数据库系统作业

作者：Yu Xiang You

学号：2312900

专业：计算机科学与技术

提交日期：2025 年 4 月 11 日

目录

1 设计描述

电影作为一门综合性艺术，蕴含着丰富的文化内涵与艺术价值。随着电影行业的蓬勃发展，与之相关的信息呈现出海量且繁杂的特征，对这些信息进行高效存储与管理的需求日益凸显。基于此，我计划设计并实现一个专门的电影数据库。

我个人对电影怀有浓厚的兴趣，闲暇时热衷于沉浸在电影的世界中，探寻导演精心设置的情节伏笔，品味独特的镜头语言。出于这份热爱，我期望以电影为主题构建数据库，并将其作为本学期课程大作业。以兴趣为驱动，我相信能尽己所能高质量地完成此项任务。

该数据库涵盖多个关键实体，包括电影、演员、导演、奖项、制片公司、发行公司以及系列电影。实体间存在着多样的关系，如电影与电影系列、奖项与导演、奖项与电影、演员与电影、导演与电影等。此外，我还希望将其与本学期选修的《信息检索》课程相结合，使数据库具备强大的复杂查询功能，成为一个专业的电影知识宝库。

2 自己实现部分

2.1 自制 ER 图

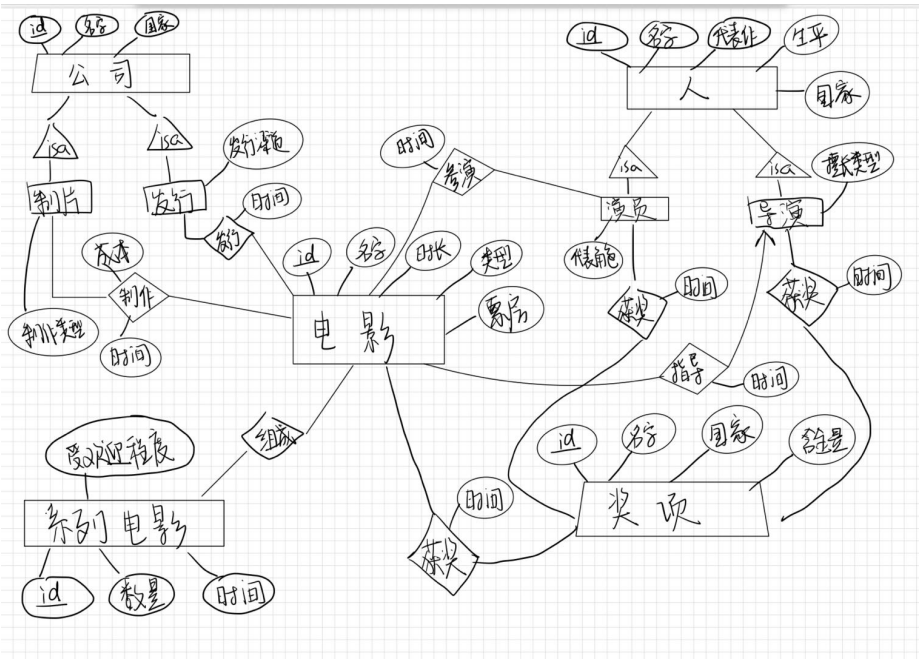

图 1: 自制 ER 图

2.2 ER 转为关系模式

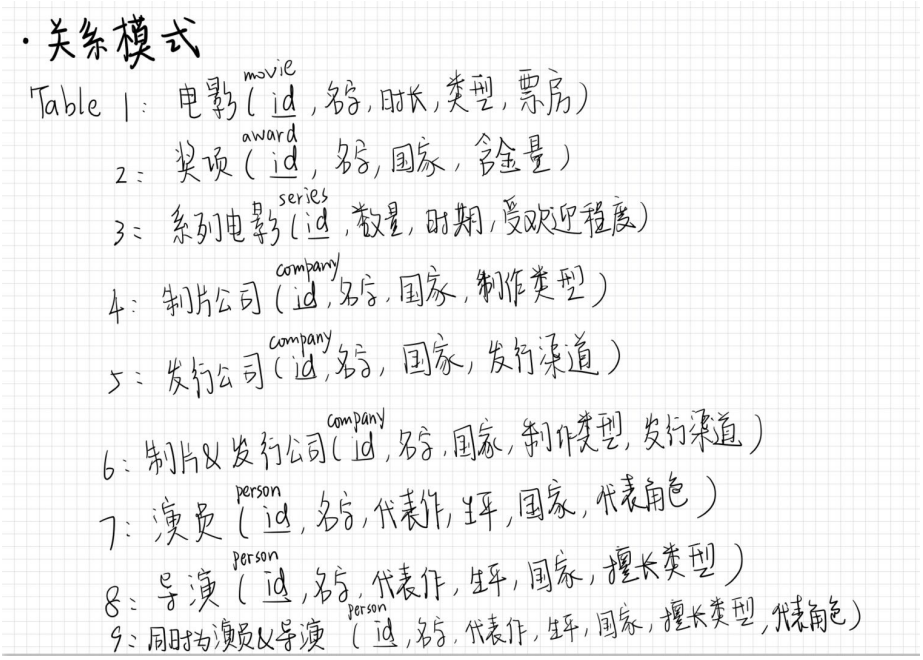

图 2: ER 转为关系模式:Entity 部分

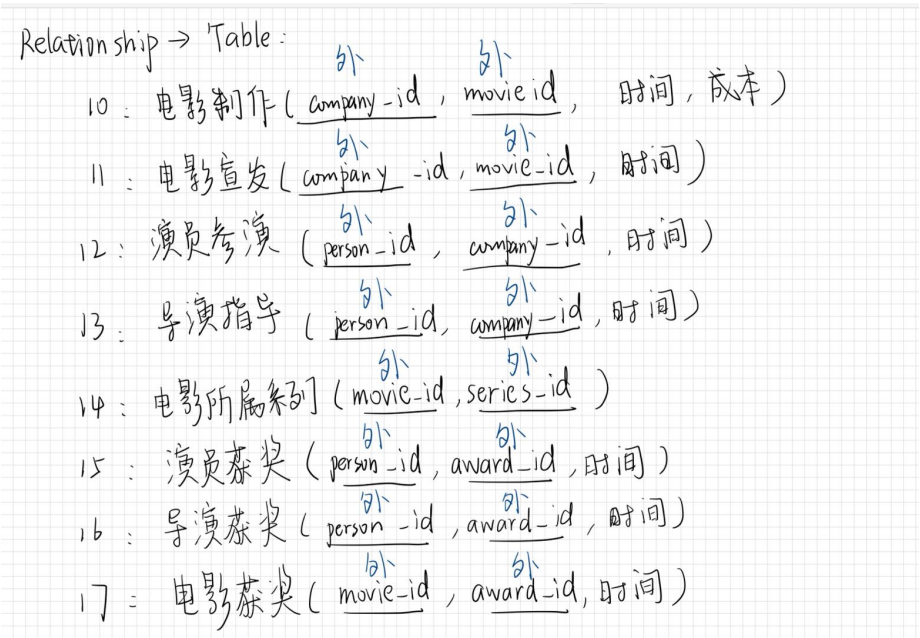

图 3: ER 转为关系模式:Relation 部分

2.3 用 MySQL 实现关系模式转换

```sql
# People
create table People (
    People_id             int not null,
    People_name           char(256),
    People_country        char(256),
    People_masterpiece    char(256),
    People_briefintro     char(256),
    primary key (People_id)
);

# Actor
create table Actor (
    People_id             int not null,
    People_name           char(256),
    People_country        char(256),
    People_masterpiece    char(256),
```

```
    People_briefintro    char(256),
    Actor_masterrole     char(256),
    primary key (People_id)
);

# Movie
create table Movie (
    Movie_id             int not null,
    Movie_name           char(256),
    Movie_length         smallint,
    Movie_boxoffice      INT,
    Movie_type           char(256),
    primary key (Movie_id)
);

# Actor_And_Movie
create table Actor_And_Movie (
    People_id            int not null,
    Movie_id             int not null,
    time                 datetime,
    primary key (People_id, Movie_id)
);

# Award
create table Award (
    Award_id             int not null,
    Award_name           char(256),
    Award_country        char(256),
    Award_Ranking        char(256),
    primary key (Award_id)
);

# Actor_Win_Award
create table Actor_Win_Award (
    People_id            int not null,
    Award_id             int not null,
    time                 datetime,
    primary key (People_id, Award_id)
);
```

```
56
57 # Company
58 create table Company (
59    Company_id           int not null,
60    Company_name         char(256),
61    Company_country      char(256),
62    primary key (Company_id)
63 );
64
65 # Director
66 create table Director (
67    People_id            int not null,
68    People_name          char(256),
69    People_country       char(256),
70    People_masterpiece   char(256),
71    People_briefintro    char(256),
72    Director_type        char(256),
73    primary key (People_id)
74 );
75
76 # Director_And_Movie
77 create table Director_And_Movie (
78    People_id            int not null,
79    Movie_id             int not null,
80    time                 datetime,
81    primary key (People_id, Movie_id)
82 );
83
84 # Director_Win_Award
85 create table Director_Win_Award (
86    Award_id             int not null,
87    People_id            int not null,
88    time                 datetime,
89    primary key (Award_id, People_id)
90 );
91
92 # Distribution_Company
93 create table Distribution_Company (
94    Company_id           int not null,
```

```
    Company_name          char(256),
    Company_country       char(256),
    Distribution_Company_Way char(256),
    primary key (Company_id)
);

# Series_Movie
create table Series_Movie (
    Series_Movie_id       int not null,
    Series_Movie_number   smallint,
    Series_Movie_time     datetime,
    Series_Movie_popularity char(256),
    primary key (Series_Movie_id)
);

# Movie_Belongs_To_Series_Movies
create table Movie_Belongs_To_Series_Movies (
    Series_Movie_id       int not null,
    Movie_id              int not null,
    primary key (Series_Movie_id, Movie_id)
);

# Movie_Distributed_By_Distribution_Company
create table Movie_Distributed_By_Distribution_Company (
    Company_id            int not null,
    Movie_id              int not null,
    time                  datetime,
    primary key (Company_id, Movie_id)
);

# Production_Company
create table Production_Company (
    Company_id            int not null,
    Company_name          char(256),
    Company_country       char(256),
    Production_Company_type char(256),
    primary key (Company_id)
);

```

```
# Movie_Produced_By_Production_Company
create table Movie_Produced_By_Production_Company (
   Company_id           int not null,
   Movie_id             int not null,
   time                 datetime,
   cost                 int,
   primary key (Company_id, Movie_id)
);

# Movies_Win_Award
create table Movies_Win_Award (
   Movie_id             int not null,
   Award_id             int not null,
   time                 datetime,
   primary key (Movie_id, Award_id)
);
```

2.4 查询示例

1. 单表的查询

```
# 查询需求：从Actor表查询小李子（Leonardo_DiCaprio）的国家
SELECT People_country
FROM Actor
WHERE People_name = 'Leonardo_DiCaprio'
```

2. 多表连接查询

```
# 查询需求：查询所有国籍为USA的公司和演员的名字
SELECT People_name,Company_name
FROM Movie,People
WHERE People_country = 'USA' or Company_country = 'USA'
```

3. 多表嵌套查询

```
# 查询需求：查询小李子（Leonardo_DiCaprio）的代表作的票房
SELECT Movie_boxoffice
FROM Movie
WHERE Movie_name =
 (SELECT People_masterpiece
```

```
  FROM Actor
  WHERE People_name = 'Leonardo_DiCaprio'
)
```

4.EXISTS 查询

```
# 查询所有至少获得过一个奖的演员
SELECT People_name
FROM Actor
WHERE EXISTS (
    SELECT Actor_Win_Award.People_id
    FROM Actor_Win_Award
    WHERE Actor_Win_Award.People_id = Actor.People_id
)
```

5. 聚合操作查询

```
# 查询每个制片公司的制片数量
SELECT Company_name, COUNT(Movie_id)
FROM Company, Movie
WHERE Company_id = Company_id
GROUP BY Company_name;
```

3 PowerDesigner 实现部分

3.1 PowerDesigner 实现 ER 图

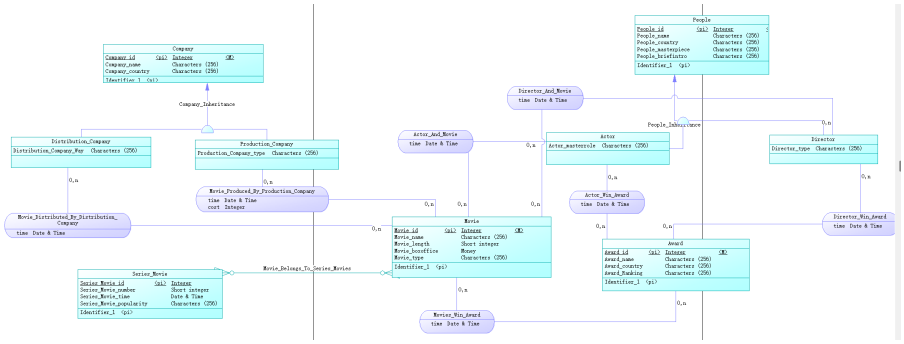

图 4: PowerDesign 绘制 ER 图

3.2 PowerDesigner 转为关系模式

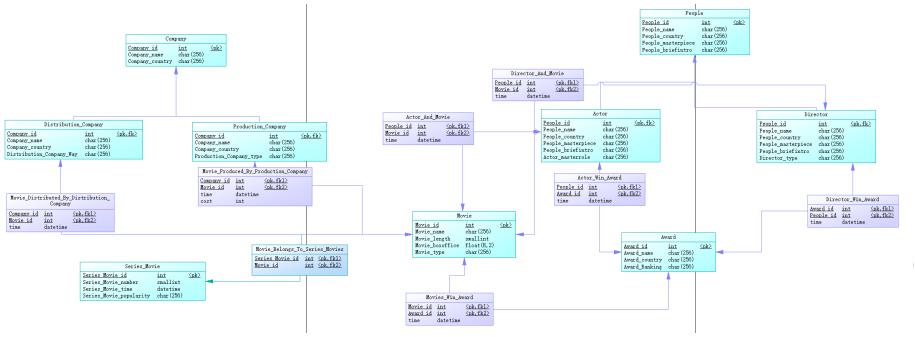

图 5: PowerDesign 转换为关系模式

3.3 PowerDesigner 生成表的 MySQL 语句

```
/*
    ==============================================================
    */
/* DBMS name:      MySQL 5.0                                    */
/* Created on:     2025/4/11 12:04:49                           */
/*
    ==============================================================
    */


drop table if exists Actor;

drop table if exists Actor_And_Movie;

drop table if exists Actor_Win_Award;

drop table if exists Award;

drop table if exists Company;

```

```
drop table if exists Director;

drop table if exists Director_And_Movie;

drop table if exists Director_Win_Award;

drop table if exists Distribution_Company;

drop table if exists Movie;

drop table if exists Movie_Belongs_To_Series_Movies;

drop table if exists Movie_Distributed_By_Distribution_Company;

drop table if exists Movie_Produced_By_Production_Company;

drop table if exists Movies_Win_Award;

drop table if exists People;

drop table if exists Production_Company;

drop table if exists Series_Movie;

/*==========================*/
/* Table: People                                                */
/*==========================*/
create table People
(
   People_id            int not null,
   People_name          char(256),
   People_country       char(256),
   People_masterpiece   char(256),
   People_briefintro    char(256),
   primary key (People_id)
);

/*==========================*/
```

```
/* Table: Actor                                                 */
/*==========================*/
create table Actor
(
   People_id            int not null,
   People_name          char(256),
   People_country       char(256),
   People_masterpiece   char(256),
   People_briefintro    char(256),
   Actor_masterrole     char(256),
   primary key (People_id),
   constraint FK_People_Inheritance foreign key (People_id)
      references People (People_id) on delete restrict on
          update restrict
);

/*==========================*/
/* Table: Movie                                                 */
/*==========================*/
create table Movie
(
   Movie_id             int not null,
   Movie_name           char(256),
   Movie_length         smallint,
   Movie_boxoffice      float(8,2),
   Movie_type           char(256),
   primary key (Movie_id)
);

/*==========================*/
/* Table: Actor_And_Movie                                       */
/*==========================*/
create table Actor_And_Movie
(
   People_id            int not null,
   Movie_id             int not null,
```

```
90     time                 datetime,
91     primary key (People_id, Movie_id),
92     constraint FK_Actor_And_Movie foreign key (People_id)
93        references Actor (People_id) on delete restrict on update
              restrict,
94     constraint FK_Actor_And_Movie2 foreign key (Movie_id)
95        references Movie (Movie_id) on delete restrict on update
              restrict
96  );
97
98  /*==========================*/
99  /* Table: Award                                                 */
100 /*==========================*/
101 create table Award
102 (
103    Award_id             int not null,
104    Award_name           char(256),
105    Award_country        char(256),
106    Award_Ranking        char(256),
107    primary key (Award_id)
108 );
109
110 /*==========================*/
111 /* Table: Actor_Win_Award                                       */
112 /*==========================*/
113 create table Actor_Win_Award
114 (
115    People_id            int not null,
116    Award_id             int not null,
117    time                 datetime,
118    primary key (People_id, Award_id),
119    constraint FK_Actor_Win_Award foreign key (People_id)
120       references Actor (People_id) on delete restrict on update
              restrict,
121    constraint FK_Actor_Win_Award2 foreign key (Award_id)
122       references Award (Award_id) on delete restrict on update
              restrict
```

```
);

/*==========================*/
/* Table: Company                                                */
/*==========================*/
create table Company
(
   Company_id           int not null,
   Company_name         char(256),
   Company_country      char(256),
   primary key (Company_id)
);

/*==========================*/
/* Table: Director                                               */
/*==========================*/
create table Director
(
   People_id            int not null,
   People_name          char(256),
   People_country       char(256),
   People_masterpiece   char(256),
   People_briefintro    char(256),
   Director_type        char(256),
   primary key (People_id),
   constraint FK_People_Inheritance2 foreign key (People_id)
      references People (People_id) on delete restrict on
          update restrict
);

/*==========================*/
/* Table: Director_And_Movie                                     */
/*==========================*/
create table Director_And_Movie
(
   People_id            int not null,
```

```
158     Movie_id              int not null,
159     time                  datetime,
160     primary key (People_id, Movie_id),
161     constraint FK_Director_And_Movie foreign key (People_id)
162        references Director (People_id) on delete restrict on
               update restrict,
163     constraint FK_Director_And_Movie2 foreign key (Movie_id)
164        references Movie (Movie_id) on delete restrict on update
               restrict
165 );
166
167 /*==========================*/
168 /* Table: Director_Win_Award
                                        */
169 /*==========================*/
170 create table Director_Win_Award
171 (
172     Award_id              int not null,
173     People_id             int not null,
174     time                  datetime,
175     primary key (Award_id, People_id),
176     constraint FK_Director_Win_Award foreign key (Award_id)
177        references Award (Award_id) on delete restrict on update
               restrict,
178     constraint FK_Director_Win_Award2 foreign key (People_id)
179        references Director (People_id) on delete restrict on
               update restrict
180 );
181
182 /*==========================*/
183 /* Table: Distribution_Company
                                      */
184 /*==========================*/
185 create table Distribution_Company
186 (
187     Company_id            int not null,
188     Company_name          char(256),
189     Company_country       char(256),
190     Distribution_Company_Way char(256),
```

```
191     primary key (Company_id),
192     constraint FK_Company_Inheritance foreign key (Company_id)
193        references Company (Company_id) on delete restrict on
               update restrict
194 );
195
196 /*==========================*/
197 /* Table: Series_Movie
                                                    */
198 /*==========================*/
199 create table Series_Movie
200 (
201     Series_Movie_id      int not null,
202     Series_Movie_number  smallint,
203     Series_Movie_time    datetime,
204     Series_Movie_popularity char(256),
205     primary key (Series_Movie_id)
206 );
207
208 /*==========================*/
209 /* Table: Movie_Belongs_To_Series_Movies
                              */
210 /*==========================*/
211 create table Movie_Belongs_To_Series_Movies
212 (
213     Series_Movie_id      int not null,
214     Movie_id             int not null,
215     primary key (Series_Movie_id, Movie_id),
216     constraint FK_Movie_Belongs_To_Series_Movies foreign key (
           Series_Movie_id)
217        references Series_Movie (Series_Movie_id) on delete
               restrict on update restrict,
218     constraint FK_Movie_Belongs_To_Series_Movies2 foreign key (
           Movie_id)
219        references Movie (Movie_id) on delete restrict on update
               restrict
220 );
221
222 /*==========================*/
```

```
/* Table: Movie_Distributed_By_Distribution_Company
                */
/*==========================*/
create table Movie_Distributed_By_Distribution_Company
(
   Company_id           int not null,
   Movie_id             int not null,
   time                 datetime,
   primary key (Company_id, Movie_id),
   constraint FK_Movie_Distributed_By_Distribution_Company
       foreign key (Company_id)
      references Distribution_Company (Company_id) on delete
          restrict on update restrict,
   constraint FK_Movie_Distributed_By_Distribution_Company2
       foreign key (Movie_id)
      references Movie (Movie_id) on delete restrict on update
          restrict
);

/*==========================*/
/* Table: Production_Company
                                      */
/*==========================*/
create table Production_Company
(
   Company_id           int not null,
   Company_name         char(256),
   Company_country      char(256),
   Production_Company_type char(256),
   primary key (Company_id),
   constraint FK_Company_Inheritance2 foreign key (Company_id)
      references Company (Company_id) on delete restrict on
          update restrict
);

/*==========================*/
/* Table: Movie_Produced_By_Production_Company
                     */
/*==========================*/
```

```
254 create table Movie_Produced_By_Production_Company
255 (
256    Company_id           int not null,
257    Movie_id             int not null,
258    time                 datetime,
259    cost                 int,
260    primary key (Company_id, Movie_id),
261    constraint FK_Movie_Produced_By_Production_Company foreign
           key (Company_id)
262       references Production_Company (Company_id) on delete
              restrict on update restrict,
263    constraint FK_Movie_Produced_By_Production_Company2 foreign
           key (Movie_id)
264       references Movie (Movie_id) on delete restrict on update
              restrict
265 );
266
267 /*==========================*/
268 /* Table: Movies_Win_Award
                                              */
269 /*==========================*/
270 create table Movies_Win_Award
271 (
272    Movie_id             int not null,
273    Award_id             int not null,
274    time                 datetime,
275    primary key (Movie_id, Award_id),
276    constraint FK_Movies_Win_Award foreign key (Movie_id)
277       references Movie (Movie_id) on delete restrict on update
              restrict,
278    constraint FK_Movies_Win_Award2 foreign key (Award_id)
279       references Award (Award_id) on delete restrict on update
              restrict
280 );
```

4 两种方式比较

4.1 Q1: 两种关系模式的设计是否有差距？若有，是否会对后续设计产生影响？

A1: 很明显两种关系模式的设计是存在差距的：

1.PowerDesigner 的从 ER 转换为关系模型是不能采取 OO 方式的，只有 NULL 方式和 ER 方式

2. 而我刚好选择了 OO 方式实现，这也就导致我的手绘的关系模型与 PowerDesigner 生成的效果不一样。

3. 后续的话我会将自己的 OO 方式实现转换为 ER 方式实现，以方便后续可能还需要用 PowerDesigner 来进一步完善自己的设计。

4.2 Q2:PowerDesginer 生成的 SQL 语句有什么特点？为什么会有附带语句？作用是？

A2:

1. 首先其会有 drop 语句以防止生成表格时出现错误（重名）

2. 建立表格时会有自动的注释，方便阅读

3. 带有 constraint 语句，类似"FK_Actor_And_Movie" 这个外键约束表明表中的 'Movie_id' 必须引用 'Movie' 表中的 'Movie_id'。也就是说，一个演员能参演的电影一定是已经存在了的电影。而像这样的语句能标明外键并更好地保证外键里的值只能是已经有的值

4. 带有"on delete strict" 和"on update strict" 语句："on delete strict" 语句能起到：当尝试删除父表中被引用的记录时，如果子表中有引用该记录的行，数据库会阻止删除操作，从而避免出现悬空引用的情况。而"on update strict" 能起到：当尝试更新父表中被引用的记录的主键值时，如果子表中有引用该记录的行，数据库会阻止更新操作，保证数据的一致性。