数据库系统作业

1.调研某一应用领域，给出该应用领域的详细需求描述

对于医院信息管理系统而言，有如下需求描述：

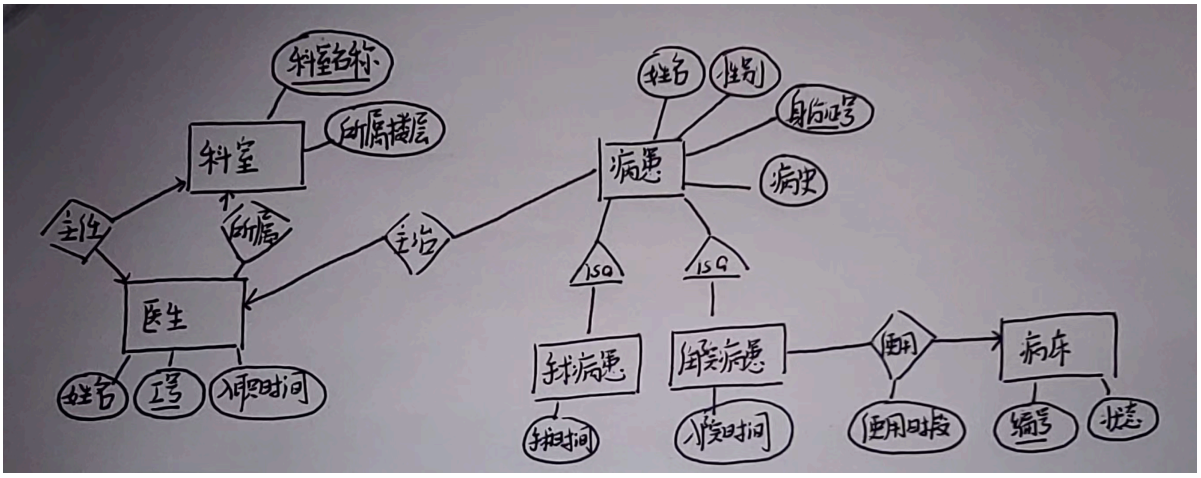
(1)医院下分多个科室，每个科室有各自的科室名称，科室各自分布在固定的楼层，每个科室下有多个医生，每个科室有且仅有一名主任医生，每个医生需要记录其姓名，入职时间，并以工号进行区分

(2)医院中有多名病患，需要记录每名病患的姓名，性别，过往病史以及身份证号，每名病患都会有一个主治医生，但每个医生可以同时作为多名病患的主治医生。病患分为手术病患和住院病患，手术病患需要记录其手术时间，住院病患则需记录住院入院时间。

(3)医院中有多个病床，每个病床需要记录其编号和目前的使用状态(空闲中还是使用中)，住院病患使用这些病床，需要记录使用的时间段

2.采用教材介绍的方法实现如下设计

(a)画出该领域的概念模型ER图，如下图所示：



(b)将上述ER图转换为关系模式，并标明每个关系的主键属性和外键属性

科室={科室名称，所属楼层}；

主任={科室名称，医生工号}；

医生={姓名，工号，入职时间，科室名称(FK)}；

病患={姓名，性别，身份证号，病史，医生工号(FK)}

手术病患={身份证号，手术时间}；

住院病患={身份证号，入院时间}；

病床={编号，状态}；

使用={身份证号，编号，使用时段}；

(c)用SQL语句创建上述关系模式

```
//科室表
CREAT TABLE Departments(
    depname CHAR(30),
```

```
    floor INT(8),
    PRIMARY KEY(depname)
);
//主任表
CREAT TABLE Directors(
    depname CHAR(30),
    docid CHAR(20),
    PRIMARY KEY(depname,docid)
);
//医生表
CREAT TABLE Doctors(
    docname CHAR(30),
    docid CHAR(20) PRIMARY KEY,
    onboardtime INT,
    depname CHAR(30),
    FOREIGN KEY (depname) REFERENCES Departments(depname)
);
//病患表
CREAT TABLE Patients(
    patient_name CHAR(30),
    gender CHAR(10),
    personalID CHAR(20) PRIMARY KEY,
    medical_history CAHR(100),
    docid CHAR(20),
    FOREIGN KEY(docid) REFERENCES Doctors(docid)
);
//手术病患
CREAT TABLE OperPatients(
    personalID CHAR(20) PRIMARY KEY,
    operation_time DATE
);
//住院病患
CREAT TABLE Inpatients(
    personalID CHAR(20) PRIMARY KEY,
    hospitalizedtime DATE
);
//病床表
CREAT TABLE BEDS(
    bedid CHAR(20) PRIMARY KEY,
    statu CHAR(20)
);
//使用表
CREAT Uses(
    personalID CHAR(20) PRIMARY KEY,
    bedid CHAR(20) PRIMARY KEY,
    usingtime CHAR(30)
    FOREIGN KEY(personalID) REFERENCES Inpatients(personalID)
    FOREIGN KEY(bedid) REFERENCES Beds(bedid)
);
```

(d)单表查询：查询目前空闲状态的床位id

```
SELECT bedid
FROM BEDS
WHERE statu="free";
```

多表连接查询：查询医生编号为00153的医生在哪个楼层

```
SELECT floor
FROM Doctors,Departments
Where Doctors.depname=Departments.depname and docid = "00153";
```

多表嵌套查询：查询有基础病史为“AAA”的病患的主治医生的工号

```
SELECT docid
FROM Doctors
WHERE docid IN(
    SELECT docid
    FROM Patients
    WHERE medical_history = "AAA"
);
```

EXISTS查询：查询外科科室中入职时间大于五年(60个月)的医生工号

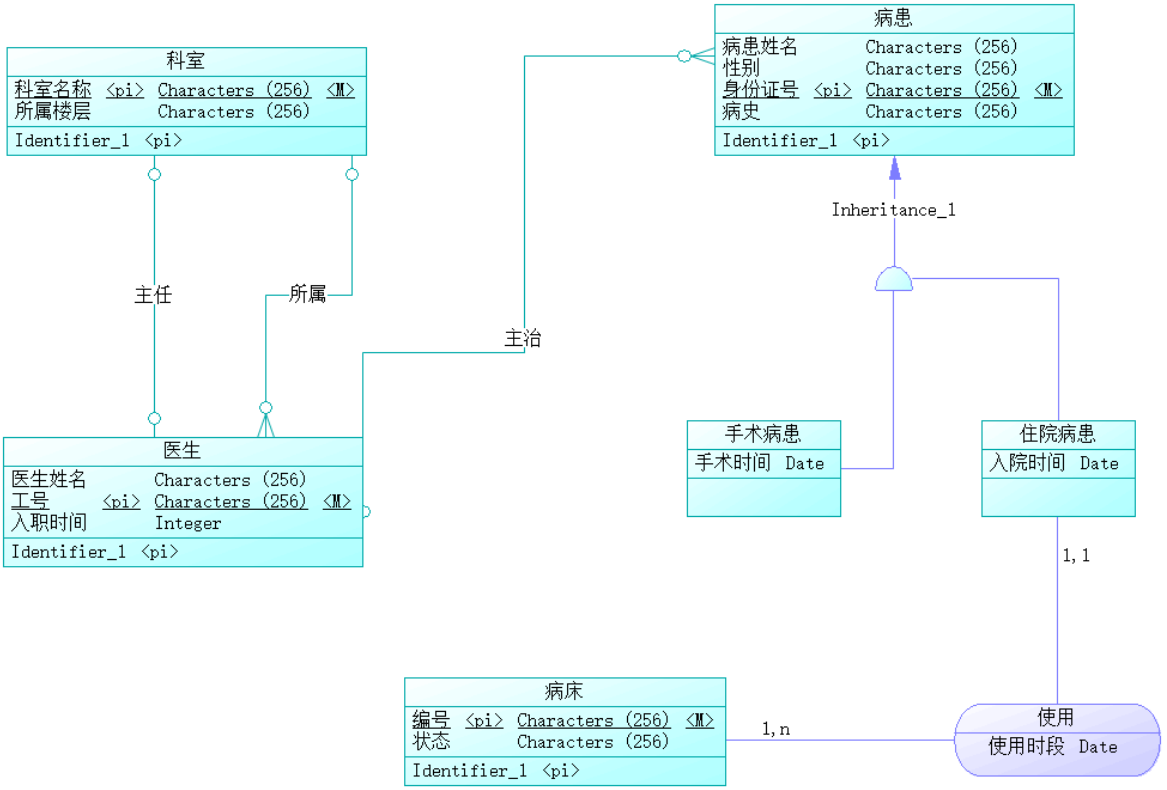
```
SELECT docid
FROM Doctors
WHERE EXISTS(
    SELECT *
    FROM Doctors,Departments
    WHERE Doctors.depname=Departments depname and depname="外科" and
onboardtime>60
);
```

聚合操作查询：查询每个科室的医生数目和医生们的平均入职时间

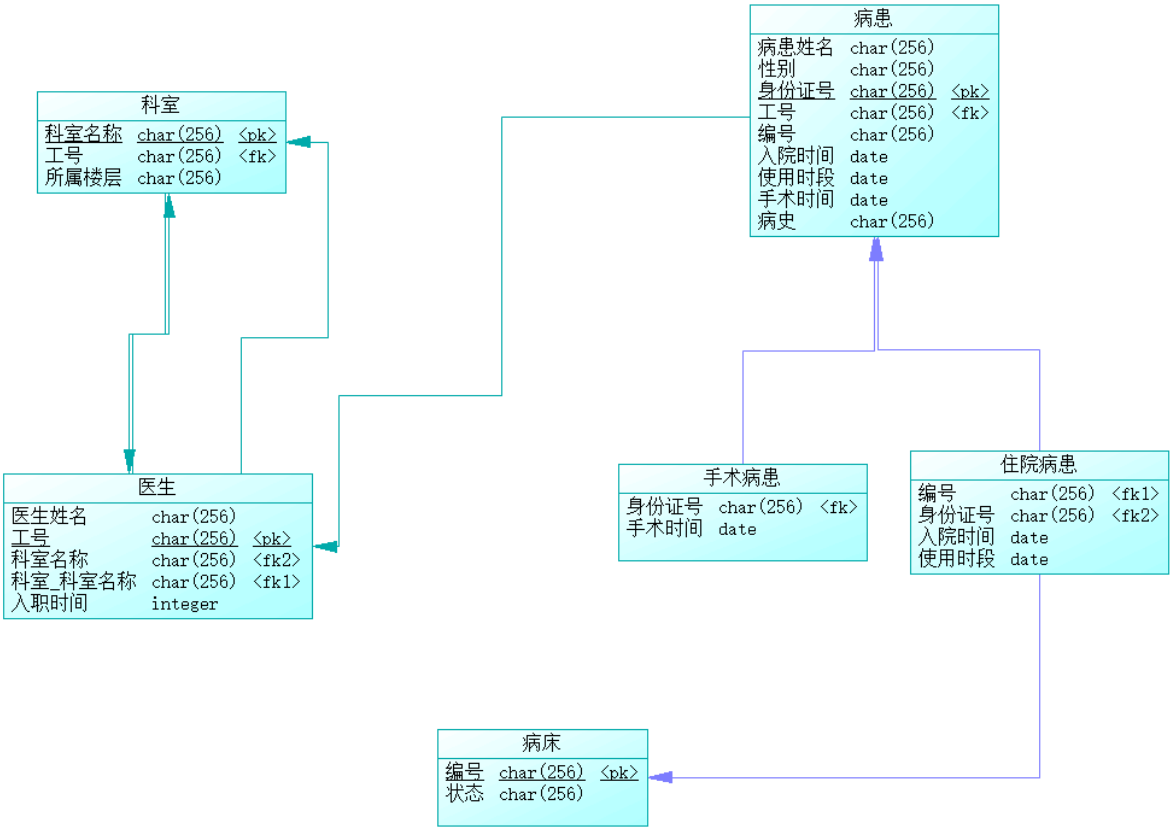
```
SELECT COUNT(*) AS doctor_sum,AVG(onboardtime) AS avg_onboardtime
FROM Doctors
GROUP BY depname;
```

3.使用PowerDesigner工具实现如下设计：

(a)画出该领域的概念模型ER图，给出ER图截图：



(b)使用PowerDesigner工具，将上述ER图转为关系模型图，给出关系模型图截图



(c)使用PowerDesigner工具，生成创建数据库的SQL语句

```
/*==============================================================*/
/* Table: 住院病患                                              */
/*==============================================================*/
```

```
create table 住院病患
(
   编号                   char(256)                      not null,
   身份证号                 char(256)                       null,
   入院时间                 date                            null,
   使用时段                 date                            null
);

/*==============================================================*/
/* Table: 医生                                                  */
/*==============================================================*/
create table 医生
(
   医生姓名                 char(256)                       null,
   工号                   char(256)                      not null,
   科室名称                 char(256)                       null,
   科室_科室名称                char(256)                       null,
   入职时间                 integer                         null
);

/*==============================================================*/
/* Table: 手术病患                                              */
/*==============================================================*/
create table 手术病患
(
   身份证号                 char(256)                       null,
   手术时间                 date                            null
);

/*==============================================================*/
/* Table: 病床                                                  */
/*==============================================================*/
create table 病床
(
   编号                   char(256)                      not null,
   状态                   char(256)                      null
);

/*==============================================================*/
/* Table: 病患                                                  */
/*==============================================================*/
create table 病患
(
   病患姓名                 char(256)                       null,
   性别                   char(256)                      null,
   身份证号                 char(256)                       not null,
   工号                   char(256)                      null,
   编号                   char(256)                      null,
   入院时间                 date                            null,
   使用时段                 date                            null,
   手术时间                 date                            null,
   病史                   char(256)                      null
);

/*==============================================================*/
/* Table: 科室                                                  */
```

```
/*==============================================================*/
create table 科室
(
   科室名称                 char(256)                      not null,
   工号                   char(256)                     null,
   所属楼层                 char(256)                      null
);
```

4.分析比较采用上述两种方法

(a) 两种关系模式的设计是否存在差异？如有差异，这种差异是否对后期的实现带来不同的影响？

存在一些差异，最大的差异体现在对关系表的构建上，在PowerDesigner中，并不会为某些关系生成单独的关系模型，一般是将其join到某个实体上，而自己创建关系模式时会为关系单独创建。

(b) PowerDesigner工具生成的SQL语句有什么样的特点？为什么会出现一些附加语句？它的作用是什么？

使用 PowerDesigner生成的SQL语句有如下特点：

- 包含完整的DDL语句（CREATE TABLE, ALTER TABLE等）
- 自动生成主键、外键约束
- 包含索引创建语句
- 有注释语句(表/列的注释)

而附加语句，比如说注释语句的出现，是为了增加其可读性和可维护性；约束检查类语句是为了确保数据完整性，在创建表后添加约束验证