

# 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

<b>1 设计描述</b>	<b>2</b>
<b>2 自己实现部分</b>	<b>3</b>
2.1 自制 ER 图 . . . . .	3
2.2 ER 转为关系模式 . . . . .	4
2.3 用 MySQL 实现关系模式转换 . . . . .	5
2.4 查询示例 . . . . .	9
<b>3 PowerDesigner 实现部分</b>	<b>10</b>
3.1 PowerDesigner 实现 ER 图 . . . . .	10
3.2 PowerDesigner 转为关系模式 . . . . .	11
3.3 PowerDesigner 生成表的 MySQL 语句 . . . . .	11
<b>4 两种方式比较</b>	<b>20</b>
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? . . . . .	20
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? . . . . .	20

## 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

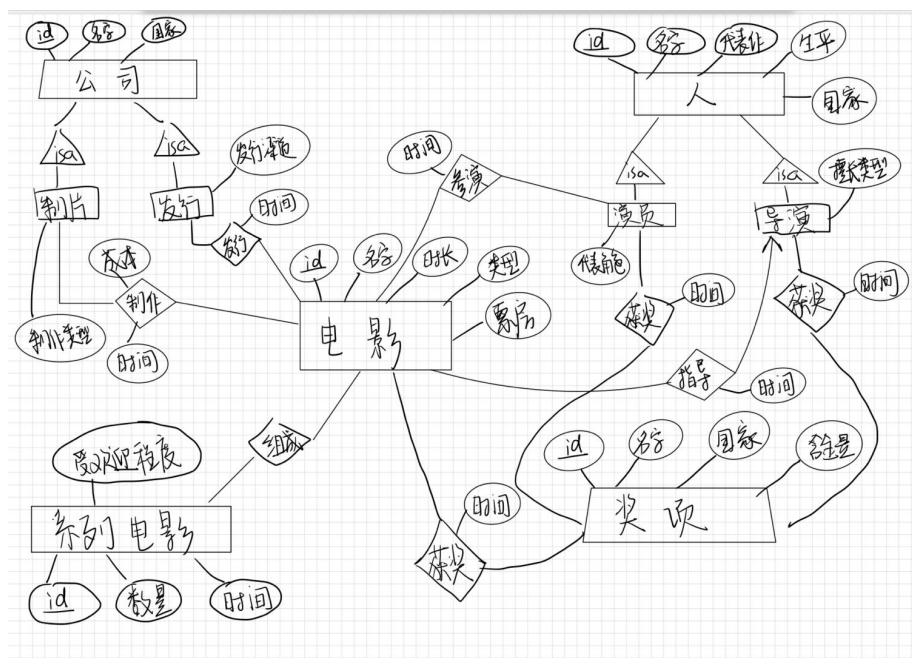


图 1: 自制 ER 图

## 2.2 ER 转为关系模式

### · 关系模式

- Table 1: 电影 (<sup>movie</sup>id, 名字, 时长, 类型, 票房)
- 2: 奖项 (<sup>award</sup>id, 名字, 国家, 含金量)
- 3: 系列电影 (<sup>series</sup>id, 数量, 时期, 受欢迎程度)
- 4: 制片公司 (<sup>company</sup>id, 名字, 国家, 制作类型)
- 5: 发行公司 (<sup>company</sup>id, 名字, 国家, 发行渠道)
- 6: 制片&发行公司 (<sup>company</sup>id, 名字, 国家, 制作类型, 发行渠道)
- 7: 演员 (<sup>person</sup>id, 名字, 代表作, 生平, 国家, 代表角色)
- 8: 导演 (<sup>person</sup>id, 名字, 代表作, 生平, 国家, 擅长类型)
- 9: 同时为演员&导演 (<sup>person</sup>id, 名字, 代表作, 生平, 国家, 擅长类型, 代表角色)

图 2: ER 转为关系模式:Entity 部分

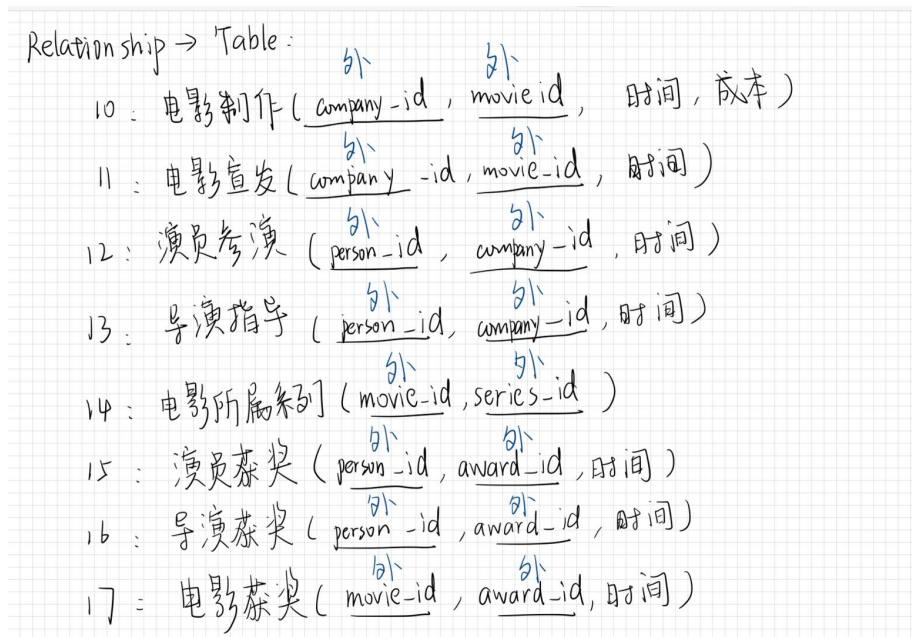


图 3: ER 转为关系模式:Relation 部分

## 2.3 用 MySQL 实现关系模式转换

```

1 # People
2 create table People (
3     People_id          int not null,
4     People_name        char(256),
5     People_country     char(256),
6     People_masterpiece char(256),
7     People_briefintro  char(256),
8     primary key (People_id)
9 );
10
11 # Actor
12 create table Actor (
13     People_id          int not null,
14     People_name        char(256),
15     People_country     char(256),
16     People_masterpiece char(256),

```

```

17     People_briefintro      char(256),
18     Actor_masterrole      char(256),
19     primary key (People_id)
20 );
21
22 # Movie
23 create table Movie (
24     Movie_id               int not null,
25     Movie_name             char(256),
26     Movie_length           smallint,
27     Movie_boxoffice        INT,
28     Movie_type             char(256),
29     primary key (Movie_id)
30 );
31
32 # Actor_And_Movie
33 create table Actor_And_Movie (
34     People_id              int not null,
35     Movie_id               int not null,
36     time                   datetime,
37     primary key (People_id, Movie_id)
38 );
39
40 # Award
41 create table Award (
42     Award_id               int not null,
43     Award_name             char(256),
44     Award_country          char(256),
45     Award_Ranking          char(256),
46     primary key (Award_id)
47 );
48
49 # Actor_Win_Award
50 create table Actor_Win_Award (
51     People_id              int not null,
52     Award_id               int not null,
53     time                   datetime,
54     primary key (People_id, Award_id)
55 );

```

```

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,

```



```

95     Company_name          char(256),
96     Company_country       char(256),
97     Distribution_Company_Way char(256),
98     primary key (Company_id)
99 );
100
101 # Series_Movie
102 create table Series_Movie (
103     Series_Movie_id        int not null,
104     Series_Movie_number    smallint,
105     Series_Movie_time      datetime,
106     Series_Movie_popularity char(256),
107     primary key (Series_Movie_id)
108 );
109
110 # Movie_Belongs_To_Series_Movies
111 create table Movie_Belongs_To_Series_Movies (
112     Series_Movie_id        int not null,
113     Movie_id               int not null,
114     primary key (Series_Movie_id, Movie_id)
115 );
116
117 # Movie_Distributed_By_Distribution_Company
118 create table Movie_Distributed_By_Distribution_Company (
119     Company_id             int not null,
120     Movie_id               int not null,
121     time                   datetime,
122     primary key (Company_id, Movie_id)
123 );
124
125 # Production_Company
126 create table Production_Company (
127     Company_id             int not null,
128     Company_name           char(256),
129     Company_country        char(256),
130     Production_Company_type char(256),
131     primary key (Company_id)
132 );
133

```

```

134 # Movie_Produced_By_Production_Company
135 create table Movie_Produced_By_Production_Company (
136     Company_id          int not null,
137     Movie_id            int not null,
138     time                datetime,
139     cost                 int,
140     primary key (Company_id, Movie_id)
141 );
142
143 # Movies_Win_Award
144 create table Movies_Win_Award (
145     Movie_id            int not null,
146     Award_id           int not null,
147     time               datetime,
148     primary key (Movie_id, Award_id)
149 );

```

## 2.4 查询示例

### 1. 单表的查询

```

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

```

### 2. 多表连接查询

```

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

```

### 3. 多表嵌套查询

```

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

```

```

6 FROM Actor
7 WHERE People_name = 'Leonardo_DiCaprio'
8 )

```

#### 4.EXISTS 查询

```

1 # 查询所有至少获得过一个奖的演员
2 SELECT People_name
3 FROM Actor
4 WHERE EXISTS (
5     SELECT Actor_Win_Award.People_id
6     FROM Actor_Win_Award
7     WHERE Actor_Win_Award.People_id = Actor.People_id
8 )

```

#### 5. 聚合操作查询

```

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

```

### 3 PowerDesigner 实现部分

#### 3.1 PowerDesigner 实现 ER 图

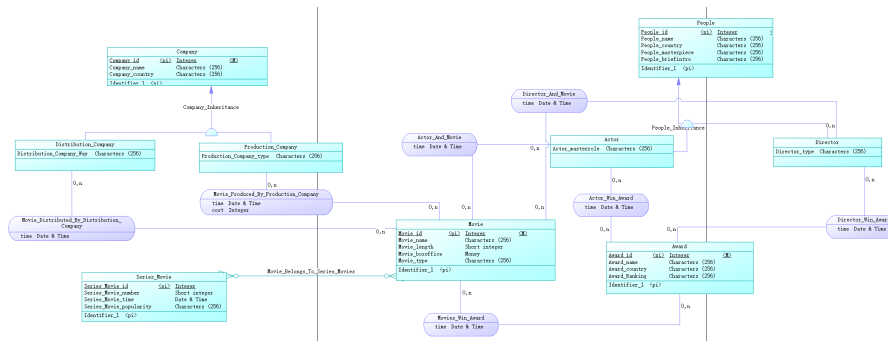


图 4: PowerDesign 绘制 ER 图

## 3.2 PowerDesigner 转为关系模式

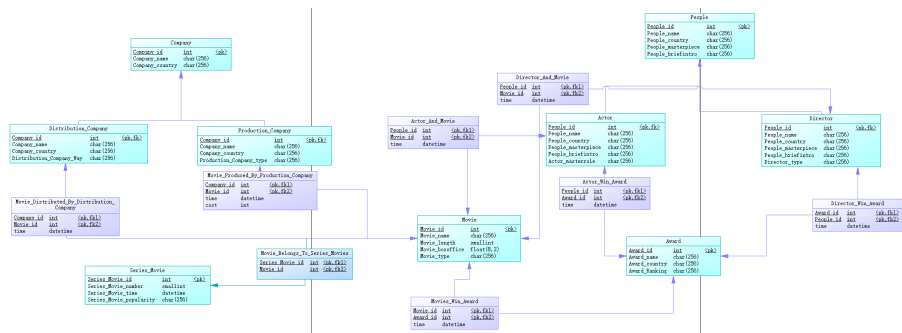


图 5: PowerDesign 转换为关系模式

## 3.3 PowerDesigner 生成表的 MySQL 语句

```

1  /*
2  /* DBMS name:      MySQL 5.0
3  /* Created on:      2025/4/11 12:04:49
4  /*
5
6
7  drop table if exists Actor;
8
9  drop table if exists Actor_And_Movie;
10
11 drop table if exists Actor_Win_Award;
12
13 drop table if exists Award;
14
15 drop table if exists Company;
16

```

```

17 drop table if exists Director;
18
19 drop table if exists Director_And_Movie;
20
21 drop table if exists Director_Win_Award;
22
23 drop table if exists Distribution_Company;
24
25 drop table if exists Movie;
26
27 drop table if exists Movie_Belongs_To_Series_Movies;
28
29 drop table if exists Movie_Distributed_By_Distribution_Company;
30
31 drop table if exists Movie_Produced_By_Production_Company;
32
33 drop table if exists Movies_Win_Award;
34
35 drop table if exists People;
36
37 drop table if exists Production_Company;
38
39 drop table if exists Series_Movie;
40
41 /*=====*/
42 /* Table: People                                     */
43 /*=====*/
44 create table People
45 (
46     People_id          int not null,
47     People_name         char(256),
48     People_country      char(256),
49     People_masterpiece  char(256),
50     People_briefintro   char(256),
51     primary key (People_id)
52 );
53
54 /*=====*/

```

```

55  /* Table: Actor
                                           */
56  /*=====*/
57  create table Actor
58  (
59      People_id          int not null,
60      People_name        char(256),
61      People_country     char(256),
62      People_masterpiece char(256),
63      People_briefintro  char(256),
64      Actor_masterrole   char(256),
65      primary key (People_id),
66      constraint FK_People_Inheritance foreign key (People_id)
67          references People (People_id) on delete restrict on
          update restrict
68  );
69
70  /*=====*/
71  /* Table: Movie
                                           */
72  /*=====*/
73  create table Movie
74  (
75      Movie_id           int not null,
76      Movie_name         char(256),
77      Movie_length       smallint,
78      Movie_boxoffice    float(8,2),
79      Movie_type         char(256),
80      primary key (Movie_id)
81  );
82
83  /*=====*/
84  /* Table: Actor_And_Movie
                                           */
85  /*=====*/
86  create table Actor_And_Movie
87  (
88      People_id          int not null,
89      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

```

```

123 );
124
125 /*=====*/
126 /* Table: Company
127                                     */
128 /*=====*/
129 create table Company
130 (
131     Company_id          int not null,
132     Company_name        char(256),
133     Company_country     char(256),
134     primary key (Company_id)
135 );
136
137 /*=====*/
138 /* Table: Director
139                                     */
140 /*=====*/
141 create table Director
142 (
143     People_id           int not null,
144     People_name         char(256),
145     People_country      char(256),
146     People_masterpiece  char(256),
147     People_briefintro   char(256),
148     Director_type       char(256),
149     primary key (People_id),
150     constraint FK_People_Inheritance2 foreign key (People_id)
151         references People (People_id) on delete restrict on
152         update restrict
153 );
154
155 /*=====*/
156 /* Table: Director_And_Movie
157                                     */
158 /*=====*/
159 create table Director_And_Movie
160 (
161     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 /*=====*/
171 create table Director_Win_Award
172 (
173     Award_id          int not null,
174     People_id         int not null,
175     time              datetime,
176     primary key (Award_id, People_id),
177     constraint FK_Director_Win_Award foreign key (Award_id)
178         references Award (Award_id) on delete restrict on update
            restrict,
179     constraint FK_Director_Win_Award2 foreign key (People_id)
180         references Director (People_id) on delete restrict on
            update restrict
181 );
182
183 /*=====*/
184 /* Table: Distribution_Company
185                                     */
186 /*=====*/
187 create table Distribution_Company
188 (
189     Company_id        int not null,
190     Company_name      char(256),
191     Company_country   char(256),
192     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 /*=====*/

```

```

223  /* Table: Movie_Distributed_By_Distribution_Company
      */
224  /*=====*/
225  create table Movie_Distributed_By_Distribution_Company
226  (
227      Company_id          int not null,
228      Movie_id            int not null,
229      time                datetime,
230      primary key (Company_id, Movie_id),
231      constraint FK_Movie_Distributed_By_Distribution_Company
          foreign key (Company_id)
232          references Distribution_Company (Company_id) on delete
              restrict on update restrict,
233      constraint FK_Movie_Distributed_By_Distribution_Company2
          foreign key (Movie_id)
234          references Movie (Movie_id) on delete restrict on update
              restrict
235  );
236
237  /*=====*/
238  /* Table: Production_Company
      */
239  /*=====*/
240  create table Production_Company
241  (
242      Company_id          int not null,
243      Company_name        char(256),
244      Company_country     char(256),
245      Production_Company_type char(256),
246      primary key (Company_id),
247      constraint FK_Company_Inheritance2 foreign key (Company_id)
          references Company (Company_id) on delete restrict on
248          update restrict
249  );
250
251  /*=====*/
252  /* Table: Movie_Produced_By_Production_Company
      */
253  /*=====*/

```

```

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: PowerDesigner 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。