2. 概念结构设计 1](#_Toc137141661)

[3. 逻辑结构设计 2](#_Toc137141662)

[3.1建立关系模式 2](#_Toc137141663)

[3.2 关系模式规范化处理 2](#_Toc137141664)

[3.3 关系模式逻辑结构定义 3](#_Toc137141665)

[4. 数据库物理设计 5](#_Toc137141666)

[5. 数据库实施与测试 8](#_Toc137141667)

[5.1 数据入库 8](#_Toc137141668)

[5.2 建立视图 10](#_Toc137141669)

[5.3 增删改查测试 12](#_Toc137141670)

[5.3.1 插入测试 12](#_Toc137141671)

[5.3.2 修改测试 12](#_Toc137141672)

[5.3.3 删除测试 14](#_Toc137141673)

[5.3.4 查询测试 14](#_Toc137141674)

[5.4 建立并测试触发器 16](#_Toc137141675)

[6. 数据库建模实验总结 18](#_Toc137141676)

[7. 样本数据库的安装和数据加载 18](#_Toc137141677)

[8. 样本数据库的分析 19](#_Toc137141678)

[9. 样本数据库上的查询 19](#_Toc137141679)

[9.1 查询一 19](#_Toc137141680)

[9.2 查询二 20](#_Toc137141681)

[9.3 查询三 21](#_Toc137141682)

[9.4 查询四 22](#_Toc137141683)

[9.5 查询五 23](#_Toc137141684)

[9.6 查询六 24](#_Toc137141685)

[9.7 查询七 25](#_Toc137141686)

[9.8 查询八 26](#_Toc137141687)

[9.9 查询九 26](#_Toc137141688)

[9.10 查询十 28](#_Toc137141689)

[9.11 查询十一 28](#_Toc137141690)

[9.12 查询十二 29](#_Toc137141691)

[9.13 查询十三 30](#_Toc137141692)

[9.14 插入十四 31](#_Toc137141693)

[10. 运用SQL EXPLAIN进行查询分析 33](#_Toc137141694)

[10.1 对于以上查询的分析 33](#_Toc137141695)

[10.1.1 查询一分析 33](#_Toc137141696)

[10.1.2 查询二分析 33](#_Toc137141697)

[10.1.3 查询三分析 33](#_Toc137141698)

[10.1.4 查询四分析 34](#_Toc137141699)

[10.1.5 查询五分析 34](#_Toc137141700)

[10.1.6 查询六分析 34](#_Toc137141701)

[10.1.7 查询七分析 35](#_Toc137141702)

[10.1.8 查询八分析 35](#_Toc137141703)

[10.1.9 查询九分析 36](#_Toc137141704)

[10.1.10 查询十分析 36](#_Toc137141705)

[10.1.11 查询十一分析 36](#_Toc137141706)

[10.1.12 查询十二分析 37](#_Toc137141707)

[10.1.13 查询十三分析 37](#_Toc137141708)

[10.2 索引与查询优化 37](#_Toc137141709)

[11. SQL查询实验总结 38](#_Toc137141710)

[12. 实验中遇到的问题和问题解决 38](#_Toc137141711)

[12.1 插入嵌套查询时，Error code:1093 38](#_Toc137141712)

[12.2 插入嵌套查询时，Error code:1248 39](#_Toc137141713)

[12.3 修改时，Error code:1175 39](#_Toc137141714)

# 需求分析

公交线路查询系统用于存储一个城市的公交数据，存储这个城市内的公交线路信息、公交站点信息、公交线路的停靠信息和公交站点附近的单位等。对于公交线路，需要存储其编号、名称、运行里程、首末班车发车时间、高峰及平时发车间隔和首末站点；对于公交站点，需要存储其编号、名称、地理坐标；对于沿线单位，需要存储其编号、名称、电话、地理坐标。公交线路与公交站点存在停靠关系，需要记录公交线路停靠站点的先后顺序、相邻站点的运行距离和平均运行时间；沿线单位与公交站点存在临近关系，需要记录沿线单位与临近站点的距离。

# 概念结构设计

根据以上需求建立E-R图。公交线路和公交站点的首末站点关系是n:1的关系；公交线路和公交站点的停靠关系是m:n的关系；沿线单位和公交站点的临近关系是n:1的关系。

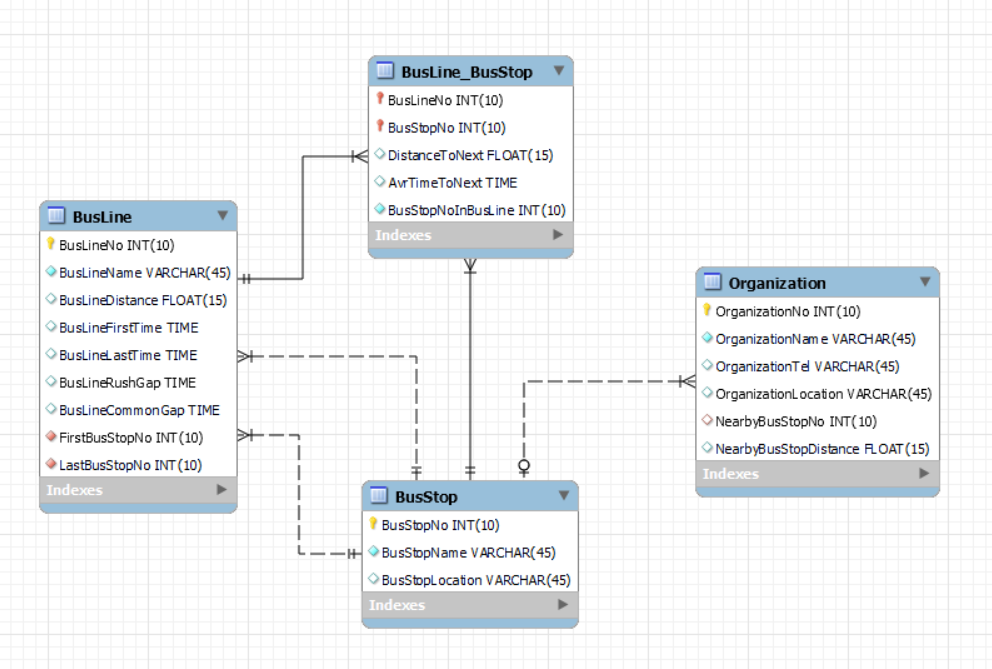


图2-1 公交线路查询系统ER图

# 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。

对于公交查询系统，根据其E-R图可建立如下几个关系模式：

公交线路（线路编号，线路名称，运行里程，首班车发车时间，末班车发车时间，高峰发车间隔，平时发车间隔，首站点，末站点）

公交站点（站点编号，站点名称，地理坐标）

沿线单位（单位编号，单位名称，联系电话，地理坐标，邻近站点，邻近站点距离）

公交线路-公交站点（线路编号，站点编号，站点在线路内编号，至下站距离，至下站平均时间）

其中，公交线路关系模式中的外键是首站点、末战点，沿线单位关系模式中的外键是邻近站点，公交线路-公交站点关系模式中的外键是线路编号、站点编号。

## 3.2 关系模式规范化处理

公交线路关系模式中，线路编号和线路名称是主属性，且它们相互完全函数依赖。线路距离、运行里程、首班车发车时间、末班车发车时间、高峰发车间隔、平时发车间隔、首站点、末战点均完全函数依赖于线路编号或线路名称，该关系模式是3NF。

公交站点关系模式中，站点编号、站点名称和地理坐标均能唯一确定一个元组，因此所有属性都是主属性，该关系模式是3NF。

沿线单位关系模式中，单位编号和单位名称是主属性，联系电话，地理坐标，邻近站点，邻近站点距离均完全函数依赖于单位编号或单位名称，该关系模式是3NF。

公交线路-公交站点关系模式中，线路编号、站点编号和站点在线路内编号是主属性，至下站距离、至下站平均时间完全函数依赖于候选键（线路编号,站点编号）或（线路编号,站点在线路内编号），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公交线路关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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** |
| BusLineNo | 线路编号 | INT | 10 | 是 |  | Not null  Unsigned |
| BusLineName | 线路名称 | VARCHAR | 45 |  |  | Not null |
| BusLineDistance | 运行里程 | FLOAT | 15 |  |  | Unsigned |
| BusLineFirstTime | 首班发车时间 | TIME |  |  |  | null |
| BusLineLastTime | 末班发车时间 | TIME |  |  |  | null |
| BusLineRushGap | 高峰发车间隔 | TIME |  |  |  | null |
| BusLineCommonGap | 平时发车间隔 | TIME |  |  |  | null |
| FirstBusStopNo | 首站编号 | INT | 10 |  | 是 | Not null  Unsigned |
| LastBusStopNo | 末站编号 | INT | 10 |  | 是 | Not null  Unsigned |

表3-2公交站点关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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** |
| BusStopNo | 站点编号 | INT | 10 | 是 |  | Not null  Unsigned |
| BusStopName | 站点名称 | VARCHAR | 45 |  |  | Not null |
| BusStopLocation | 站点位置 | VARCHAR | 45 |  |  | null |

表3-3沿线单位关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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** |
| OrganizationNo | 单位编号 | INT | 10 | 是 |  | Not null  Unsigned |
| OrganizationName | 单位名称 | VARCHAR | 45 |  |  | Not null |
| OrganizationTel | 单位电话 | VARCHAR | 45 |  |  | null |
| OrganizationLocation | 单位位置 | VARCHAR | 45 |  |  | null |
| NearbyBusStopNo | 邻近站点编号 | INT | 10 |  | 是 | Unsigned |
| NearBusStopDistance | 邻近站点距离 | FLOAT | 15 |  |  | Unsigned |

表3-4公交线路-公交站点关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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** |
| BusLineNo | 线路编号 | INT | 10 | 是 | 是 | Not null  Unsigned |
| BusStopNo | 站点编号 | INT | 10 | 是 | 是 | Not null  Unsigned |
| DistanceToNext | 至线路下站距离 | FLOAT | 15 |  |  | null |
| AvrTimeToNext | 至线路下站时间 | TIME |  |  |  | null |
| BusStopNoInBusLine | 站点在线路内编号 | INT | 10 |  |  | Not null  Unsigned |

# 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\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE SCHEMA IF NOT EXISTS `bus\_manage` DEFAULT CHARACTER SET utf8 ;

USE `bus\_manage` ;

CREATE TABLE IF NOT EXISTS `bus\_manage`.`BusStop` (

`BusStopNo` INT(10) UNSIGNED NOT NULL,

`BusStopName` VARCHAR(45) NOT NULL,

`BusStopLocation` VARCHAR(45) NULL,

PRIMARY KEY (`BusStopNo`))

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `bus\_manage`.`BusLine` (

`BusLineNo` INT(10) UNSIGNED NOT NULL,

`BusLineName` VARCHAR(45) NOT NULL,

`BusLineDistance` FLOAT(15) UNSIGNED NULL,

`BusLineFirstTime` TIME NULL,

`BusLineLastTime` TIME NULL,

`BusLineRushGap` TIME NULL,

`BusLineCommonGap` TIME NULL,

`FirstBusStopNo` INT(10) UNSIGNED NOT NULL,

`LastBusStopNo` INT(10) UNSIGNED NOT NULL,

PRIMARY KEY (`BusLineNo`),

INDEX `fk\_BusLine\_BusStop1\_idx` (`FirstBusStopNo` ASC) VISIBLE,

INDEX `fk\_BusLine\_BusStop2\_idx` (`LastBusStopNo` ASC) VISIBLE,

CONSTRAINT `fk\_BusLine\_BusStop1`

FOREIGN KEY (`FirstBusStopNo`)

REFERENCES `bus\_manage`.`BusStop` (`BusStopNo`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_BusLine\_BusStop2`

FOREIGN KEY (`LastBusStopNo`)

REFERENCES `bus\_manage`.`BusStop` (`BusStopNo`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `bus\_manage`.`Organization` (

`OrganizationNo` INT(10) UNSIGNED NOT NULL,

`OrganizationName` VARCHAR(45) NOT NULL,

`OrganizationTel` VARCHAR(45) NULL,

`OrganizationLocation` VARCHAR(45) NULL,

`NearbyBusStopNo` INT(10) UNSIGNED NULL,

`NearbyBusStopDistance` FLOAT(15) UNSIGNED NULL,

PRIMARY KEY (`OrganizationNo`),

INDEX `fk\_Organization\_BusStop1\_idx` (`NearbyBusStopNo` ASC) VISIBLE,

CONSTRAINT `fk\_Organization\_BusStop1`

FOREIGN KEY (`NearbyBusStopNo`)

REFERENCES `bus\_manage`.`BusStop` (`BusStopNo`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `bus\_manage`.`BusLine\_BusStop` (

`BusLineNo` INT(10) UNSIGNED NOT NULL,

`BusStopNo` INT(10) UNSIGNED NOT NULL,

`DistanceToNext` FLOAT(15) NULL,

`AvrTimeToNext` TIME NULL,

`BusStopNoInBusLine` INT(10) UNSIGNED NOT NULL,

PRIMARY KEY (`BusLineNo`, `BusStopNo`),

INDEX `fk\_BusLine\_has\_BusStop\_BusStop1\_idx` (`BusStopNo` ASC) VISIBLE,

INDEX `fk\_BusLine\_has\_BusStop\_BusLine\_idx` (`BusLineNo` ASC) VISIBLE,

CONSTRAINT `fk\_BusLine\_has\_BusStop\_BusLine`

FOREIGN KEY (`BusLineNo`)

REFERENCES `bus\_manage`.`BusLine` (`BusLineNo`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_BusLine\_has\_BusStop\_BusStop1`

FOREIGN KEY (`BusStopNo`)

REFERENCES `bus\_manage`.`BusStop` (`BusStopNo`)

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：

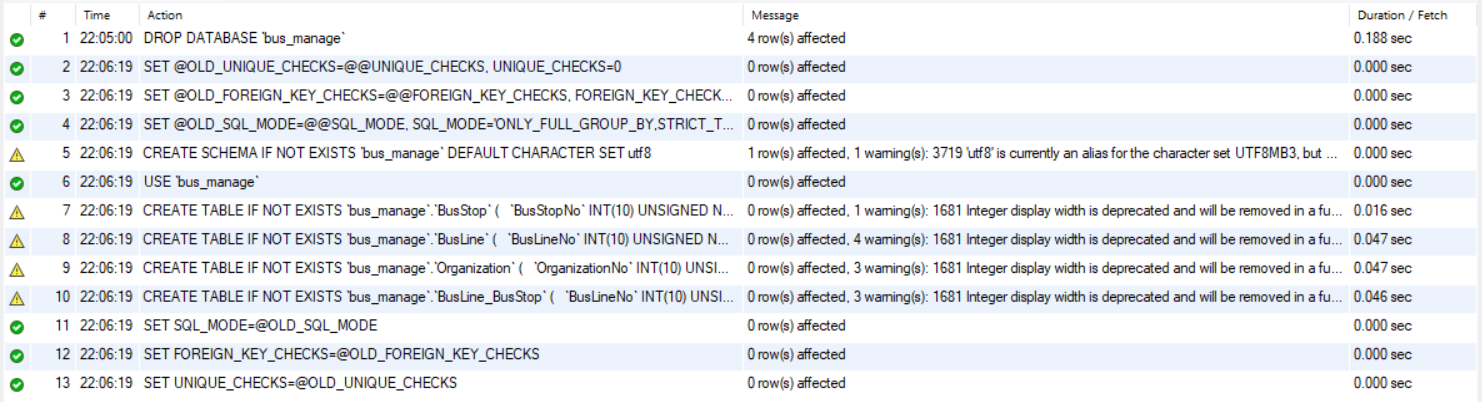


图4-1 数据库建立结果

# 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 `bus\_manage`.`busstop`

VALUES

('1', '沙坡', '二环南路兴庆路西'),

('2', '交大南门', '友谊东路金水路西'),

('3', '兴庆宫公园南门', '咸宁西路兴庆宫公园南门'),

('4', '交大一村西门', '西安交通大学彩虹桥下'),

('5', '乐居场', '兴庆西路西安交通大学西门'),

('6', '市九院', '兴庆西路二环南路西'),

('7', '武警医院', '二环南路金水路'),

('8', '铁一中', '友谊路太乙路东'),

('9', '建东街东段', '建东街太乙路西'),

('10', '九八六医院', '友谊路安西街东'),

('11', '建筑科技大学西门', '雁塔北路测绘路南'),

('12', '大雁塔北广场', '小寨东路雁塔西路西'),

('13', '西安科技大学', '二环南路雁塔北路西'),

('14', '长安大学', '二环南路文艺南路东'),

('15', '文艺路地铁站', '友谊路文艺路东')

INSERT INTO `bus\_manage`.`busline`

VALUES

(1,'交大1路',3.33,'07:00:00','21:30:00','00:10:00','00:20:00',1,3),

(2,'建大2路',2.51,'06:00:00','22:30:00','00:05:00','00:10:00',1,11),

(3,'科大3路',4.95,'08:00:00','20:10:00','00:15:00','00:25:00',2,13),

(4,'长大4路',5.13,'07:00:00','21:00:00','00:05:00','00:12:30',5,14),

(5,'雁塔5路',6.0,'06:00:00','00:00:00','00:07:00','00:17:00',4,12)

INSERT INTO `bus\_manage`.`organization`

VALUES

(1,'西安交大南洋大酒店','029-87665566','二环南路兴庆路东北',1,0.3),

(2,'西安交大思源活动中心','029-82668110','西安交通大学兴庆校区内',2,0.25),

(3,'大雁塔旅游景区','029-85521890','雁塔南路附近',12,1.1),

(4,'长安大学南校区','029-82334813','南二环中段文艺路',14,0.5),

(5,'西安建筑科技大学雁塔校区','029-82202247','友谊路雁塔西路',11,0.1),

(6,'华清广场','029-85306333','友谊路雁塔西路东南',11,0.4),

(7,'西安市铁一中学','13001230456','友谊路太乙路东南',8,0.15),

(8,'西安市第九医院','15771979120','南二环东段151号',6,0.15),

(9,'武警陕西省总队医院','17006540321','南二环东段85号',7,0.2),

(10,'兴庆宫公园','029-82485372','咸宁西路55号',3,0.05)

INSERT INTO `bus\_manage`.`busline\_busstop`

VALUES

(1,1,0.5,'00:02:46',1),

(1,2,0.3,'00:01:56',2),

(1,5,0.7,'00:04:16',3),

(1,3,0,'00:00:00',4),

(2,1,0.5,'00:02:46',1),

(2,2,0.6,'00:05:11',2),

(2,9,1.0,'00:05:27',3),

(2,8,0.7,'00:04:46',4),

(2,11,0,'00:02:46',5),

(3,2,1.5,'00:08:46',1),

(3,5,0.6,'00:05:11',2),

(3,10,1.2,'00:06:27',3),

(3,6,0.4,'00:02:46',4),

(3,7,1.1,'00:01:11',5),

(3,13,0,'00:00:00',6),

(4,5,1.1,'00:05:06',1),

(4,11,0.7,'00:04:11',2),

(4,13,1.7,'00:08:27',3),

(4,10,1.4,'00:06:00',4),

(4,14,0,'00:00:00',5),

(5,4,0.3,'00:01:06',1),

(5,3,0.4,'00:01:11',2),

(5,5,1.9,'00:10:17',3),

(5,10,1.4,'00:06:00',4),

(5,13,1.5,'00:08:30',5),

(5,12,0,'00:00:00',6)

## 5.2 建立视图

1. 建立视图bus，供用户查看每条线路和线路经过的站点。

create view bus

as

select buslinename,busstopname,busstopnoinbusline

from busline,busstop,busline\_busstop

where busline.BusLineNo=busline\_busstop.BusLineNo

and busstop.BusStopNo=busline\_busstop.BusStopNo

在视图bus上执行查询：

select \*

from bus

order by buslinename,busstopnoinbusline

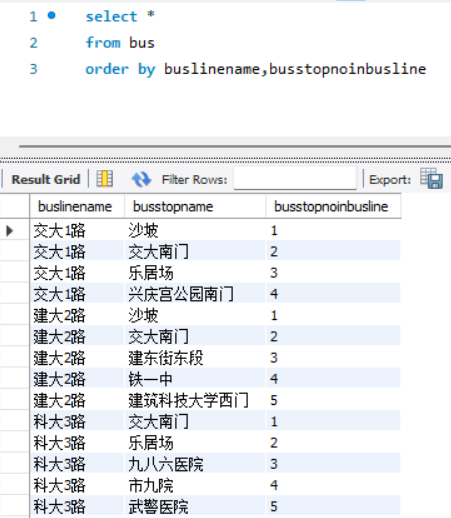


图5-1 视图一的查询测试

在视图bus上修改用户权限：

grant all privileges

on bus

to abc

查看视图的定义信息：

show create view bus



图5-2 查看视图一的定义信息

删除视图：

drop view if exists bus

1. 建立视图nearbybus，供用户查看各个单位的邻近站点及位置。

create view nearbybus

as

select organizationname as 单位名称,

busstopname as 邻近站点,

busstoplocation as 站点位置

from busstop,organization

where organization.NearbyBusStopNo=busstop.BusStopNo

在视图bus上执行查询：

select \*

from nearbybus



图5-3 视图二上的查询一

在视图上查询站点“建筑科技大学西门”的邻近单位：

select 单位名称

from nearbybus

where 邻近站点='建筑科技大学西门'

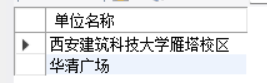


图5-4 视图二上的查询二

## 5.3 增删改查测试

### 5.3.1 插入测试

1. 插入一个新单位“西安科技大学”，其邻近站点是“西安科技大学”。

insert into organization

values

((select \* from (select max(organizationno)+1 from organization) as a),

'西安科技大学','029-85523032','雁塔中路58号',

(select busstopno from busstop where BusStopName='西安科技大学'),

0.3)

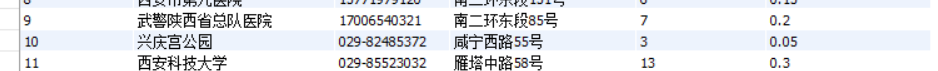


图5-5 插入测试一

1. 插入一个新公交站点“兰蒂斯城”。

insert into busstop

values

((select \* from (select max(busstopno)+1 from busstop) as a),

'兰蒂斯城','交大街北沙坡路')

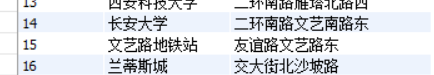


图5-6 插入测试二

### 5.3.2 修改测试

1. 修改公交线路“交大1路”，使之在末站之前增加一站“兰蒂斯城”。

update busline\_busstop

set busstopno=(select busstopno from busstop where busstopname='兰蒂斯城'),

distancetonext=1.0,

avrtimetonext='00:03:00'

where buslineno=(select buslineno from busline where buslinename='交大1路')

and busstopno=(select lastbusstopno from busline where buslinename='交大1路');

insert busline\_busstop

values

((select buslineno from busline where buslinename='交大1路'),

(select lastbusstopno from busline where buslinename='交大1路'),

0,'00:00:00',

(select \* from

(select max(busstopnoinbusline)+1 from busline\_busstop

where buslineno=(select buslineno from busline where buslinename='交大1路'))

as a));

以上sql语句修改了原来存储的最后一站信息，并添加了新的最后一站信息，修改结果如图。第四站是经过修改的信息，第五站是添加的信息。

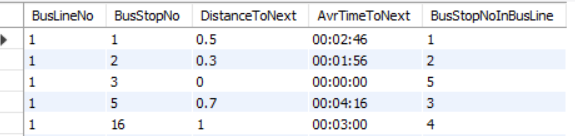


图5-7 修改测试一

1. 修改“兴庆宫公园南门”的公交站地址为“西安交通大学北门”。

update busstop

set busstoplocation='西安交通大学北门'

where busstopno=(select \* from(select busstopno from busstop where busstopname='兴庆宫公园南门') as a)



图5-8 修改测试二

### 5.3.3 删除测试

1. 删除公交站“文艺路地铁站”。



图5-9 删除测试一删除前

delete from busstop

where BusStopName='文艺路地铁站'

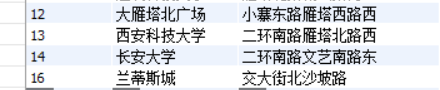


图5-10 删除测试一删除后

1. 删除单位“西安交大思源活动中心”。



图5-11 删除测试二删除前

delete from organization

where OrganizationName='西安交大思源活动中心'

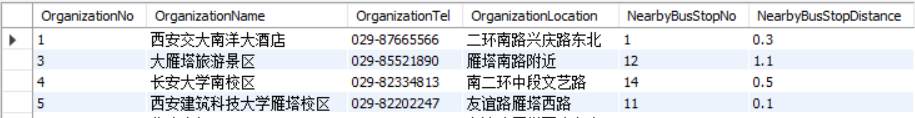


图5-12 删除测试二删除后

### 5.3.4 查询测试

1. 查询公交线路“交大1路”途经站点附近的所有沿线单位及其电话、地址。

select busstopname as 站点名称,

organizationname as 单位名称,

organizationtel as 单位电话,

organizationlocation as 单位位置

from busstop,busline\_busstop,busline,organization

where busline.BusLineName='交大1路'

and busline\_busstop.busstopno=busstop.BusStopNo

and busline\_busstop.buslineno=busline.BusLineNo

and organization.NearbyBusStopNo=busstop.BusStopNo

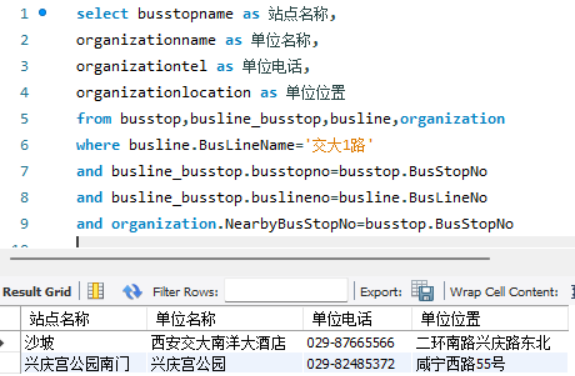


图5-13 查询测试一

1. 查询所有的公交线路及其线路长度、首末班车时间、发车间隔及途径站点数量。

select buslinename as 线路名称,

buslinedistance as 线路长度,

buslinefirsttime as 首班车时间,

buslinelasttime as 末班车时间,

buslinerushgap as 高峰期发车间隔,

buslinecommongap as 平时发车间隔,

count(\*) as 途径站点数量

from busline,busline\_busstop

where busline.buslineno=busline\_busstop.buslineno

group by busline.buslineno

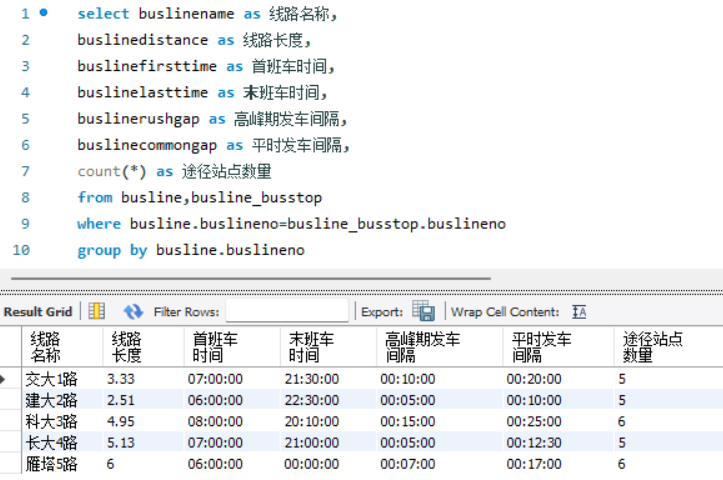


图5-14 查询测试二

## 5.4 建立并测试触发器

1. 建立一个触发器，当删除公交线路时，将公交线路的停靠信息一并删除。

delimiter //

create trigger deletetrigger

before delete on busline

for each row

begin

delete from busline\_busstop where busline\_busstop.buslineno=OLD.buslineno;

end

// delimiter ;

建立成功后，删除一条公交线路：

delete from busline

where buslinename='交大1路'

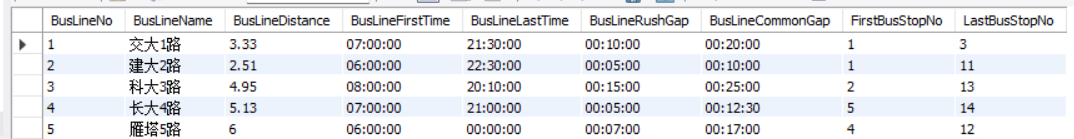


图5-15 触发器测试一删除前

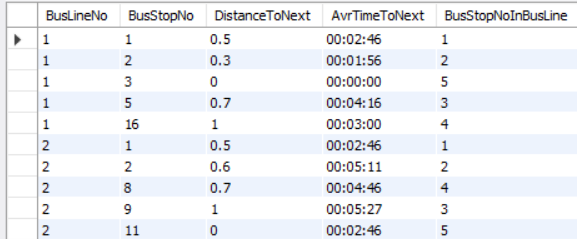


图5-16 触发器测试一删除前

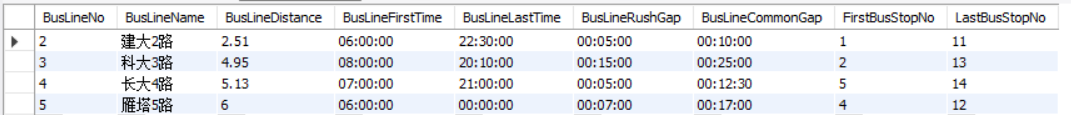


图5-17 触发器测试一删除后

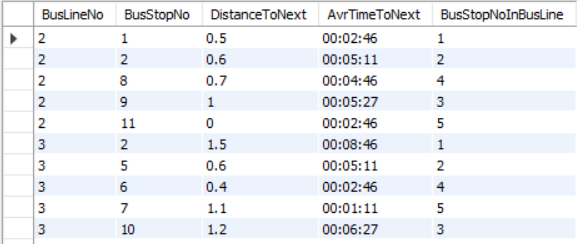


图5-18 触发器测试一删除后

可以看到公交线路的停靠信息被一并删除了。

1. 建立一个触发器，当插入公交线路停靠信息时，更新其末站。

delimiter //

create trigger inserttrigger

after insert on busline\_busstop

for each row

begin

update busline

set lastbusstopno=(select busstopno from busline\_busstop a

where a.buslineno=NEW.buslineno

and a.busstopnoinbusline=(select max(busstopnoinbusline) from busline\_busstop b

where b.buslineno=a.buslineno))

where buslineno=NEW.buslineno;

end

// delimiter ;

插入前的数据表如图5-17，5-18所示。插入一条停靠信息后：

insert into busline\_busstop(buslineno,busstopno,busstopnoinbusline)

value(2,10,6)

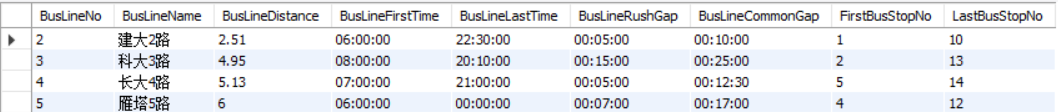


图5-19 触发器测试二插入后

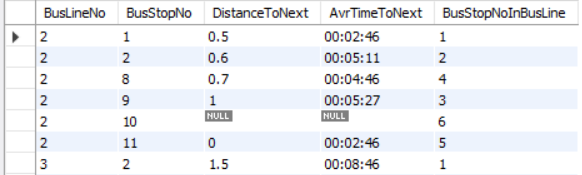


图5-20 触发器测试二插入后

可以看到公交线路中的末站点随之产生了变化。

# 数据库建模实验总结

实验要求的所有内容均已完成，且建立的数据库经过增删查改和触发器测试均未出现异常。实验练习了设计数据库的能力，包括需求分析、E-R图的绘制、逻辑结构设计、建立数据库等。实验还练习了使用SQL语句进行增删查改、建立触发器的能力。通过实验，我对数据库系统和SQL语言形成进一步认识。

# 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。

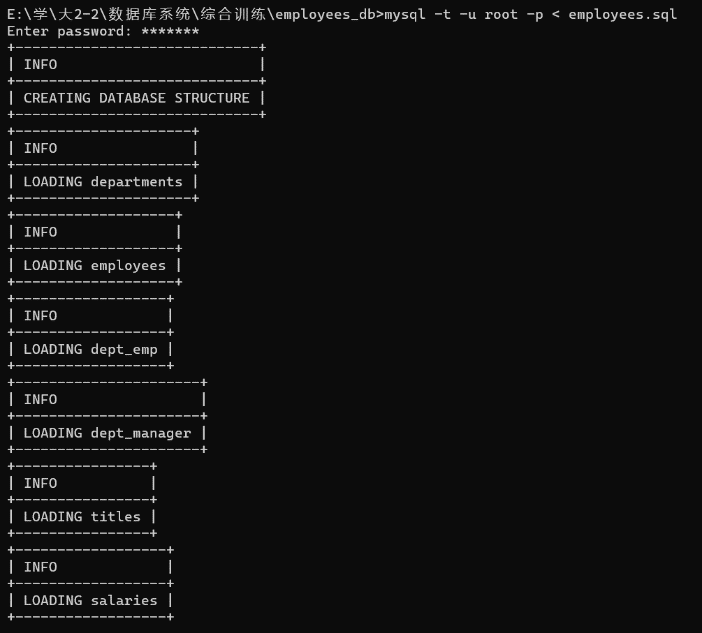


图7-1 样本数据库的加载

# 样本数据库的分析

根据样本数据库的结构绘制E-R图。

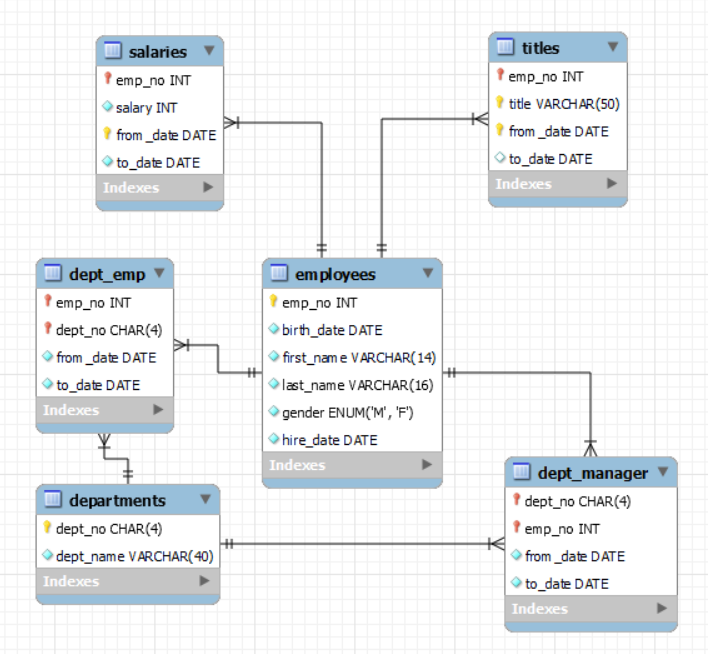


图8-1 样本数据库的E-R图

# 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

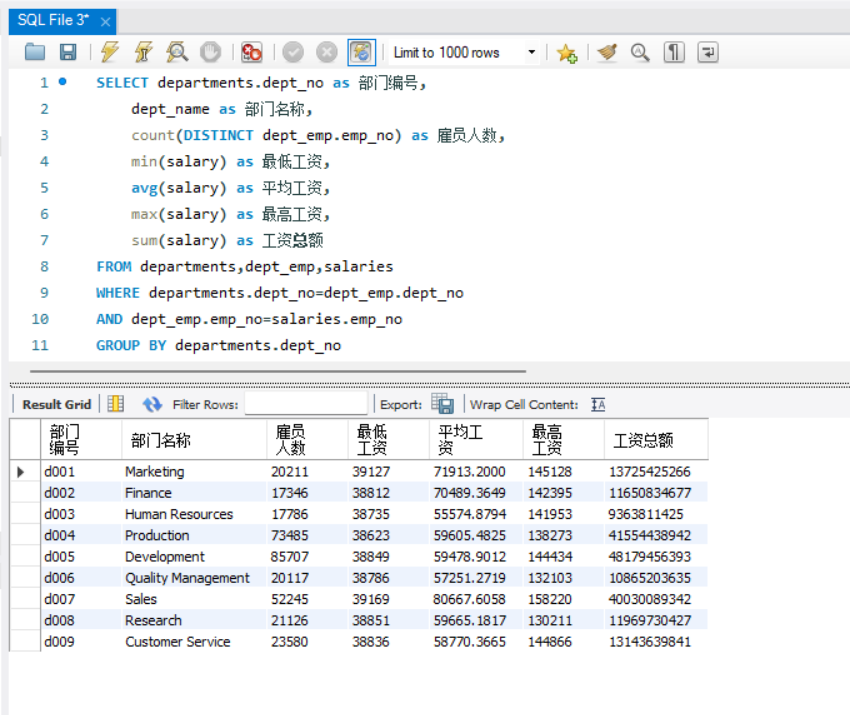


图9-1 查询一的查询代码及结果

## 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

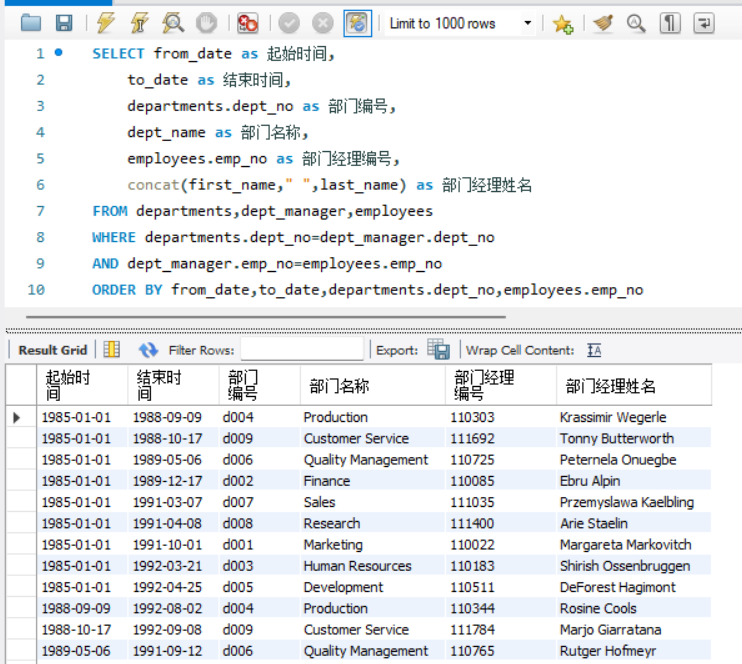


图9-2 查询二的查询代码及结果

## 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

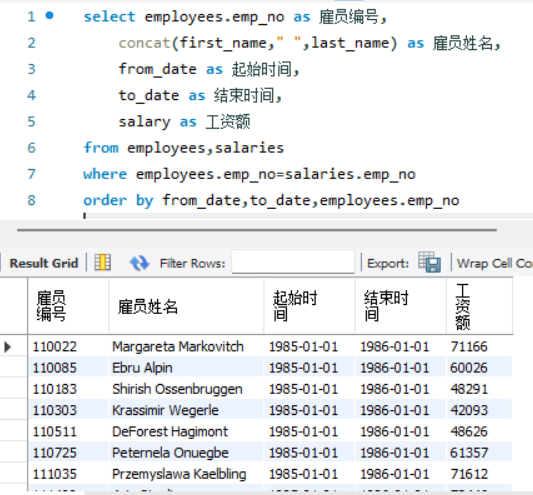


图9-3 查询三的查询代码及结果

## 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

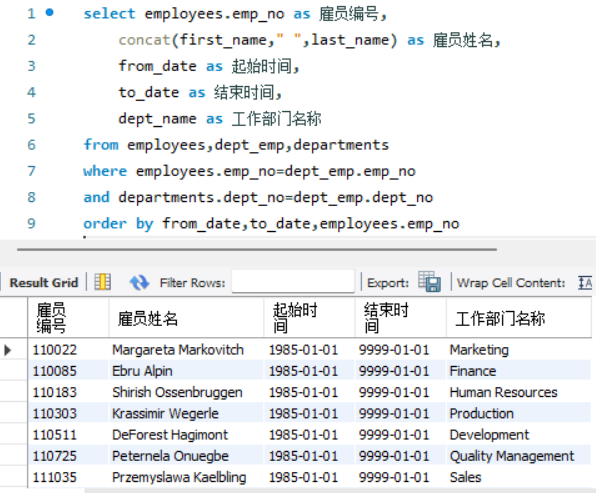


图9-4 查询四的查询代码及结果

## 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

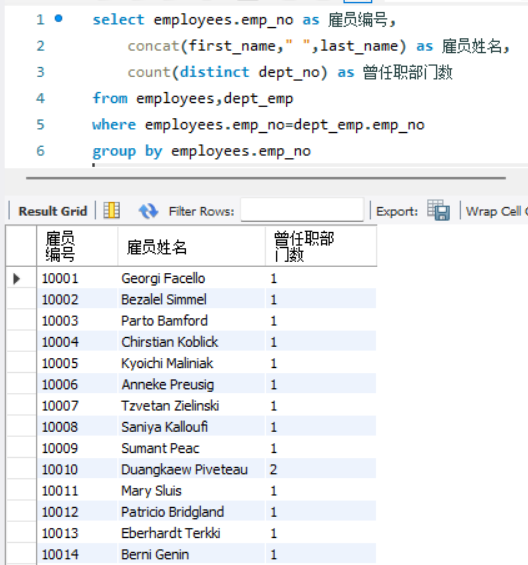


图9-5 查询五的查询代码及结果

## 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

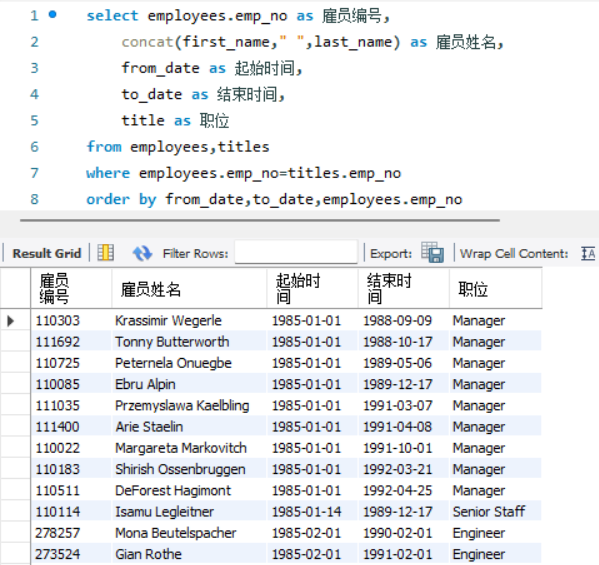


图9-6 查询六的查询代码及结果

## 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

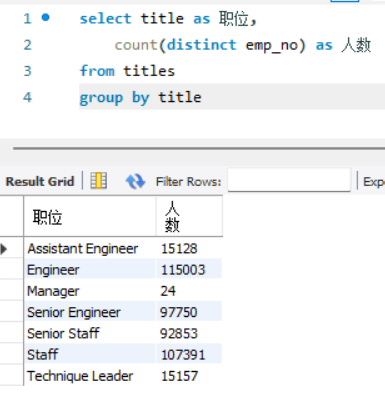


图9-7 查询七的查询代码及结果

## 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

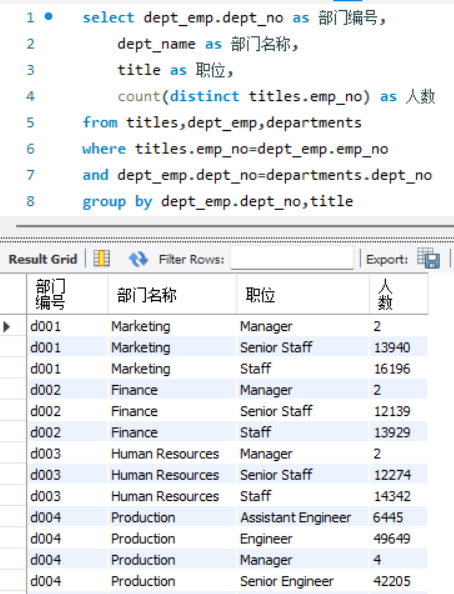


图9-8 查询八的查询代码及结果

## 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)

该查询语句运用了嵌套的select，即在每一个emp\_no寻找对应salary的最大值，查询与该最大值相等的记录中的from\_date，to\_date。

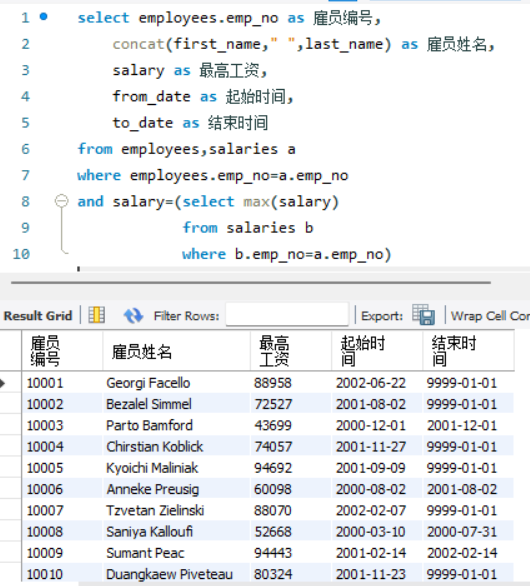


图9-9 查询九的查询代码及结果

## 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

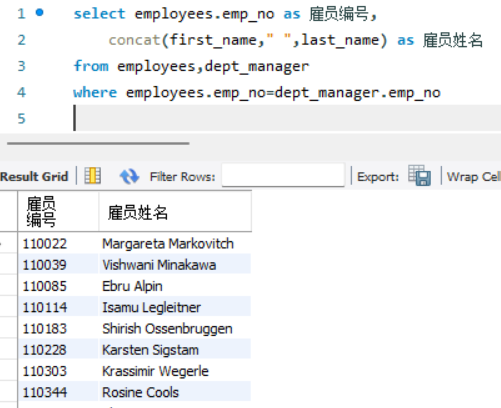


图9-10 查询十的查询代码及结果

## 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

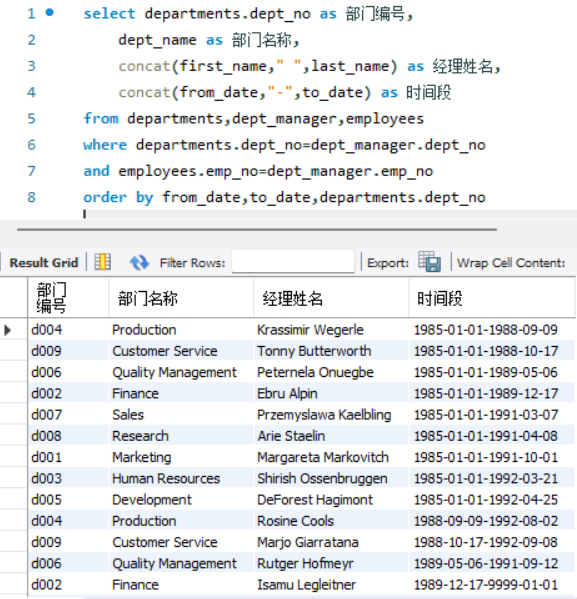


图9-11 查询十一的查询代码及结果

## 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



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。

## 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9-13。

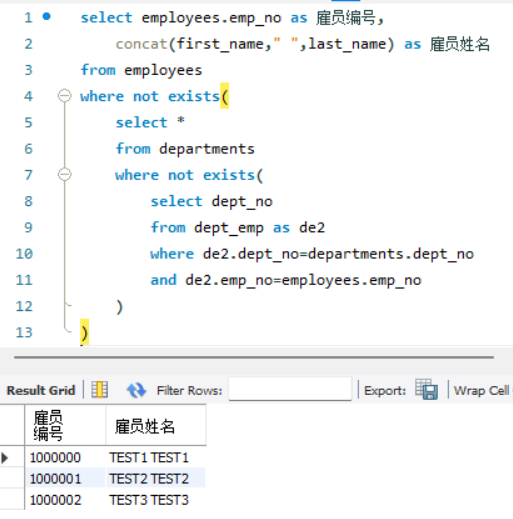


图9-13 执行插入后查询十三的查询代码及结果

# 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

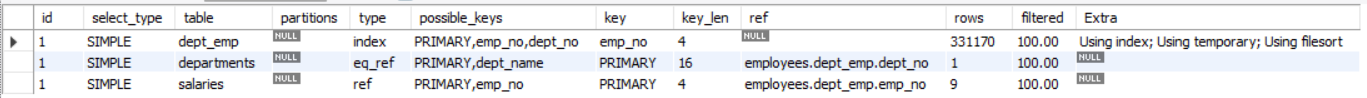


图10-1 查询一分析

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

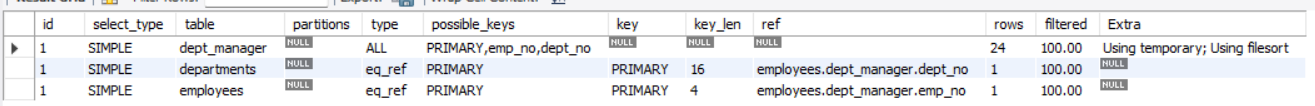


图10-2 查询二分析

读取表的顺序：dept\_manager，departments，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：ALL，eq\_ref，eq\_ref

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

表之间的引用：departments 引用dept\_manager.dept\_no，employees 引用dept\_manager.emp\_no

每张表有多少行被物理查询：24，1，1

### 10.1.3 查询三分析



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：299560，9

### 10.1.4 查询四分析

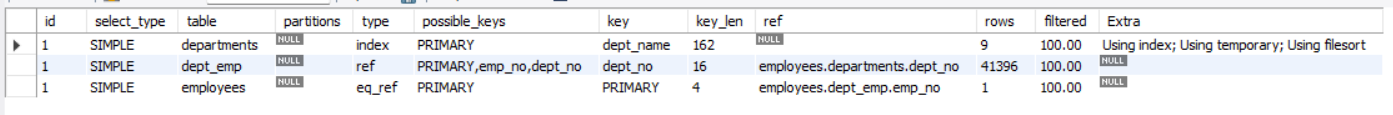


图10-4 查询四分析

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：9，41396，1

### 10.1.5 查询五分析

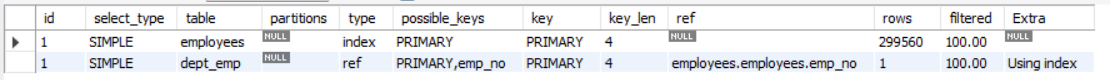


图10-5 查询五分析

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：299560，1

### 10.1.6 查询六分析

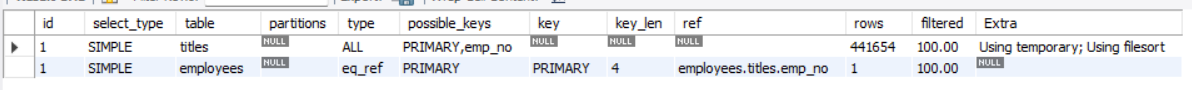


图10-6 查询六分析

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：441654，1

### 10.1.7 查询七分析



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：441654

### 10.1.8 查询八分析

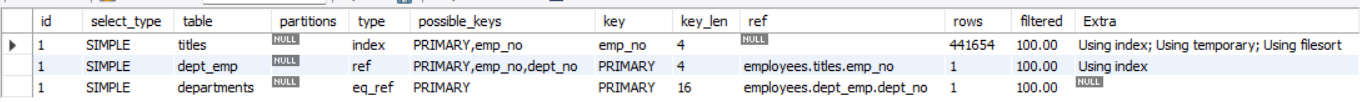


图10-8 查询八分析

读取表的顺序：titles，dept\_emp，departments

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，eq\_ref

哪些索引能够被实际使用：titles.emp\_no，dept\_emp的主键索引，employees的主键索引

表之间的引用：dept\_emp引用titles.emp\_no，departments引用dept\_emp.dept\_no

每张表有多少行被物理查询：441654，1，1

### 10.1.9 查询九分析

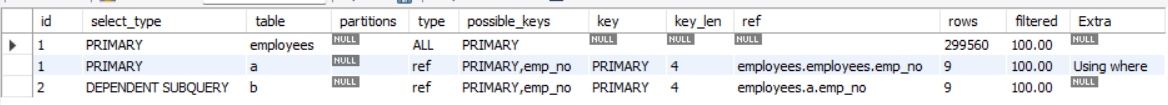


图10-9 查询九分析

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：299560，9，9

### 10.1.10 查询十分析

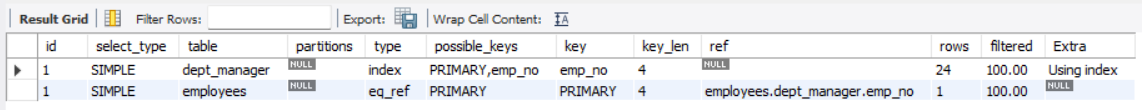


图10-10 查询十分析

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

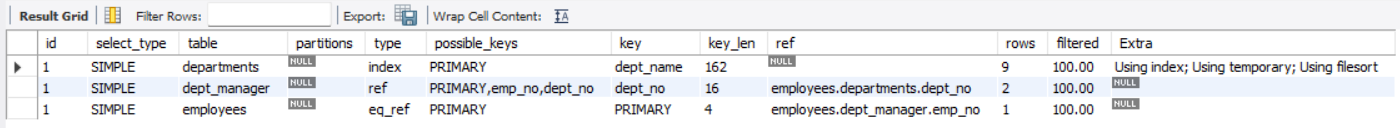
数据读取操作的操作类型：index，eq\_ref

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

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

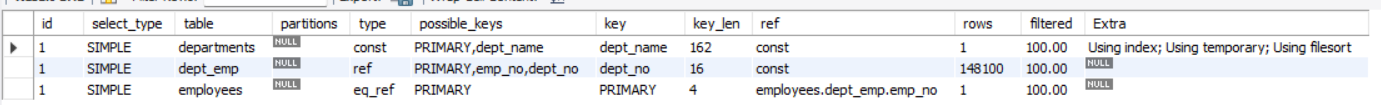


图10-12 查询十二分析

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：1，148100，1

### 10.1.13 查询十三分析

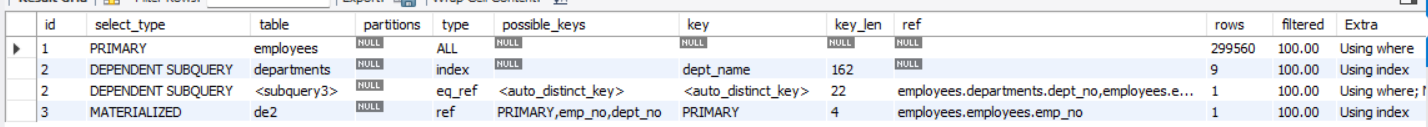


图10-13 查询十三分析

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：299560，9，1，1

## 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，判断哪些索引将可以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;

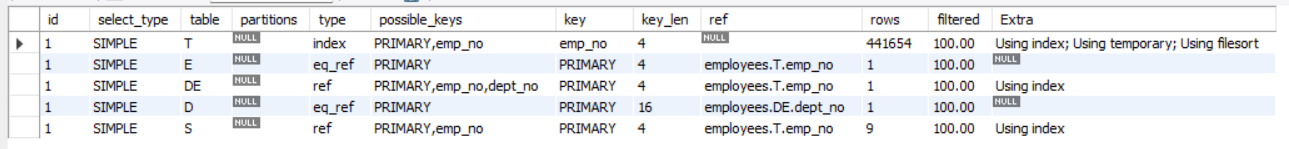


图10-14 explain分析结果

能够被实际使用的索引有：T.emp\_no，E的主键索引，DE的主键索引，D的主键索引，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21.156sec。



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20.937sec。



图10-16 有索引查询时间

可见索引对查询具有优化作用。

# SQL查询实验总结

该实验练习了SQL查询语句的使用、用EXPLAIN输出查询语句的查询计划并进行分析，最后利用EXPLAIN语句等进行了索引与查询优化的分析。实验帮助我熟练运用SQL进行数据查询和分析。

# 实验中遇到的问题和问题解决

## 12.1 插入嵌套查询时，Error code:1093

在进行插入嵌套查询时，遇到错误1093，插入代码是：

insert into organization

values

(select max(organizationno)+1 from organization),

'西安科技大学','029-85523032','雁塔中路58号',

(select busstopno from busstop where BusStopName='西安科技大学'),

0.3)

产生问题的原因是mysql中不允许将要更新的表中查询的数据用于本表的更新。解决方案是利用中间表，即多加一层select的嵌套。

## 12.2 插入嵌套查询时，Error code:1248

解决12.1所述的问题后，遇到错误1248，此时的代码是：

insert into organization

values

((select \* from (select max(organizationno)+1 from organization)), …

产生问题的原因是派生出的表需要有自己的别名。解决后代码如下：

…((select \* from (select max(organizationno)+1 from organization) as a), …

## 12.3 修改时，Error code:1175

更新数据时，如果where后面不是用主键进行约束的，会出现此错误。

update busstop

set busstoplocation='西安交通大学北门'

where busstopname='兴庆宫公园南门'

修改为：

update busstop

set busstoplocation='西安交通大学北门'

where busstopno=(select \* from(select busstopno from busstop where busstopname='兴庆宫公园南门') as a)

或者可以在Perference设置中关闭safe updates。

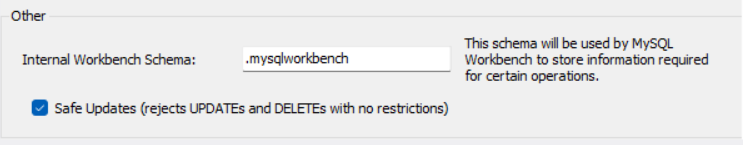


图12-1 关闭safe updates