

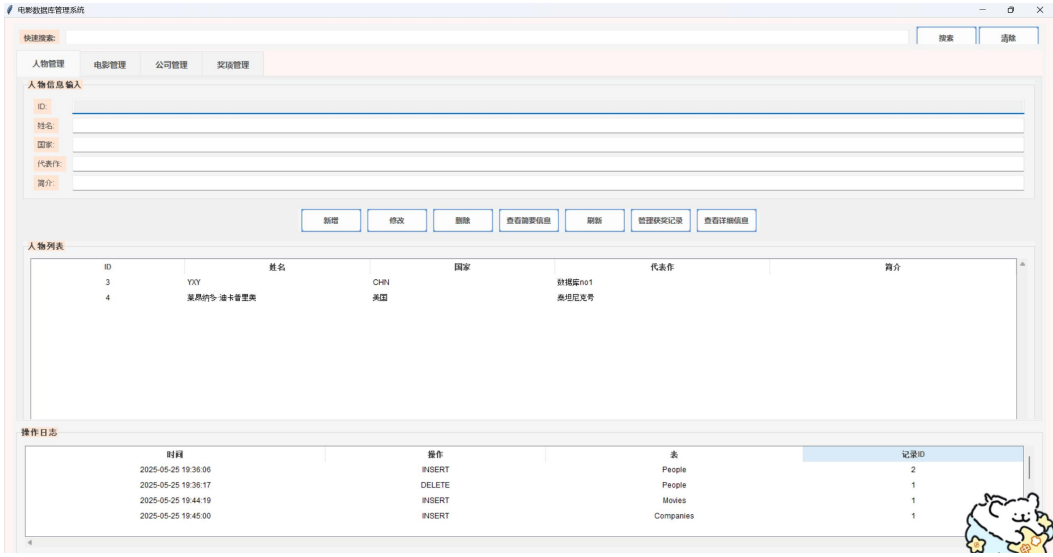
数据库工程作业

要求:

- 1. 完成一个小型的数据库信息管理系统（或部分功能），并填写工程作业报告；程序和报告请在规定时间之内上传。
- 2. 开发模式（B/S 或 C/S）、开发高级语言任选，后台数据库使用大型数据库管理系统（SQL Server、Oracle、MySQL 等），不要使用桌面数据库。
- 3. 报告中所列举的四种操作，每种操作举一个例子即可。
- 4. 作业成绩按照报告中的标准评分，程序只实现报告中涉及的部分即可。
- 5. 作业完成后，请将工程作业报告和程序打包提交给助教老师，并联系助教老师进行系统说明和演示，回答相关问题。

工程作业报告

1. 项目信息（10 分）

学号	2312900	姓名	禹相祐	专业	计算机科学与技术
项目名称	基于 Python+MySQL 的电影信息管理系统				
必备环境	MySQL + Python + Tkinter + Pillow + PyMySQL				
系统主要功能简介（4 分）	<div>1. 支持对于人物、电影、公司、奖项等四类实体的增删改查操作；</div> <div>2. 在“增”中使用触发器来记录日志；</div> <div>3. 在“删”中使用事务保证一致性；</div> <div>4. 在“改”中通过储存过程执行且用日志记录；</div> <div>5. 支持选中一条记录后，点击“查看简要信息”或“查看详细信息”获取有关该记录的简略信息或具体信息；</div> <div>6. 支持视图的查询和查看操作日志，包含时间、操作、操作表名、所操作的记录 id；</div>				
系统主要页面截图（6 分）	<div>1. 演员信息界面：</div> 				

2. 电影信息界面：

快速搜索:

搜索

清除

人物管理

电影管理

公司管理

奖项管理

电影信息输入

ID:

标题:

上映年份:

导演:

类型:

票房:

描述:

新增

修改

删除

查看详细信息

刷新

管理获奖记录

查看详细信息

管理演员

电影列表

ID	标题	年份	导演	类型	票房	描述
1	肖申克的救赎	1994	Director A	悬疑	10000000.00	
2	阿甘正传	1994	Director B	喜剧	10000000.00	
3	泰坦尼克号	1997	詹姆斯卡梅隆	灾难	2216000000.00	

操作日志

时间	操作	表	记录ID
2025-05-25 19:36:06	INSERT	People	2
2025-05-25 19:36:17	DELETE	People	1
2025-05-25 19:44:19	INSERT	Movies	1
2025-05-25 19:45:00	INSERT	Companies	1

3. 公司信息界面：

快速搜索:

搜索

清除

人物管理

电影管理

公司管理

奖项管理

公司信息输入

ID:

公司名称:

国家:

成立年份:

行业:

营收:

描述:

新增

修改

删除

查看详细信息

刷新

公司列表

ID	名称	国家	成立年份	行业	营收	描述
1	123	1234	123	1	2.00	3
2	二十世纪福斯	美国	1935	制片	10000000000.00	十大制片公司之一

操作日志

时间	操作	表	记录ID
2025-05-25 19:36:06	INSERT	People	2
2025-05-25 19:36:17	DELETE	People	1
2025-05-25 19:44:19	INSERT	Movies	1
2025-05-25 19:45:00	INSERT	Companies	1

4. 奖项信息界面：

快速搜索:

搜索

清除

人物管理

电影管理

公司管理

奖项管理

奖项信息输入

ID:

奖项名称:

类别:

年份:

描述:

新增

修改

删除

查看详细信息

刷新

奖项列表

ID	名称	类别	年份	描述
1	奥斯卡最佳男主	全球性	2020	
2	奥斯卡最佳影片	全球性	0	
3	奥斯卡最佳配乐	全球性	0	
4	金球奖最佳男主角	全球性	0	

操作日志

时间	操作	表	记录ID
2025-05-25 19:36:06	INSERT	People	2
2025-05-25 19:36:17	DELETE	People	1
2025-05-25 19:44:19	INSERT	Movies	1
2025-05-25 19:45:00	INSERT	Companies	1

2. 系统配置（10 分）

说明		(2 分) 请说明系统配置情况 (后台数据库, 高级语言); (8 分) 请使用连接串连接高级语言和数据库, 并分析字符串的各个部分。			
配置 步骤 2 分	DBMS	1. MySQL 5.0			
				
	高级 语言	1. Python 3.12			
				
连接串 分析 (6 分)		序 号	名 称	功 能 说 明	取 值
		1	host	MySQL 服 务 器 的 网 络 地 址。默 认 值 为 “localhost”	localhost
		2	user	登录的用户名, 用于身份的验证。	root
		3	password	登录密码	123456
		4	database	制定要操作的数据库名字	test01
		5	charset	定义所用的字符集	utf8mb4
		6	cursorclass	指定返回的查询结果的类型, 此处为 Python 的字典	DictCursor
连接串代码 (截屏) (2 分)		如图: 			
备注					

3. 数据库设计（14 分）

说明		（10 分）按照数据表的创建顺序，依次给出所涉及数据表的信息，其中参照字段以“（字段 1，字段 2，……，字段 n）”的形式给出，被参照字段以“表名（字段 1，字段 2，……，字段 n）”的形式给出； （4 分）一般 DBMS 都可以为数据库生成关系图，请将该图片截屏并粘贴到表格中。			
数据 表 （10）	创建 顺 序	数据表名称	主键	参照属性	被参照表及属性
	1	Movies	movie_id		

数 据 表 (续)	2	Companies	company_id		
	3	Awards	award_id		
	4	People	people_id		
	5	People_Awards	people_award_id	people_id	People(people_id)
				award_id	Awards(award_id)
	6	Movie_Awards	movie_award_id	movie_id	Movies(movie_id)
				award_id	Awards(award_id)
	7	Movie_Companies	movie_company_id	movie_id	Movies(movie_id)
				company_id	Companies(company_id)
	8	Movie_Actors	movie_actor_id	movie_id	Movies(movie_id)
				people_id	People(people_id)
	9	UI_Settings	setting_id		
	10	operation_logs	log_id		

关系
图 (4)

4. 含有事务应用的删除操作（13 分）

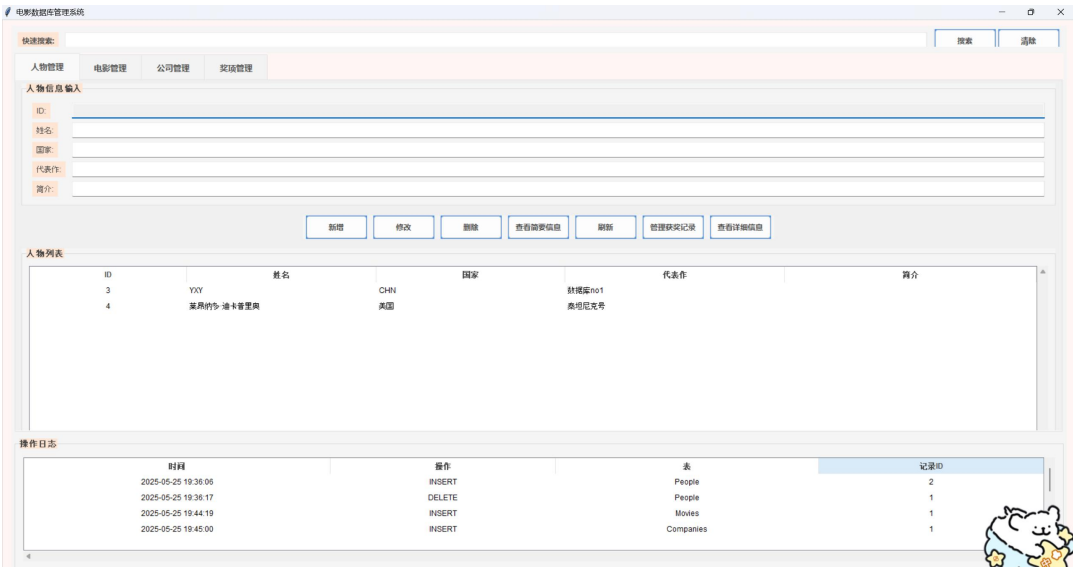
说明	<p>（1 分）简要说明该操作所要完成的功能；</p> <p>（2 分）该操作会涉及的表（必须含有两张或两张以上的关系表，同时以“表名”的形式给出）</p> <p>（1 分）表连接涉及字段描述（描述方式为“表 1. 属性=表 2. 属性”）</p> <p>（1 分）删除条件涉及的字段描述（以“表名. 属性=? ”形式给出）</p> <p>（4 分）实现该操作的关键代码（高级语言、SQL），截图即可；（其中如果删除语句中不包含任何形式的事务应用将扣除 3 分）</p> <p>（4 分）如何执行该操作，按所述方法能够正常演示程序则给分。</p>	
功能描述 （1 分）	<p>对“人物”和“电影”的删除操作均通过事务完成，以确保数据一致性与完整性。每次删除操作将包括两个步骤：</p> <ol style="list-style-type: none"> 1. 在 operation_logs 表中插入一条记录（记录删除的行为） 2. 从 People 或者 Movies 表中删除对应的记录 <p>整个过程在数据库事务控制下进行，若任何一步失败，将进行回滚，避免出现“日志已写但主数据未删”或“数据删了但日志丢失”的异常情况。</p>	
涉及的表 （2 分）	<p>删除人物时： People 表+operation_logs 表</p> <p>删除电影时： Movies 表+operation_logs 表</p>	
表连接涉及字段 （1 分）	<p>删除时不涉及多表的连接，只在事务中按顺序操作两张表，通过以下字段进行相连：</p> <p>People.people_id 对应 operation_logs.record_id</p> <p>Movies.movie_id 对应 operation_logs.record_id</p>	
删除条件字段描述 （1 分）	字段	规则
	People.people_id	当 people_id = 用户选中的指定 id 时进行删除
	Movies.movie_id	当 movie_id = 用户选中的指定 id 时进行删除
代码 （4 分）	<p>People 部分如下：</p>	

```
def delete_person_with_transaction(people_id): 1个用法
    conn = get_connection()
    try:
        conn.begin()
        with conn.cursor() as cursor:
            cursor.execute(query: "INSERT INTO operation_logs (operation_type, table_name, record_id) VALUES ('DELETE', 'People', %s)", args: (people_id,))
            cursor.execute(query: "DELETE FROM People WHERE people_id = %s", args: (people_id,))
        conn.commit()
    except Exception as e:
        conn.rollback()
        raise e
    finally:
        conn.close()
```

Movies 部分如下：

```
def delete_movie_with_transaction(movie_id): 1个用法
    conn = get_connection()
    try:
        conn.begin()
        with conn.cursor() as cursor:
            cursor.execute(query: "INSERT INTO operation_logs (operation_type, table_name, record_id) VALUES ('DELETE', 'Movies', %s)", args: (movie_id,))
            cursor.execute(query: "DELETE FROM Movies WHERE movie_id = %s", args: (movie_id,))
        conn.commit()
    except Exception as e:
        conn.rollback()
        raise e
    finally:
        conn.close()
```

我们将 People 表中的“yxy”一行删掉：
原先：

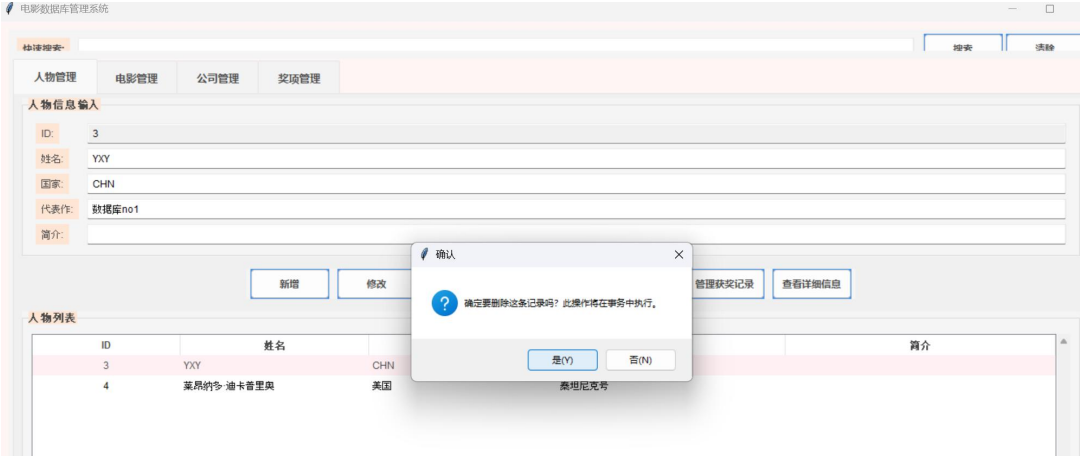


程序演示
(4分)

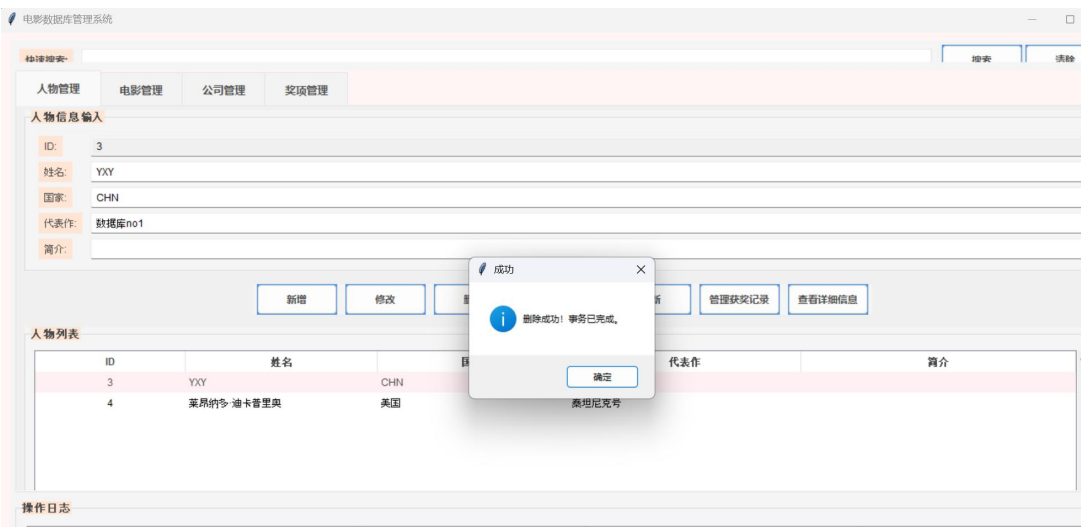
然后我们鼠标悬停在 yxy 一行并左键点击选中：



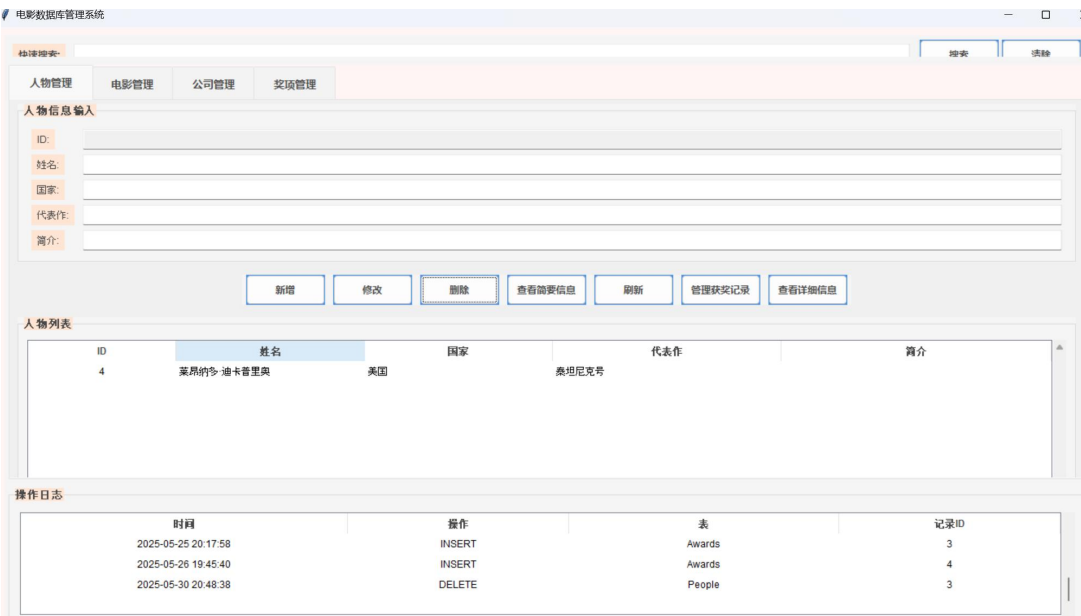
然后再是点击上方的删除按钮：



再次点击是后：



然后查看：我们可以发现日志多了一条删除 yxy 记录的 log，且人物列表中删除掉了 yxy 这一列。**成功实现删除！**



备注

触发器控制下的添加操作（20 分）

说明	（1 分）简要说明该操作所要完成的功能； （2 分）简要说明该触发器所要完成的功能 （1 分）该操作会涉及的表（以“表名”的形式给出）。 （2 分）该操作输入数据以及输入数据应该满足的条件，如：数值范围、是否为空； （6 分）实现该操作的关键代码（高级语言、SQL），截图即可； （8 分）如何执行该操作，按所述方法能够正常演示程序则给分。	
功能描述 (1 分)	系统在对“人物”、“电影”、“公司”和“奖项”等表执行插入操作时，自动通过触发器将本次插入行为记录到日志表 operation_logs 中。该日志记录包括操作类型（INSERT）、表名及被插入记录的主键 ID。	
触发器描述 (2 分)	在数据库中定义了如下触发器： after_person_insert : 在 People 表新增记录后触发，插入对应日志； after_movie_insert : 在 Movies 表新增记录后触发； after_company_insert : 在 Companies 表新增记录后触发； after_award_insert : 在 Awards 表新增记录后触发。 每个触发器在目标表成功插入新记录后，自动向 operation_logs 表插入一条 INSERT 类型的操作 log，记录表名和被插入记录的主键——即 id。	
涉及的表 (1 分)	People、Movies、Companies、Awards、operation_logs	
输入数据 (2 分)	字段	规则
	name, country	创建 People 表中记录时，名字和国籍是必填字段
	title	创建 Movies 表中记录时，名字是必填字段
	name	创建 Companies 表中记录时，电影名字是必填字段
	name, year, category	创建 Awards 表中记录时，奖项名称、年份以及分类是必填字段
插入操作源码 (3 分)	People 表: <pre>def insert_person(name, country, masterpiece, brief_intro): 1个用法 conn = get_connection() try: with conn.cursor() as cursor: sql = "INSERT INTO People (name, country, masterpiece, brief_intro) VALUES (%s, %s, %s, %s)" cursor.execute(sql, args=(name, country, masterpiece, brief_intro)) conn.commit() finally: conn.close()</pre> Movies 表: <pre>def insert_movie(title, release_year, director, genre, box_office, description): 1个用法 conn = get_connection() try: with conn.cursor() as cursor: sql = """INSERT INTO Movies (title, release_year, director, genre, box_office, description) VALUES (%s, %s, %s, %s, %s, %s)""" cursor.execute(sql, args=(title, release_year, director, genre, box_office, description)) conn.commit() finally: conn.close()</pre>	

Companies 表:

```
def insert_company(name, country, founded_year, industry, revenue, description): 1个用法
    conn = get_connection()
    try:
        with conn.cursor() as cursor:
            sql = """INSERT INTO Companies (name, country, founded_year, industry, revenue, description)
                VALUES (%s, %s, %s, %s, %s, %s)"""
            cursor.execute(sql, args=(name, country, founded_year, industry, revenue, description))
        conn.commit()
    finally:
        conn.close()
```

Awards 表:

```
def insert_award(name, category, year, description): 1个用法
    conn = get_connection()
    try:
        with conn.cursor() as cursor:
            sql = """INSERT INTO Awards (name, category, year, description)
                VALUES (%s, %s, %s, %s)"""
            cursor.execute(sql, args=(name, category, year, description))
        conn.commit()
    finally:
        conn.close()
```

如下:

```
# 创建触发器 - People表
cursor.execute("DROP TRIGGER IF EXISTS after_person_insert")
cursor.execute("""
    CREATE TRIGGER after_person_insert
    AFTER INSERT ON People
    FOR EACH ROW
    BEGIN
        INSERT INTO operation_logs (operation_type, table_name, record_id)
        VALUES ('INSERT', 'People', NEW.person_id);
    END
""")

# 创建触发器 - Movies表
cursor.execute("DROP TRIGGER IF EXISTS after_movie_insert")
cursor.execute("""
    CREATE TRIGGER after_movie_insert
    AFTER INSERT ON Movies
    FOR EACH ROW
    BEGIN
        INSERT INTO operation_logs (operation_type, table_name, record_id)
        VALUES ('INSERT', 'Movies', NEW.movie_id);
    END
""")
```

触发器
源码
(3分)

```
# 创建触发器 - Companies表
cursor.execute("DROP TRIGGER IF EXISTS after_company_insert")
cursor.execute("""
    CREATE TRIGGER after_company_insert
    AFTER INSERT ON Companies
    FOR EACH ROW
    BEGIN
        INSERT INTO operation_logs (operation_type, table_name, record_id)
        VALUES ('INSERT', 'Companies', NEW.company_id);
    END
""")

# 创建触发器 - Awards表
cursor.execute("DROP TRIGGER IF EXISTS after_award_insert")
cursor.execute("""
    CREATE TRIGGER after_award_insert
    AFTER INSERT ON Awards
    FOR EACH ROW
    BEGIN
        INSERT INTO operation_logs (operation_type, table_name, record_id)
        VALUES ('INSERT', 'Awards', NEW.award_id);
    END
""")
```

我们为 People 表添加一个演员——基努里维斯，国籍为加拿大：

然后点击添加：

程序演示
(4分)

然后点击确定：

人物管理

电影管理

公司管理

奖项管理

人物信息输入

ID:

姓名:

国家:

代表作:

简介:

新增

修改

删除

查看简要信息

刷新

管理获奖记录

查看详细信息

人物列表

ID	姓名	国家	代表作	简介
4	莱昂纳多·迪卡普里奥	美国	泰坦尼克号	
5	基努里维斯	加拿大		

操作日志

时间	操作	表	记录ID
2025-05-26 19:45:40	INSERT	Awards	4
2025-05-30 20:48:38	DELETE	People	3
2025-05-30 21:10:31	INSERT	People	5

可以看到生成了新日志：

成功添加！完成！

我们为 People 表增加一个鲍勃奥登科克，这次不写国籍：

人物管理

电影管理

公司管理

奖项管理

人物信息输入

ID:

姓名: 鲍勃奥登科克

国家:

代表作:

简介:

新增

修改

删除

查看简要信息

刷新

管理获奖记录

查看详细信息

人物列表

ID	姓名	国家	代表作	简介
4	莱昂纳多·迪卡普里奥	美国	泰坦尼克号	
5	基努里维斯	加拿大		

点击新增：

人物管理

电影管理

公司管理

奖项管理

人物信息输入

ID:

姓名: 鲍勃奥登科克

国家:

代表作:

简介:

新增

修改

删除

查看简要信息

刷新

管理获奖记录

查看详细信息

人物列表

ID	姓名	国家	代表作	简介
4	莱昂纳多·迪卡普里奥	美国	泰坦尼克号	

程序演示
(4分)

错误

姓名和国家是必填项！

确定

	<div>点击确定：</div> <div></div> <div>注意到此次没有增加新的记录。完成！</div>
备注	

存储过程控制下的更新操作（18 分）

说明	<p>（1 分）简要说明该操作所要完成的功能；</p> <p>（1 分）简要说明该存储过程所要完成的功能；</p> <p>（2 分）说明该操作涉及操作的表（必须包含两张或两张以上的关系表，以“表名形式”描述）</p> <p>（1 分）表连接涉及字段描述（描述方式为“表 1. 属性=表 2. 属性”）</p> <p>（2 分）该操作会修改字段（以“表名. 字段名”的形式给出），以及修改规则，如新数值的计算方法、在何种条件下予以修改等；</p> <p>（6 分）实现该操作的关键代码（高级语言、SQL），截图即可；</p> <p>（5 分）如何执行该操作，按所述方法能够正常演示程序则给分。</p>	
功能描述 （1 分）	在更新 People 表信息时，通过 update_person_info 完成数据的修改。该存储过程除了更新 People 表记录，还会同步向表 operation_logs 插入一条 UPDATE 类型的日志，记录本次修改操作的对象表名和主键 ID。	
存储过程 功能描述 （1 分）	存储过程 update_person_info 接受 5 个参数（人物 ID 及需更新的 4 项字段），执行以下两个步骤： 1. 使用 UPDATE 修改 People 表中对应记录 2. 使用 INSERT 将操作信息写入 operation_logs 表，记录表名“People”、操作类型“UPDATE”与被修改记录的主键 ID。	
涉及的关系表 (2分)	People 以及 operation_logs	
表连接涉及 字段(1)	更新不涉及表连接操作，只是按顺序更新两张表，依赖以下字段： People.people_id: 主键，用于定位要更新的记录 operation_logs.record_id: 记录更新操作对应的主键 ID（即 people_id）	
更改字段 （2 分）	字段	规则
	name	非空，人物姓名

	country	非空，人物国籍
	masterpiece	可为空，代表作品
	brief_intro	可为空，人物简介
更新代码 (3分)	<p>程序中没有直接编写 update 的 SQL 语句，而是通过调用数据库的储存逻辑实现更新。但在此处给出直接 update 的 SQL 语句：</p> <pre>''' # 直接SQL更新People def update_person_directly(people_id, name, country, masterpiece, brief_intro): conn = get_connection() try: with conn.cursor() as cursor: sql = """ UPDATE People SET name = %s, country = %s, masterpiece = %s, brief_intro = %s WHERE people_id = %s """ cursor.execute(sql, (name, country, masterpiece, brief_intro, people_id)) sql_log = """ INSERT INTO operation_logs (operation_type, table_name, record_id) VALUES (%s, %s, %s) """ cursor.execute(sql_log, ('UPDATE', 'People', people_id)) conn.commit() except Exception as e: conn.rollback() raise e finally: conn.close() '''</pre>	
创建存储 过程源码 (3分)	<p>如图：</p> <pre>''' # 创建存储过程 - People表 cursor.execute("DROP PROCEDURE IF EXISTS update_person_info") cursor.execute(""" CREATE PROCEDURE update_person_info(IN p_id INT, IN p_name VARCHAR(100), IN p_country VARCHAR(100), IN p_masterpiece TEXT, IN p_brief_intro TEXT) BEGIN UPDATE People SET name = p_name, country = p_country, masterpiece = p_masterpiece, brief_intro = p_brief_intro WHERE people_id = p_id; INSERT INTO operation_logs (operation_type, table_name, record_id) VALUES ('UPDATE', 'People', p_id); END """)</pre>	
存储过程 执行源码 (1分)		

如图：

```
def update_person_with_procedure(people_id, name, country, masterpiece, brief_intro): 1个用法
    conn = get_connection()
    try:
        with conn.cursor() as cursor:
            cursor.execute( query: "CALL update_person_info(%s, %s, %s, %s, %s)",
                            args: (people_id, name, country, masterpiece, brief_intro))
        conn.commit()
    finally:
        conn.close()
```

我们此处对基努里维斯这条记录进行更新，为其加上代表作《疾速追杀》，如图：

然后点击“修改”：

然后点击确定：

程序演示
(2分)

日志依然更新成功：

人物管理

电影管理

公司管理

奖项管理

人物信息输入

ID:

姓名:

国家:

代表作:

简介:

新增

修改

删除

查看简要信息

刷新

管理获奖记录

查看详细信息

人物列表

ID	姓名	国家	代表作	简介
4	莱昂纳多·迪卡普里奥	美国	泰坦尼克号	
5	基努里维斯	加拿大	疾速追杀	

操作日志

时间	操作	表	记录ID
2025-05-30 20:48:38	DELETE	People	3
2025-05-30 21:10:31	INSERT	People	5
2025-05-30 21:50:18	UPDATE	People	5

修改操作实现！

我们依然对基努里维斯这条记录进行修改，此处我们故意将名字改为空并点击“修改”：

人物管理

电影管理

公司管理

奖项管理

人物信息输入

ID:

姓名:

国家:

代表作:

简介:

新增

修改

删除

查看简要信息

刷新

管理获奖记录

查看详细信息

人物列表

ID	姓名	国家	代表作	简介
4	莱昂纳多·迪卡普里奥	美国	泰坦尼克号	
5	基努里维斯	加拿大	疾速追杀	

程序演示
(2分)

点击确定后发现记录没有被更新：

人物管理

电影管理

公司管理

奖项管理

人物信息输入

ID:

姓名:

国家:

代表作:

简介:

新增

修改

删除

查看简要信息

刷新

管理获奖记录

查看详细信息

人物列表

ID	姓名	国家	代表作	简介
4	莱昂纳多·迪卡普里奥	美国	泰坦尼克号	
5	基努里维斯	加拿大	疾速追杀	

出错，修改终止！

备注

5. 含有视图的查询操作（15 分）

说明	<p>(1 分) 简要说明该操作所要完成的功能；</p> <p>(1 分) 简要说明建立的该视图的功能；</p> <p>(2 分) 简要说明该操作涉及的关系数据表（以“表名”的形式给出）</p> <p>(1 分) 简要说明表连接涉及的字段（以“表 1. 属性=表 2. 属性”）</p> <p>(6 分) 实现该操作的关键代码（高级语言、SQL），截图即可；</p> <p>(4 分) 如何执行该操作，按所述方法能够正常演示程序则给分。</p>
操作功能描述(1分)	系统在展示人物信息简要列表时，并未直接查询主表 People，而是通过数据库视图 people_summary 实现。该视图仅保留最核心的 4 个字段（ID、姓名、国籍、代表作），方便用于界面展示和快速查询。
视图功能描述(1分)	该视图仅包含 People 表中的部分字段（除 brief_intro）。当用户点击“查看简要信息”时，程序从该视图中提取数据并显示在新窗口中，而不是直接查询整个主表，加快了效率。
涉及的关系表(2分)	主表：People 视图：people_summary
表连接字段（1 分）	不涉及多表连接，people_summary 和 People 表的字段一一对应。
创建视图代码(3分)	<p>如图：</p>  <pre> # 创建视图 - People cursor.execute("DROP VIEW IF EXISTS people_summary") cursor.execute(""" CREATE VIEW people_summary AS SELECT people_id, name, country, masterpiece FROM People """) </pre>
查询代码（3 分）	<p>如图：</p>  <pre> def get_people_summary(): 1 个用法 conn = get_connection() try: with conn.cursor() as cursor: cursor.execute("SELECT * FROM people_summary") result = cursor.fetchall() return result finally: conn.close() </pre>


```
def show_summary(self): 1 个用法
    """显示人物简要信息（视图查询）"""
    summary_window = tk.Toplevel(self)
    summary_window.title("人物简要信息视图")

    tree = ttk.Treeview(summary_window, columns=("ID", "姓名", "国家", "代表作"), show="headings")
    for col in ["ID", "姓名", "国家", "代表作"]:
        tree.heading(col, text=col)
        tree.column(col, width=100)

    scrollbar = ttk.Scrollbar(summary_window, orient=tk.VERTICAL, command=tree.yview)
    tree.configure(yscrollcommand=scrollbar.set)

    tree.grid(row=0, column=0, sticky=(tk.W, tk.E))
    scrollbar.grid(row=0, column=1, sticky=(tk.N, tk.S))

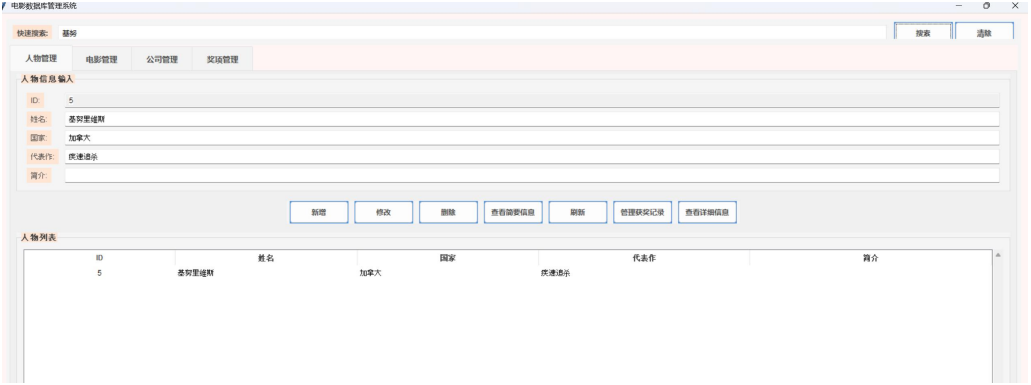
    # 从视图获取数据
    summary_data = db.get_people_summary()
    for item in summary_data:
        tree.insert(parent="", tk.END, values=(
            item['people_id'],
            item['name'],
            item['country'],
            item['masterpiece']
        ))
    )
```

我们依然查询“基努里维斯”，我们只需要打一个“基努”即可：



程序演示
(4分)

然后点击搜索：



成功查询，然后我们再点击“查看简要信息”：

人物简要信息视图			
ID	姓名	国家	代表作
4	莱昂纳多·迪卡普里奥	美国	泰坦尼克号
5	基努里维斯	加拿大	疾速追杀

可以看到选择的是“基努里维斯”那条记录。

我们再试试搜不存在的记录，例如搜“黄飞鸿”：

快速搜索: 黄飞鸿

搜索清除

人物管理

电影管理

公司管理

奖项管理

人物信息输入

ID: 5

姓名: 基努里维斯

国家: 加拿大

代表作: 疾速追杀

简介:

新增

修改

删除

查看简要信息

刷新

管理获奖记录

查看详细信息

人物列表

ID	姓名	国家	代表作	简介
----	----	----	-----	----

搜索结果

未找到匹配项

确定

会显示不存在，证明功能正确！

另外，此处还有查看详细信息的功能，其在 People 表基础上增加了查看获得奖的记录：

人物详细信息 - 基努里维斯

—

□

×

基本信息

姓名: 基努里维斯

国家: 加拿大

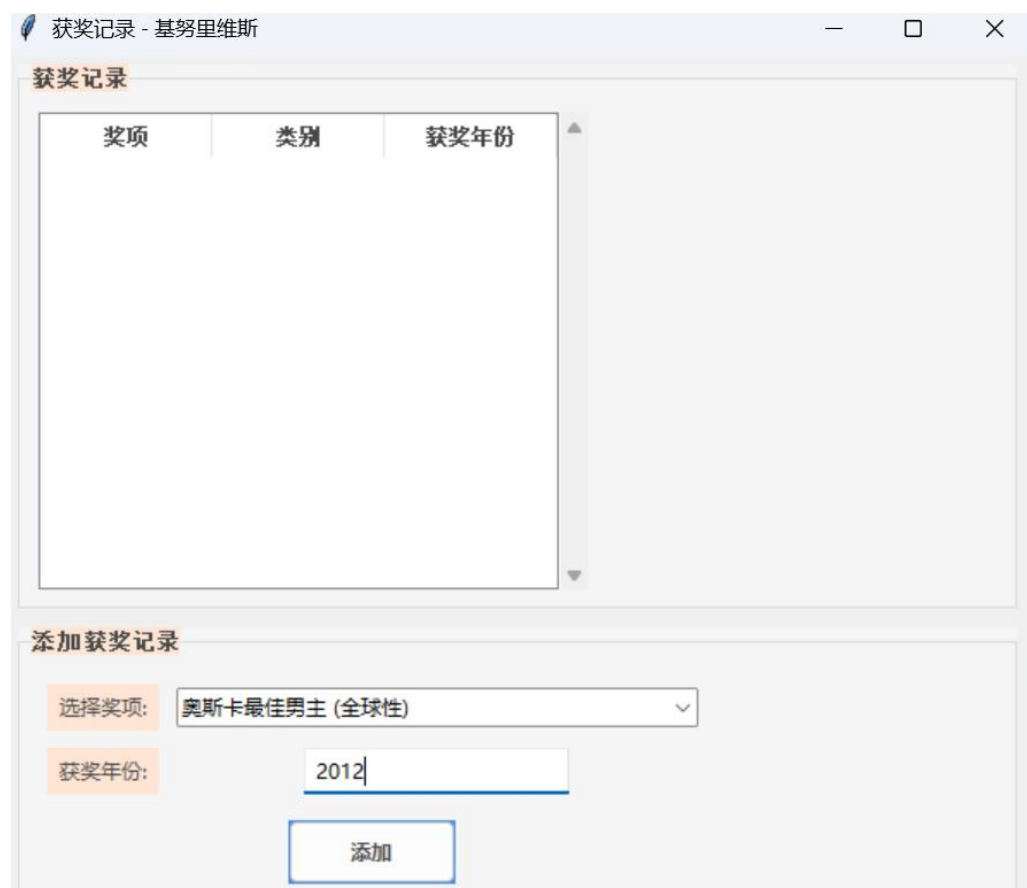
代表作: 疾速追杀

简介:

获奖记录

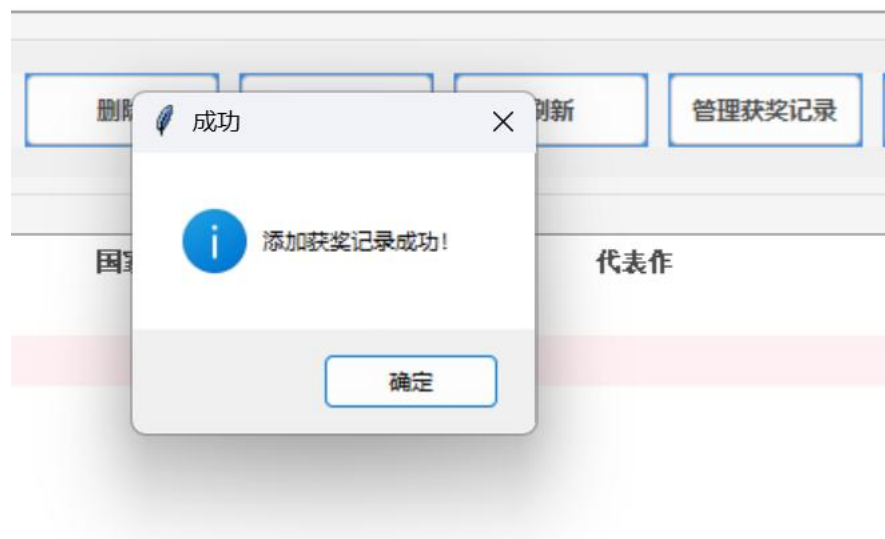
奖项	类别	年份
----	----	----

我们还可以点击基努里维斯这条记录后，点击“管理获奖记录”，为其补充所获得的奖：



The screenshot shows a window titled "获奖记录 - 基努里维斯" (Award Record - Keanu Reeves). Inside, there is a section labeled "获奖记录" (Award Record) containing a table with three columns: "奖项" (Award), "类别" (Category), and "获奖年份" (Award Year). Below this is a section labeled "添加获奖记录" (Add Award Record) with a form. The form has a dropdown menu for "选择奖项:" (Select Award) with the value "奥斯卡最佳男主 (全球性)" (Oscar Best Actor (Global)), a text input for "获奖年份:" (Award Year) with the value "2012", and a "添加" (Add) button.

然后点击“添加”：



然后点击“确定”后再查看详细信息：
(此处我添加了两次奖项)

人物详细信息 - 基努里维斯

基本信息

姓名: 基努里维斯

国家: 加拿大

代表作: 疾速追杀

简介:

获奖记录

奖项	类别	年份
奥斯卡最佳男主	全球性	2020
奥斯卡最佳男主	全球性	2012

至此，全部完成！

备注