

《数据库原理》课程 实验报告

Lab01 上机实验 1: Employees 数据库 (1)

一、实验目的

1. 通过上机练习巩固关系数据模型及其基本概念。
2. 通过上机练习巩固关系数据库语言 SQL。
3. 通过上机练习巩固关系数据库设计方法。

二、实验原理

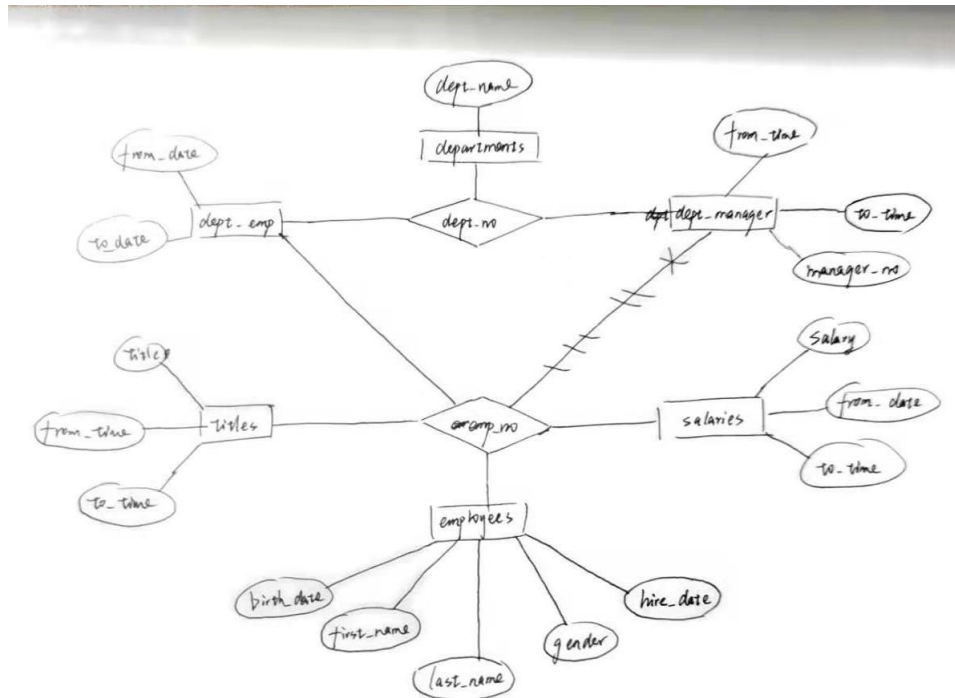
4. 采用 PostgreSQL 数据库作为实验用 DBMS。
5. 用 E/R 图建立数据库的概念模型。
6. 将 E/R 模型转换为关系模型。
7. 用 SQL 创建数据库模式。
8. 将数据批量装载到数据库中。
9. 用 SQL 进行查询和更新操作。

三、实验内容

1. 用户需求。
 - (1) 某公司为管理员工相关数据需要设计名为 Employees 的数据库。该数据库中要管理的数据包括：员工数据 (employees)、职称数据 (titles)、工资数据 (salaries)、部门数据 (departments) 等。
 - (2) 员工数据包括：员工编号 (emp_no)、出生日期 (birth_date)、名字 (first_name)、姓氏 (last_name)、性别 (gender)、入职日期 (hire_date)。
 - (3) 职称数据包括：职称名称 (title)、起始时间 (from_date)、终止时间 (to_date)。一条职称数据记录了某员工从起始时间到终止时间这个时间段内的职称名称。
 - (4) 工资数据包括：工资数额 (salary)、起始时间 (from_date)、终止时间 (to_date)。一条工资数据记录了某员工从起始时间到终止时间这个时间段内的工资数额。
 - (5) 部门数据包括：部门编号 (dept_no)、部门名称 (dept_name)。
 - (6) 部门和员工间的关系 1 (dept_emp)：一个部门下属有多名员工，一名员工可隶属于多个部门。需要记录某员工为某部门工作的起始时间和终止时间。

- (7) 部门和员工间的关系 2 (dept_manager) 一个部门有多位经理 (不用区分正副职), 经理也是一名员工, 一名员工可同时担任多个部门的经理。需要记录某员工担任某部门经理的起始时间和终止时间。

2. 分析用户需求, 画出 Employees 数据库的 E/R 模型图。



3. 将 E/R 模型转换为关系模型, 用 SQL 创建关系表, 写出 CREATE TABLE 语句。

要求: 用 CONSTRAINT 关键字建立有名称的主键和外键约束。

主键名称格式为: pk_表名

外键名称格式为: fk_本表名_引用表名

```
CREATE TABLE employees(  
    emp_no      INT,  
    birth_date  DATE,  
    first_name  CHAR(20),  
    last_name   CHAR(20),  
    gender      CHAR(1) CHECK(gender='M' OR gender='F')  
    NOT NULL,  
    hire_date   DATE,  
    CONSTRAINT pk_employees PRIMARY KEY(emp_no)  
);
```

```
CREATE TABLE titles(  
    emp_no      INT,  
    title       CHAR(30),  
    from_date   DATE,  
    to_date     DATE,  
    CONSTRAINT fk_titles_employees FOREIGN KEY(emp_no)  
    REFERENCES employees(emp_no)  
);
```

```
CREATE TABLE salaries(  
    emp_no      INT,  
    salary      INT,  
    from_date   DATE,  
    to_date     DATE,  
    CONSTRAINT fk_salaries_employees FOREIGN KEY(emp_no)  
    REFERENCES employees(emp_no)  
);
```

```
CREATE TABLE departments(  
    dept_no      CHAR(10),  
    dept_name    CHAR(30),  
    CONSTRAINT pk_departments PRIMARY KEY(dept_no)  
);  
  
CREATE TABLE dept_emp(  
    emp_no       INT,  
    dept_no      CHAR(10),  
    from_date    DATE,  
    to_date      DATE,  
    CONSTRAINT fk_dept_emp_employees FOREIGN KEY(emp_no)  
        REFERENCES employees(emp_no)  
);  
  
CREATE TABLE dept_manager(  
    dept_no      CHAR(10),  
    manager_no   INT,  
    from_date    DATE,  
    to_date      DATE,  
    CONSTRAINT pk_dept_manager PRIMARY KEY(manager_no),  
    CONSTRAINT fk_dept_manager_departments FOREIGN  
KEY(dept_no)  
        REFERENCES departments(dept_no)  
);
```

4. 将提供的示例数据导入到已创建的表中。

数据文件说明:

data_employees.txt	员工数据
data_departments.txt	部门数据
data_dept_emp.txt	部门员工关系数据
data_dept_manager.txt	部门经理关系数据

学号: 3019244266 姓名: 李润泽 日期: 2021 年 3 月 24 日 地点: 智能与计算学部

data_salaries.txt 工资数据

data_titles.txt 职称数据

使用 PostgreSQL 提供的批量导入数据的语句 COPY...FROM...。

(关于 COPY...FROM...语法解释, 请自己查询 PostgreSQL 文档)

导入之后的结果:

employees 表	300024	行数据
departments 表	9	行数据
dept_emp 表	331603	行数据
dept_manager 表	24	行数据
titles 表	443308	行数据
salaries 表	2844047	行数据

COPY

```
employees(emp_no,birth_date,first_name,last_name,gender,hire_date)
```

FROM

```
'D:\Database_Lab\lab01_student\data_employees.txt'(FORMAT 'text', DELIMITER ',');
```

```
COPY departments(dept_no,dept_name)
```

FROM

```
'D:\Database_Lab\lab01_student\data_departments.txt'(FORMAT 'text', DELIMITER ',');
```

```
COPY dept_emp(emp_no,dept_no,from_date,to_date)
```

FROM

```
'D:\Database_Lab\lab01_student\data_dept_emp.txt'(FORMAT 'text', DELIMITER ',');
```

COPY

```
dept_manager(dept_no,manager_no,from_date,to_date)
```

FROM

```
'D:\Database_Lab\lab01_student\data_dept_manager.txt'(FORMAT 'text', DELIMITER ',');
```

```
COPY titles(emp_no,title,from_date,to_date)
```

学号: 3019244266 姓名: 李润泽 日期: 2021 年 3 月 24 日 地点: 智能与计算学部

```
FROM
'D:\Database_Lab\lab01_student\data_titles.txt' (FORMAT
'text', DELIMITER ',');
COPY salaries(emp_no,salary,from_date,to_date)
FROM
'D:\Database_Lab\lab01_student\data_salaries.txt' (FORM
AT 'text', DELIMITER ',');
```

5. 按照下列查询要求编写 SQL 语句。

5.1 返回前 10 行员工数据。

(提示: 用 **LIMIT** 关键字, 具体用法查文档)

```
SELECT *
FROM employees
LIMIT 10;
```

查询执行结果:

104 -- 5.1

105 SELECT *

106 FROM employees

107 LIMIT 10;

108

Data Output

Explain

Messages

Notifications

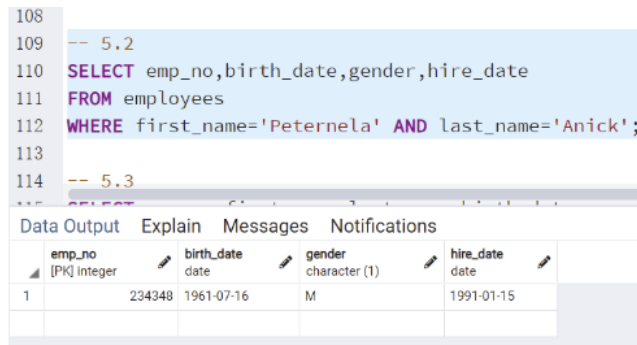
	emp_no [PK] integer	birth_date date	first_name character (20)	last_name character (20)	gender character (1)	hire_date date
1	10001	1953-09-02	Georgi	Facello	M	1986-06-26
2	10002	1964-06-02	Bezael	Simmel	F	1985-11-21
3	10003	1959-12-03	Parto	Bamford	M	1986-08-28
4	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
5	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
6	10006	1953-04-20	Anneke	Preusig	F	1989-06-02
7	10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
8	10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
9	10009	1952-04-19	Sumant	Peac	F	1985-02-18
10	10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24

5.2 查询 first_name 为 Peternela 且 last_name 为 Anick 的员工的编号、出生日期、性别和入职日期。

学号: 3019244266 姓名: 李润泽 日期: 2021 年 3 月 24 日 地点: 智能与计算学部

```
SELECT emp_no,birth_date,gender,hire_date
FROM employees
WHERE first_name='Peternela' AND last_name='Anick';
```

查询执行结果:



The screenshot shows a SQL query execution interface. The query is: `SELECT emp_no,birth_date,gender,hire_date FROM employees WHERE first_name='Peternela' AND last_name='Anick';`. The results are displayed in a table with the following columns: emp_no (integer), birth_date (date), gender (character(1)), and hire_date (date). A single row is returned with the values: 234348, 1961-07-16, M, and 1991-01-15.

emp_no	birth_date	gender	hire_date
234348	1961-07-16	M	1991-01-15

5.3 查询出生日期在 1961-7-15（包括）到 1961-7-20（包括）之间的员工的编号、姓名和出生日期。

```
SELECT emp_no,first_name,last_name,birth_date
FROM employees
WHERE birth_date>='1961-7-15' AND
birth_date<='1961-7-20';
```

查询执行结果:

(386 行数据返回)

```

113
114 -- 5.3
115 SELECT emp_no,first_name,last_name,birth_date
116 FROM employees
117 WHERE birth_date>='1961-7-15' AND birth_date<='1961-7-20';
118

```

	emp_no [PK] integer	first_name character (20)	last_name character (20)	birth_date date
1	10143	Sakthirel	Bakhtari	1961-07-16
2	10336	Goa	Rothe	1961-07-16
3	10482	Goa	Pleszkun	1961-07-17
4	10798	Elliott	Bednarek	1961-07-18
5	11600	Yurij	Pardalos	1961-07-18
6	11712	Mohua	Lyonns	1961-07-17
7	13530	Irene	Greibach	1961-07-15
8	13935	Greger	Gopalakrishnan	1961-07-20
9	14615	Dinkar	Mitchem	1961-07-15
10	14737	Houman	Impagliazzo	1961-07-17
11	15097	Gudjon	Winter	1961-07-19

5.4 查询所有 first_name 中含有前缀 Peter 或 last_name 中含有前缀 Peter 的员工数据（返回所有列）。

```

SELECT *
FROM employees
WHERE first_name LIKE 'Peter%' OR last_name LIKE 'Peter%';

```

查询执行结果：
（771 行数据返回）

```

119 -- 5.4
120 SELECT *
121 FROM employees
122 WHERE first_name LIKE 'Peter%' OR last_name LIKE 'Peter%';
123
124 -- 5.5

```

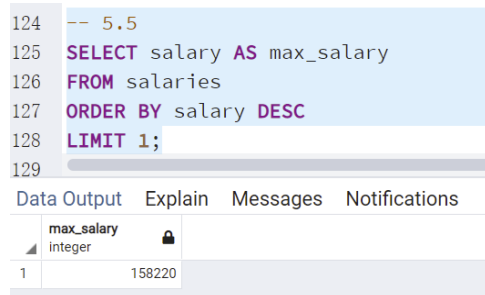
	emp_no [PK] integer	birth_date date	first_name character (20)	last_name character (20)	gender character (1)	hire_date date
1	10470	1956-04-12	Peternela	Iwayama	M	1988-06-01
2	10625	1958-02-08	Leszek	Petereit	F	1985-10-31
3	10895	1952-03-21	Vishwani	Petersohn	M	1986-04-26
4	10975	1962-07-27	Peternela	Birnbaum	F	1986-10-31
5	11212	1961-08-10	Teruyuki	Peternell	M	1985-10-16
6	11268	1954-05-14	Aimee	Petersohn	F	1985-05-17
7	11358	1960-04-21	Nevin	Petereit	F	1990-06-02
8	11499	1957-07-02	Peternela	Chartres	F	1989-03-08
9	12409	1963-08-01	Ebru	Petereit	F	1994-10-07
10	12597	1952-09-07	Kazuhito	Petereit	F	
11	13135	1959-02-25	Ranya	Peternell	F	

Successfully run. Total c

5.5 查询工资数额的最大值，并将查询结果的列名命名为 max_salary。


```
SELECT salary AS max_salary
FROM salaries
ORDER BY salary DESC
LIMIT 1;
```

查询执行结果:



The screenshot shows a SQL query execution interface. The query is: `SELECT salary AS max_salary FROM salaries ORDER BY salary DESC LIMIT 1;`. The results are displayed in a table with one column, `max_salary`, and one row with the value `158220`.

	max_salary integer
1	158220

5.6 查询部门编号及相应部门的员工人数，并按照部门编号由小到大的顺序排序（将员工人数列命名为 `dept_emp_count`）。

```
SELECT dept_no, count(dept_no) AS dept_emp_count
FROM dept_emp
GROUP BY dept_no
ORDER BY dept_no ASC;
```

查询执行结果:

```
130 -- 5.6
131 SELECT dept_no, count(dept_no) AS dept_emp_count
132 FROM dept_emp
133 GROUP BY dept_no
134 ORDER BY dept_no ASC;
135
136 -- 5.7
```

	dept_no character (10)	dept_emp_count bigint	
1	d001	20211	
2	d002	17346	
3	d003	17786	
4	d004	73485	
5	d005	85707	
6	d006	20117	
7	d007	52245	
8	d008	21126	
9	d009	23580	

5.7 查询员工“Peternela Anick”的员工编号、所在部门编号和在该部门的工作起始时间。

```
SELECT
dept_emp.emp_no,dept_emp.dept_no,dept_emp.from_date
FROM employees,dept_emp
WHERE      employees.emp_no=dept_emp.emp_no      AND
employees.first_name='Peternela'      AND
employees.last_name='Anick';
```

查询执行结果:

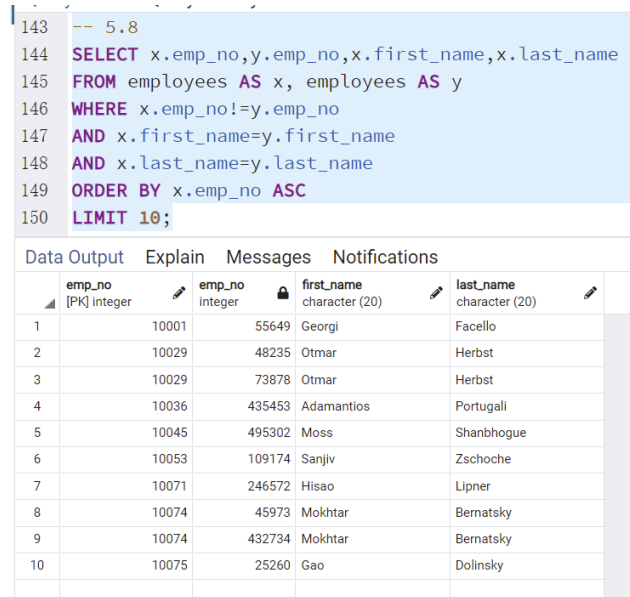
```
136 -- 5.7
137 SELECT dept_emp.emp_no,dept_emp.dept_no,dept_emp.from_date
138 FROM employees,dept_emp
139 WHERE employees.emp_no=dept_emp.emp_no
140 AND employees.first_name='Peternela'
141 AND employees.last_name='Anick';
142
143 -- 5.8
```

	emp_no integer	dept_no character (10)	from_date date	
1	234348	d009	1991-01-15	

5.8 查询姓名相同的员工 x 和员工 y 的编号和姓名（只列出前 10 行结果）。

```
SELECT x.emp_no,y.emp_no,x.first_name,x.last_name
FROM employees AS x, employees AS y
WHERE x.emp_no!=y.emp_no
AND x.first_name=y.first_name
AND x.last_name=y.last_name
ORDER BY x.emp_no ASC
LIMIT 10;
```

查询执行结果:



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

```
-- 5.8
SELECT x.emp_no,y.emp_no,x.first_name,x.last_name
FROM employees AS x, employees AS y
WHERE x.emp_no!=y.emp_no
AND x.first_name=y.first_name
AND x.last_name=y.last_name
ORDER BY x.emp_no ASC
LIMIT 10;
```

Below the query editor, there is a table with the following columns: emp_no [PK] integer, emp_no integer, first_name character (20), and last_name character (20). The table contains 10 rows of data:

	emp_no [PK] integer	emp_no integer	first_name character (20)	last_name character (20)
1		10001	Georgi	Facello
2		10029	Otmar	Herbst
3		10029	Otmar	Herbst
4		10036	Adamantios	Portugali
5		10045	Moss	Shanhogue
6		10053	Sanjiv	Zschoche
7		10071	Hisao	Lipner
8		10074	Mokhtar	Bernatsky
9		10074	Mokhtar	Bernatsky
10		10075	Gao	Dolinsky

5.9 查询姓名为“Margo Anily”的员工编号和出生日期为“1959-10-30”且入职日期为“1989-09-12”的员工编号的并集。

```
SELECT emp_no FROM employees WHERE first_name='Margo' AND
last_name='Anily'
UNION
SELECT emp_no FROM employees WHERE
birth_date='1959-10-30' AND hire_date='1989-09-12';
```

学号： 3019244266 姓名： 李润泽 日期： 2021 年 3 月 24 日 地点： 智能与计算学部

查询执行结果：

151

152 -- 5.9

153 SELECT emp_no FROM employees WHERE first_name='Margo' AND last_name='Anily'

154 UNION

155 SELECT emp_no FROM employees WHERE birth_date='1959-10-30' AND hire_date='1989-09-12';

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

1323

1324

1325

1326

1327

1328

1329

1330

1331

1332

1333

1334

1335

1336

1337

1338

1339

1340

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

1415

1416

1417

1418

1419

1420

1421

1422

1423

1424

1425

1426

1427

1428

1429

1430

1431

1432

1433

1434

1435

1436

1437

1438

1439

1440

1441

1442

1443

1444

1445

1446

1447

1448

1449

1450

1451

1452

1453

1454

1455

1456

1457

1458

1459

1460

1461

1462

1463

1464

1465

1466

1467

1468

1469

1470

1471

1472

1473

1474

1475

1476

1477

1478

1479

1480

1481

1482

1483

1484

1485

1486

1487

1488

1489

1490

1491

1492

1493

1494

1495

1496

1497

1498

1499

1500

1501

1502

1503

1504

1505

1506

1507

1508

1509

1510

1511

1512

1513

1514

1515

1516

1517

1518

1519

1520

1521

1522

1523

1524

1525

1526

1527

1528

1529

1530

1531

1532

1533

1534

1535

1536

1537

1538

1539

1540

1541

1542

1543

1544

1545

1546

1547

1548

1549

1550

1551

1552

1553

1554

1555

1556

1557

1558

1559

1560

1561

1562

1563

1564

1565

1566

1567

1568

1569

1570

1571

1572

1573

1574

1575

1576

1577

1578

1579

1580

1581

1582

1583

1584

1585

1586

1587

1588

1589

1590

1591

1592

1593

1594

1595

1596

1597

1598

1599

1600

1601

1602

1603

1604

1605

1606

1607

1608

1609

1610

1611

1612

1613

1614</

5.10 查询员工 “Margo Anily” 所在的部门的名称（要求用子查询实现）。

```
SELECT dept_name
FROM departments
WHERE departments.dept_no=(
    SELECT dept_emp.dept_no
    FROM dept_emp
    WHERE dept_emp.emp_no=(
        SELECT employees.emp_no
        FROM employees
        WHERE first_name='Margo' AND last_name='Anily'
    )
);
```

查询执行结果：

```
157 -- 5.10
158 SELECT dept_name
159 FROM departments
160 WHERE departments.dept_no=(
161     SELECT dept_emp.dept_no
162     FROM dept_emp
163     WHERE dept_emp.emp_no=(
164         SELECT employees.emp_no
165         FROM employees
166         WHERE first_name='Margo' AND last_name='Anily'
167     )
168 )
```

	dept_name	
1	Production	

5.11 要求用 JOIN...ON 连接语法实现查询 5.10。

```
SELECT dept_name
FROM departments
JOIN dept_emp ON departments.dept_no=dept_emp.dept_no
JOIN employees ON (dept_emp.emp_no=employees.emp_no
                  AND employees.first_name='Margo'
                  AND employees.last_name='Anily'
                  );
```

查询执行结果:

```
170 -- 5.11
171 SELECT dept_name
172 FROM departments
173 JOIN dept_emp ON departments.dept_no=dept_emp.dept_no
174 JOIN employees ON (dept_emp.emp_no=employees.emp_no
175                   AND employees.first_name='Margo'
176                   AND employees.last_name='Anily'
177                   );
178
179 -- 5.12
```

	dept_name	
1	Production	

5.12 查询在全部部门中工作过的员工的编号和姓名（提示：用 NOT EXISTS 连接的子查询）。

```
SELECT employees.emp_no,first_name,last_name
FROM employees
WHERE EXISTS (
    SELECT dept_emp.emp_no,count (dept_emp.dept_no)
    FROM dept_emp
    GROUP BY dept_emp.emp_no
    HAVING count (dept_emp.dept_no)=9
);
```

查询执行结果：



```
179 -- 5.12
180 SELECT employees.emp_no,first_name,last_name
181 FROM employees
182 WHERE EXISTS (
183     SELECT dept_emp.emp_no,count (dept_emp.dept_no)
184     FROM dept_emp
185     GROUP BY dept_emp.emp_no
186     HAVING count (dept_emp.dept_no)=9
187 );
188
```

emp_no	first_name	last_name
[PK] integer	character (20)	character (20)

5.13 查询员工人数大于等于 50000 的部门编号、部门名称和部门员工人数，按照部门编号由小到大的顺序排序（将部门员工人数列命名为 dept_emp_count）。

```
SELECT x.dept_no,y.dept_name,x.dept_emp_count
FROM departments y
JOIN (
    SELECT dept_emp.dept_no, count (dept_emp.emp_no) AS
dept_emp_count
    FROM dept_emp
```

学号: 3019244266 姓名: 李润泽 日期: 2021 年 3 月 24 日 地点: 智能与计算学部

```
GROUP BY dept_emp.dept_no
HAVING COUNT(dept_emp.emp_no) >= 50000
ORDER BY dept_emp.dept_no ASC
) x
ON y.dept_no=x.dept_no;
```

查询执行结果:

```
189 -- 5.13
190 SELECT x.dept_no,y.dept_name,x.dept_emp_count
191 FROM departments y
192 JOIN(
193     SELECT dept_emp.dept_no, count(dept_emp.emp_no) AS dept_emp_count
194     FROM dept_emp
195     GROUP BY dept_emp.dept_no
196     HAVING COUNT(dept_emp.emp_no) >= 50000
197     ORDER BY dept_emp.dept_no ASC
198 ) x
199 ON y.dept_no=x.dept_no;
200
```

Data Output Explain Messages Notifications

	dept_no character (10)	dept_name character (30)	dept_emp_count bigint	
1	d004	Production	73485	
2	d005	Development	85707	
3	d007	Sales	52245	

5.14 在员工表中添加一行记录:

(10000, 1981-10-1, Jimmy, Lin, M, 2011-12-8)

```
INSERT INTO employees
VALUES (10000, '1981-10-1', 'Jimmy', 'Lin', 'M', '2011-12-8'
);
```

查询执行结果:

(1 行受影响)

```
203 -- 5.14
204 INSERT INTO employees
205 VALUES(10000,'1981-10-1','Jimmy','Lin','M','2011-12-8');
206
207 SELECT *
208 FROM employees
209 WHERE emp_no=10000;
210
```

	emp_no [PK] integer	birth_date date	first_name character (20)	last_name character (20)	gender character (1)	hire_date date
1	10000	1981-10-01	Jimmy	Lin	M	2011-12-08

5.15 将 5.14 添加的员工记录的 first_name 属性值修改为 Jim。

```
UPDATE employees
SET first_name='Jim'
WHERE emp_no=10000;
```

```
--SELECT *
--FROM employees
--WHERE emp_no=10000;
```

查询执行结果:

(1 行受影响)

```
207 -- 5.15
208 UPDATE employees
209 SET first_name='Jim'
210 WHERE emp_no=10000;
211
212 --SELECT *
213 --FROM employees
214 --WHERE emp_no=10000;
215
216 -- 5.16
```

	emp_no [PK] integer	birth_date date	first_name character (20)	last_name character (20)	gender character (1)	hire_date date
1	10000	1981-10-01	Jim	Lin	M	2011-12-08

5.16 删除 5.14 添加的员工记录。

```
DELETE
```


学号: 3019244266 姓名: 李润泽 日期: 2021 年 3 月 24 日 地点: 智能与计算学部

```
FROM employees
WHERE emp_no=10000;
```

查询执行结果:

(1 行受影响)

5.17 在员工表中添加一行记录:

(10001, 1981-10-1, Jimmy, Lin, M, 2011-12-8), 观察执行输出结果。

```
INSERT INTO employees
VALUES (10001, '1981-10-1', 'Jimmy', 'Lin', 'M', '2011-12-8'
);
```

```
225 -- 5.17
226 INSERT INTO employees
227 VALUES (10001, '1981-10-1', 'Jimmy', 'Lin', 'M', '2011-12-8');
228
229 -- 5.18
230
```

Data Output	Explain	Messages	Notifications
ERROR: 错误: 重复键违反唯一约束"pk_employees"			
DETAIL: 键值"(emp_no)=(10001)" 已经存在			
SQL state: 23505			

5.18 删除编号为 10001 的员工, 观察执行输出结果。

```
DELETE
FROM employees
WHERE emp_no=10001;
```

```
229 -- 5.18
230 DELETE
231 FROM employees
232 WHERE emp_no=10001;
233
```

Data Output	Explain	Messages	Notifications
ERROR: 错误: 在 "employees" 上的更新或删除操作违反了在 "titles" 上的外键约束 "fk_titles_employees"			
DETAIL: 键值对(emp_no)=(10001)仍然是从表"titles"引用的.			
SQL state: 23503			

学号： 3019244266 姓名： 李润泽 日期： 2021 年 3 月 24 日 地点： 智能与计算学部

四、实验总结

(300 字以上)

Lab01 结束之后，我感触很多，在这几天的实验中，我对数据库的理论和实操知识掌握了很多。期间我学到了很多，包括建表、导入数据、查询、插入与删除等操作。

表是建立关系数据库的基本结构，用于存储数据具有已定义的属性。在操作的过程中，我取得了巨大的收获，当然也发现了许多的问题：

1. 将 txt 文件导入数据库的期间，我始终无法访问该文件，通过和同学的交流我发现文件夹需要设定为共享，即把 lab01_student 文件夹设置为共享文件夹，才可以将数据导入数据库。

2. 在设计表的时候，我们需要使用正确的字段类型，其中，在比较 Date 类型的时候，日期需要用单引号 (') 修饰，如 5.4 题。

3. 操作试图查询的工作中，我编写代码发现，一个表的主键需要是另一个表的外键才可以对另一个表进行引用。

4. 操作统计部门人数并按部门编号排序的时候，我们需要将 GROUP BY 操作放在 ORDER BY 操作之前才可以进行正常查询，如 5.6 题。

5. 进行两个表连接操作的时候，我们需要用“表名+表的属性”的形式来编写，这样防止 error，并且在观察代码的时候更加清晰。

6. 建议编写代码的时候，有一个好的排版让人心情舒畅 OuO。

在做完 5.18 之后，我很欣慰，历经共计 5 个小时的实验取得了圆满成功。（话说时间长达 5 个小时实在是太菜了 QwQ）通过实验，我对原有的知识进行了一定的巩固，并且在一定程度上获取了在课堂上没学到的新知识。总之，通过此次课程设计，我收获匪浅。