数据库系统 week7作业 2211290 姚知言

1.设计一个员工管理系统

该公司有很多子公司（branchoffice），每个子公司以子公司ID（officeID）标识，除此之外还要记录公司名（officename），公司地址（officelocation）。

公司下设很多部门（department），以部门ID（departmentID）标识，除此之外还要记录部门名（departmentname）。

部门下雇佣雇员（employee），雇员分为技术员（technicist）和管理者（management），通过雇员ID（employeeID）进行标识。

对于技术员，除了雇员ID以外，还需要存储雇员名（employeename），入职时间（enrollmentdate），工资（salary），工作岗位（job），性别（sex），雇员类型（category），技术员等级（techlevel）。

对于管理者，除了雇员ID以外，还需要存储雇员名（employeename），入职时间（enrollmentdate），工资（salary），工作岗位（job），性别（sex），管理者等级（managelevel）。

每个部门需要有一个部长，由一名管理者担任，一个管理者仅能担任一个部门的部长。

公司中由很多项目，以项目ID（projectID）标识，还要存储项目名（projectname），预算（budget），预计收入（estimatedrevenue），开始时间（startdate），预计结束时间（estimatedenddate）。每个项目需要很多雇员参加（joinn），也需要一个负责人（principal），一个负责人可以同时负责多个项目，还需要一个责任分公司（responsibility），每个分公司可以同时负责多个项目。

2.a）该领域的ER图如下



b）关系模式如下

主键加下划线，外键字体为黄色。

technicant(employeeID,category,techlevel)

management(employeeID,managelevel,departmentID)

employee(employeeID,employeename,enrollmentdate,salary,job,sex,departmentID)

joinn(employeeID,projectID)

project(projectID,projectname,budget,estimatedrevenue,startdate,estimatedenddate,officeID,employeeID)

department(departmentID,departmentname,officeID,employeeID)

branchoffice(officeID,officename,officelocation)

c）用SQL语句创建关系模式

create table branchoffice(

officeID int primary key,

officename char(20),

officelocation char(256) );

create table employee(

employeeID int primary key,

employname char(20) not null,

enrollmentdate char(20),

salary float,

job char(40),

sex byte,

departmentID int);

create table department(

departmentID int primary key,

departname char(40),

officeID int,

employeeID int,

foreign key (employeeID) references employee(employeeID) on update cascade on delete cascade);

create table joinn(

employeeID int,

projectID int,

primary key (employeeID,projectID),

foreign key (employeeID) references employee(employeeID) on update cascade on delete cascade));

create table technicist(

techlevel smallint,

category char(20),

employeeID int primary key,

foreign key (employeeID) references employee(employeeID) on update cascade on delete cascade);

create table management(

employeeID int primary key,

managelevel smallint,

boss integer,

foreign key (boss) references department(departmentID) on update cascade on delete cascade,

foreign key (employeeID) references employee(employeeID) on update cascade on delete cascade);

create table project(

projectID int primary key,

resoffice int,

principle int,

projectname char(40),

budget int,

estimatedrevenue int,

startdate char(20),

enddate char(20),

foreign key (resoffice) references branchoffice(officeID) on update cascade on delete cascade,

foreign key(principle) references employee(employeeID) on update cascade on delete cascade);

alter table employee add foreign key(departmentID) references department(departmentID) on update cascade on delete cascade;

alter table department add foreign key(officeID) reference branchoffice(officeID) on update cascade on delete cascade;

alter table joinn add foreign key(projectID) reference project(projectID) on update cascade on delete cascade;

d）查询语句样例

单表查询：查询sex=0的雇员

select employeeID

from employee

where sex=0;

多表连接查询：查询参加项目的雇员，参加多个项目的雇员可能返回多次

select employeename

from employee,joinn

where joinn.employeeID=employee.enployeeID;  
多表嵌套查询：同上，一个雇员返回一次

select employeename

from employee

Where employeeID in(select employeeID from joinn);   
EXISTS查询：同上

select employeename

from employee

where exists (select \* from joinn where employee.employeeID=joinn.employeeID);

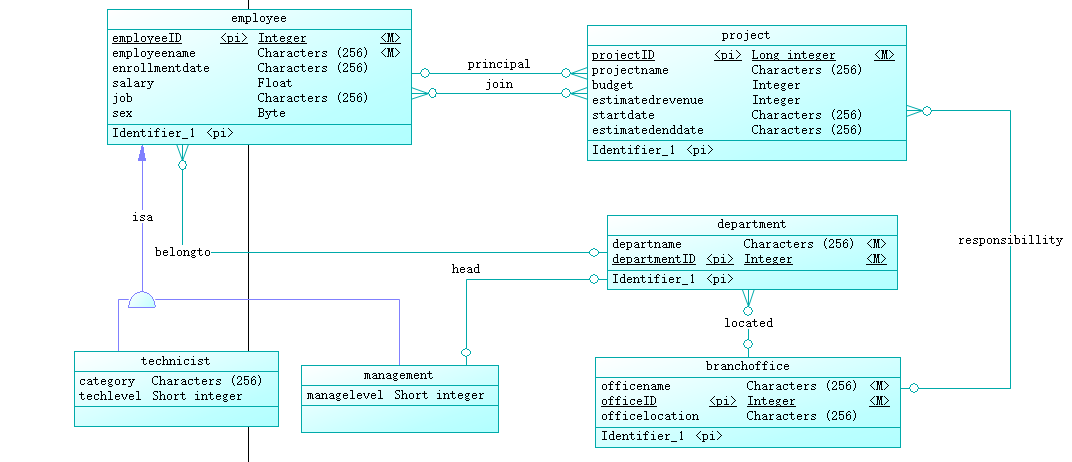
聚合操作查询：查找ID111的雇员加入了几个项目

select count(distinct projectID)

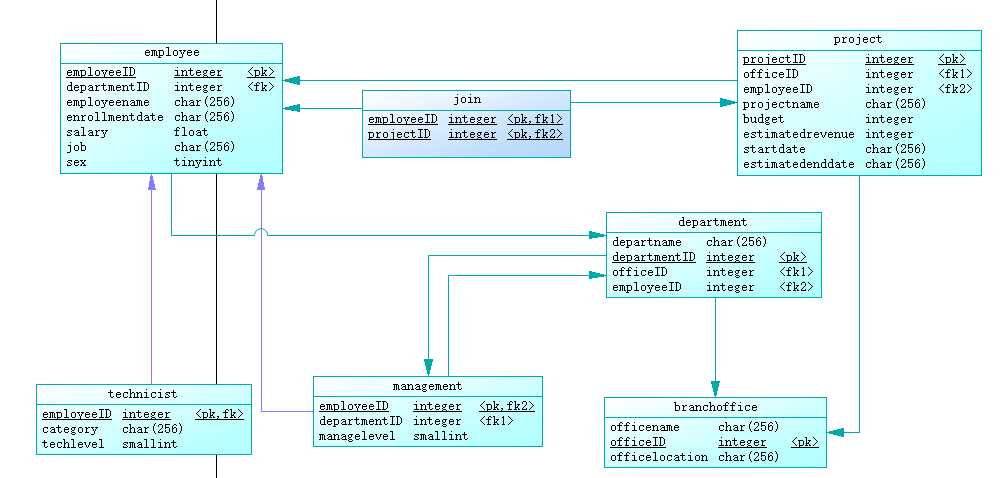
from joinn

Where employeeID=111;

3.a）该领域的ER图如下



b）该领域的关系模型图如下



C）生成创建数据库语句如下

if exists(select 1 from sys.sysforeignkey where role='FK\_DEPARTME\_HEAD\_MANAGEME') then

alter table department

delete foreign key FK\_DEPARTME\_HEAD\_MANAGEME

end if;

if exists(select 1 from sys.sysforeignkey where role='FK\_DEPARTME\_LOCATED\_BRANCHOF') then

alter table department

delete foreign key FK\_DEPARTME\_LOCATED\_BRANCHOF

end if;

if exists(select 1 from sys.sysforeignkey where role='FK\_EMPLOYEE\_BELONGTO\_DEPARTME') then

alter table employee

delete foreign key FK\_EMPLOYEE\_BELONGTO\_DEPARTME

end if;

if exists(select 1 from sys.sysforeignkey where role='FK\_joinn\_joinn\_EMPLOYEE') then

alter table "joinn"

delete foreign key FK\_joinn\_joinn\_EMPLOYEE

end if;

if exists(select 1 from sys.sysforeignkey where role='FK\_joinn\_joinn2\_PROJECT') then

alter table "joinn"

delete foreign key FK\_joinn\_joinn2\_PROJECT

end if;

if exists(select 1 from sys.sysforeignkey where role='FK\_MANAGEME\_HEAD2\_DEPARTME') then

alter table management

delete foreign key FK\_MANAGEME\_HEAD2\_DEPARTME

end if;

if exists(select 1 from sys.sysforeignkey where role='FK\_MANAGEME\_ISA2\_EMPLOYEE') then

alter table management

delete foreign key FK\_MANAGEME\_ISA2\_EMPLOYEE

end if;

if exists(select 1 from sys.sysforeignkey where role='FK\_PROJECT\_PRINCIPAL\_EMPLOYEE') then

alter table project

delete foreign key FK\_PROJECT\_PRINCIPAL\_EMPLOYEE

end if;

if exists(select 1 from sys.sysforeignkey where role='FK\_PROJECT\_RESPONSIB\_BRANCHOF') then

alter table project

delete foreign key FK\_PROJECT\_RESPONSIB\_BRANCHOF

end if;

if exists(select 1 from sys.sysforeignkey where role='FK\_TECHNICI\_ISA\_EMPLOYEE') then

alter table technicist

delete foreign key FK\_TECHNICI\_ISA\_EMPLOYEE

end if;

drop index if exists branchoffice.branchoffice\_PK;

drop table if exists branchoffice;

drop index if exists department.head\_FK;

drop index if exists department.located\_FK;

drop index if exists department.department\_PK;

drop table if exists department;

drop index if exists employee.belongto\_FK;

drop index if exists employee.employee\_PK;

drop table if exists employee;

drop index if exists "joinn".joinn2\_FK;

drop index if exists "joinn".joinn\_FK;

drop index if exists "joinn".joinn\_PK;

drop table if exists "joinn";

drop index if exists management.head2\_FK;

drop index if exists management.management\_PK;

drop table if exists management;

drop index if exists project.principal\_FK;

drop index if exists project.responsibillity\_FK;

drop index if exists project.project\_PK;

drop table if exists project;

drop index if exists technicist.technicist\_PK;

drop table if exists technicist;

/\*==============================================================\*/

/\* Table: branchoffice \*/

/\*==============================================================\*/

create table branchoffice

(

officename char(256) not null,

officeID integer not null,

officelocation char(256) null,

constraint PK\_BRANCHOFFICE primary key (officeID)

);

/\*==============================================================\*/

/\* Index: branchoffice\_PK \*/

/\*==============================================================\*/

create unique index branchoffice\_PK on branchoffice (

officeID ASC

);

/\*==============================================================\*/

/\* Table: department \*/

/\*==============================================================\*/

create table department

(

departname char(256) not null,

departmentID integer not null,

officeID integer null,

employeeID integer null,

constraint PK\_DEPARTMENT primary key (departmentID)

);

/\*==============================================================\*/

/\* Index: department\_PK \*/

/\*==============================================================\*/

create unique index department\_PK on department (

departmentID ASC

);

/\*==============================================================\*/

/\* Index: located\_FK \*/

/\*==============================================================\*/

create index located\_FK on department (

officeID ASC

);

/\*==============================================================\*/

/\* Index: head\_FK \*/

/\*==============================================================\*/

create index head\_FK on department (

employeeID ASC

);

/\*==============================================================\*/

/\* Table: employee \*/

/\*==============================================================\*/

create table employee

(

employeeID integer not null,

departmentID integer null,

employeename char(256) not null,

enrollmentdate char(256) null,

salary float null,

job char(256) null,

sex tinyint null,

constraint PK\_EMPLOYEE primary key (employeeID)

);

/\*==============================================================\*/

/\* Index: employee\_PK \*/

/\*==============================================================\*/

create unique index employee\_PK on employee (

employeeID ASC

);

/\*==============================================================\*/

/\* Index: belongto\_FK \*/

/\*==============================================================\*/

create index belongto\_FK on employee (

departmentID ASC

);

/\*==============================================================\*/

/\* Table: "joinn" \*/

/\*==============================================================\*/

create table "joinn"

(

employeeID integer not null,

projectID integer not null,

constraint PK\_joinn primary key clustered (employeeID, projectID)

);

/\*==============================================================\*/

/\* Index: joinn\_PK \*/

/\*==============================================================\*/

create unique clustered index joinn\_PK on "joinn" (

employeeID ASC,

projectID ASC

);

/\*==============================================================\*/

/\* Index: joinn\_FK \*/

/\*==============================================================\*/

create index joinn\_FK on "joinn" (

employeeID ASC

);

/\*==============================================================\*/

/\* Index: joinn2\_FK \*/

/\*==============================================================\*/

create index joinn2\_FK on "joinn" (

projectID ASC

);

/\*==============================================================\*/

/\* Table: management \*/

/\*==============================================================\*/

create table management

(

employeeID integer not null,

departmentID integer null,

managelevel smallint null,

constraint PK\_MANAGEMENT primary key clustered (employeeID)

);

/\*==============================================================\*/

/\* Index: management\_PK \*/

/\*==============================================================\*/

create unique clustered index management\_PK on management (

employeeID ASC

);

/\*==============================================================\*/

/\* Index: head2\_FK \*/

/\*==============================================================\*/

create index head2\_FK on management (

departmentID ASC

);

/\*==============================================================\*/

/\* Table: project \*/

/\*==============================================================\*/

create table project

(

projectID integer not null,

officeID integer null,

employeeID integer null,

projectname char(256) null,

budget integer null,

estimatedrevenue integer null,

startdate char(256) null,

estimatedenddate char(256) null,

constraint PK\_PROJECT primary key (projectID)

);

/\*==============================================================\*/

/\* Index: project\_PK \*/

/\*==============================================================\*/

create unique index project\_PK on project (

projectID ASC

);

/\*==============================================================\*/

/\* Index: responsibillity\_FK \*/

/\*==============================================================\*/

create index responsibillity\_FK on project (

officeID ASC

);

/\*==============================================================\*/

/\* Index: principal\_FK \*/

/\*==============================================================\*/

create index principal\_FK on project (

employeeID ASC

);

/\*==============================================================\*/

/\* Table: technicist \*/

/\*==============================================================\*/

create table technicist

(

employeeID integer not null,

category char(256) null,

techlevel smallint null,

constraint PK\_TECHNICIST primary key clustered (employeeID)

);

/\*==============================================================\*/

/\* Index: technicist\_PK \*/

/\*==============================================================\*/

create unique clustered index technicist\_PK on technicist (

employeeID ASC

);

alter table department

add constraint FK\_DEPARTME\_HEAD\_MANAGEME foreign key (employeeID)

references management (employeeID)

on update restrict

on delete restrict;

alter table department

add constraint FK\_DEPARTME\_LOCATED\_BRANCHOF foreign key (officeID)

references branchoffice (officeID)

on update restrict

on delete restrict;

alter table employee

add constraint FK\_EMPLOYEE\_BELONGTO\_DEPARTME foreign key (departmentID)

references department (departmentID)

on update restrict

on delete restrict;

alter table "joinn"

add constraint FK\_joinn\_joinn\_EMPLOYEE foreign key (employeeID)

references employee (employeeID)

on update restrict

on delete restrict;

alter table "joinn"

add constraint FK\_joinn\_joinn2\_PROJECT foreign key (projectID)

references project (projectID)

on update restrict

on delete restrict;

alter table management

add constraint FK\_MANAGEME\_HEAD2\_DEPARTME foreign key (departmentID)

references department (departmentID)

on update restrict

on delete restrict;

alter table management

add constraint FK\_MANAGEME\_ISA2\_EMPLOYEE foreign key (employeeID)

references employee (employeeID)

on update restrict

on delete restrict;

alter table project

add constraint FK\_PROJECT\_PRINCIPAL\_EMPLOYEE foreign key (employeeID)

references employee (employeeID)

on update restrict

on delete restrict;

alter table project

add constraint FK\_PROJECT\_RESPONSIB\_BRANCHOF foreign key (officeID)

references branchoffice (officeID)

on update restrict

on delete restrict;

alter table technicist

add constraint FK\_TECHNICI\_ISA\_EMPLOYEE foreign key (employeeID)

references employee (employeeID)

on update restrict

on delete restrict;

4.a）两种关系模式差异不大，但是一对一一对多的关系，我设计的表可以给外键约束的变量和被约束的变量起不同的变量名，更贴近实际使用需求，PowerDesigner只能起相同的变量名

b）powerdesigner生成的语句注释较多，逻辑清晰，便于使用者阅读和修改。同时，powerdesigner生成外键约束大多在类外实现，以防在被约束表未定义的时候发生异常。powerdisigner还会为表创建索引，加快查找效率。powerdesigner会为每个语句说明是否可以为空值。同时，powerdesigner在创建表之前会将可能存在的已有表删除，保证程序正确执行。