

[TOC]

影视管理数据库设计与操作文档

一. 数据表设计

根据提供的ER图，我们可以确定需要建立以下数据表：

1.用户实体表（用户实体）

- 列名：id, name, password
- 主键：id
- SQL语句：

```
CREATE TABLE IF NOT EXISTS 用户实体 (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(255),  
    password VARCHAR(255)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

2.视频实体表（视频实体）

- 列名：id, name, type, score, starring, date, description, path
- 主键：id
- SQL语句：

```
CREATE TABLE IF NOT EXISTS 视频实体 (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(255),  
    type VARCHAR(255),  
    score DECIMAL(5,2),  
    starring VARCHAR(255),  
    date DATE,  
    description TEXT,  
    path VARCHAR(255)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

3.收藏关系表（收藏关系）

- 列名：id（主键）， user_id（外键， 关联用户实体表的id）， video_id（外键， 关联视频实体表的id）
- 主键：id
- 外键： user_id REFERENCES 用户实体(id), video_id REFERENCES 视频实体(id)
- SQL语句：

```
CREATE TABLE IF NOT EXISTS 收藏关系 (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    user_id INT,  
    video_id INT,  
    FOREIGN KEY (user_id) REFERENCES 用户实体(id),  
    FOREIGN KEY (video_id) REFERENCES 视频实体(id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

4.留言关系表（留言关系）

- 列名：id（主键），user_id（外键，关联用户实体表的id），video_id（外键，关联视频实体表的id），description, date
- 主键：id
- 外键：user_id REFERENCES 用户实体(id), video_id REFERENCES 视频实体(id)
- SQL语句：

```
CREATE TABLE IF NOT EXISTS 留言关系 (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    user_id INT,  
    video_id INT,  
    description TEXT,  
    date DATE,  
    FOREIGN KEY (user_id) REFERENCES 用户实体(id),  
    FOREIGN KEY (video_id) REFERENCES 视频实体(id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

二. 查询和删除操作

1.查询操作

- SQL语句

```
SELECT DISTINCT u.name, avg_score  
FROM 用户实体 u  
INNER JOIN 留言关系 l ON u.id = l.user_id  
INNER JOIN (  
    SELECT user_id, AVG(v.score) AS avg_score  
    FROM 收藏关系 s  
    INNER JOIN 视频实体 v ON s.video_id = v.id  
    GROUP BY user_id  
) AS s ON u.id = s.user_id  
WHERE l.description = '喜欢看动作电影'  
GROUP BY u.id  
HAVING avg_score > 8
```

查询结果

70 -- 首先在外层查询中使用了DISTINCT关键字来确保结果集中的用户名称唯一。

71 -- 在SELECT列表中包含了用户名称和平均分avg_score，这是为了在HAVING子句中使用。

72 • SELECT DISTINCT u.name, avg_score

73 FROM 用户实体 u

74 -- 使用INNER JOIN将用户实体表与留言关系表关联，条件是用户实体表中的id与留言关系表中的user_id相匹配。

75 INNER JOIN 留言关系 l ON u.id = l.user_id

76 -- 使用INNER JOIN将用户实体表与子查询结果（计算用户收藏的视频的平均分数）关联，条件是用户实体表中的id与子查询结果中的user_id相匹配。

77 INNER JOIN (

78 -- 子查询：计算每个用户收藏的视频的平均分数

79 SELECT user_id, AVG(v.score) AS avg_score

80 FROM 收藏关系 s

81 -- 使用INNER JOIN将收藏关系表与视频实体表关联，条件是收藏关系表中的video_id与视频实体表中的id相匹配。

82 INNER JOIN 视频实体 v ON s.video_id = v.id

83 -- 使用GROUP BY子句对用户ID进行分组，以便计算每个用户收藏的视频的平均分数。

84 GROUP BY user_id

85) AS s ON u.id = s.user_id

86 -- 使用WHERE子句筛选出留言为“喜欢看动作电影”的用户。

87 WHERE l.description = '喜欢看动作电影'

88 -- 使用GROUP BY子句对用户ID进行分组，以便在HAVING子句中使用聚合函数。

89 GROUP BY u.id

90 -- 使用HAVING子句筛选出收藏的影视平均分大于8的用户。

91 HAVING avg_score > 8

92 -- 使用LIMIT子句限制返回结果的数量，避免返回过多的结果。

93 LIMIT 0, 1000;

94

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Result Grid

	name	avg_score
▶	用户1	8.150000

Result 6 x

Read O

2.删除操作

SQL语句

DELETE FROM 留言关系 WHERE user_id IN (SELECT id FROM 用户实体 WHERE name LIKE '用户1');

删除结果

删除前

95 • SELECT * FROM 留言关系;

96

97

Result Grid

Filter Rows:

Export/Import:

Wrap Cell Content:

Result Grid

id	user_id	video_id	description	date
1	1	1	喜欢看动作电影	2024-04-01
2	2	2	喜欢看爱情电影	2024-03-28
3	3	3	喜欢看喜剧电影	2024-03-25

留言关系 7 x

Apply

删除后

95 • SELECT * FROM 留言关系;

96 • DELETE FROM 留言关系 WHERE user_id IN (SELECT id FROM 用户实体 WHERE name LIKE '用户1');

97

Result Grid

Filter Rows:

Export/Import:

Wrap Cell Content:

Result Grid

id	user_id	video_id	description	date
▶	2	2	喜欢看爱情电影	2024-03-28
•	3	3	喜欢看喜剧电影	2024-03-25

留言关系 8 x

三. 查询语句有效性验证

1.正例

(1). 查询留言为“喜欢看动作电影”的用户中，收藏的影视中有类型为“动作”的视频的用户名字

- SQL语句

```
SELECT DISTINCT u.name
FROM 用户实体 u
INNER JOIN 留言关系 l ON u.id = l.user_id
INNER JOIN 收藏关系 s ON u.id = s.user_id
INNER JOIN 视频实体 v ON s.video_id = v.id
WHERE l.description = '喜欢看动作电影'
AND v.type = '动作'
LIMIT 0, 1000;
```

- 查询结果

```
100 • SELECT DISTINCT u.name
101 FROM 用户实体 u
102 INNER JOIN 留言关系 l ON u.id = l.user_id
103 INNER JOIN 收藏关系 s ON u.id = s.user_id
104 INNER JOIN 视频实体 v ON s.video_id = v.id
105 WHERE l.description = '喜欢看动作电影'
106 AND v.type = '动作'
107 LIMIT 0, 1000;
108
109
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
name			
用户1			

(2). 查询留言为“喜欢看动作电影”的用户中，收藏的影视中有演员包含“演员1”的视频的用户名字。

- SQL语句

```
SELECT DISTINCT u.name
FROM 用户实体 u
INNER JOIN 留言关系 l ON u.id = l.user_id
INNER JOIN 收藏关系 s ON u.id = s.user_id
INNER JOIN 视频实体 v ON s.video_id = v.id
WHERE l.description = '喜欢看动作电影'
AND v.starring LIKE '%演员1%'
LIMIT 0, 1000;
```

• 查询结果

```
109 • SELECT DISTINCT u.name
110 FROM 用户实体 u
111 INNER JOIN 留言关系 l ON u.id = l.user_id
112 INNER JOIN 收藏关系 s ON u.id = s.user_id
113 INNER JOIN 视频实体 v ON s.video_id = v.id
114 WHERE l.description = '喜欢看动作电影'
115 AND v.starring LIKE '%演员1%'
```

Result Grid

name

用户1

(3). 查询留言为“喜欢看动作电影”的用户中，收藏的影视中有评分高于7的视频的用户名字。

• SQL语句

```
SELECT DISTINCT u.name
FROM 用户实体 u
INNER JOIN 留言关系 l ON u.id = l.user_id
INNER JOIN 收藏关系 s ON u.id = s.user_id
INNER JOIN 视频实体 v ON s.video_id = v.id
WHERE l.description = '喜欢看动作电影'
AND v.score > 7
LIMIT 0, 1000;
```

• 查询结果

```
116 -- 3.查询留言为“喜欢看动作电影”的用户中，收藏的影视中有评分高于7的视频的用户名字。
117 • SELECT DISTINCT u.name
118 FROM 用户实体 u
119 INNER JOIN 留言关系 l ON u.id = l.user_id
120 INNER JOIN 收藏关系 s ON u.id = s.user_id
121 INNER JOIN 视频实体 v ON s.video_id = v.id
122 WHERE l.description = '喜欢看动作电影'
123 AND v.score > 7
124 LIMIT 0, 1000;
```

Result Grid

name

用户1

2.反例

(1). 查询留言为“喜欢看动作电影”的用户中，收藏的影视中没有类型为“动画”的视频的用户名字。

- SQL语句

```
SELECT DISTINCT u.name
FROM 用户实体 u
INNER JOIN 留言关系 l ON u.id = l.user_id
LEFT JOIN 收藏关系 s ON u.id = s.user_id
LEFT JOIN 视频实体 v ON s.video_id = v.id
WHERE l.description = '喜欢看动作电影'
AND (v.type = '动画' AND v.type IS NULL)
LIMIT 0, 1000;
```

- 查询结果

```
127 -- 1.查询留言为“喜欢看动作电影”的用户中，收藏的影视中没有类型为“动画”的视频的用户名字。
128 • SELECT DISTINCT u.name
129 FROM 用户实体 u
130 INNER JOIN 留言关系 l ON u.id = l.user_id
131 LEFT JOIN 收藏关系 s ON u.id = s.user_id
132 LEFT JOIN 视频实体 v ON s.video_id = v.id
133 WHERE l.description = '喜欢看动作电影'
134 AND (v.type = '动画' AND v.type IS NULL)
135 LIMIT 0, 1000;
136
137
```

Result Grid			Filter Rows: <input type="text"/>	Export:	Wrap Cell Content:
name					

(2). 查询留言为“喜欢看动作电影”的用户中，收藏的影视中没有演员包含“演员2”的视频的用户名字。

- SQL语句

```
SELECT DISTINCT u.name
FROM 用户实体 u
INNER JOIN 留言关系 l ON u.id = l.user_id
LEFT JOIN 收藏关系 s ON u.id = s.user_id
LEFT JOIN 视频实体 v ON s.video_id = v.id
WHERE l.description = '喜欢看动作电影'
AND (v.starring NOT LIKE '%演员2%' OR v.starring IS NULL)
LIMIT 0, 1000;
```

• 查询结果

136 -- 2.查询留言为“喜欢看动作电影”的用户中，收藏的影视中没有演员包含“演员2”的视频的用户名字。

137 • SELECT DISTINCT u.name

138 FROM 用户实体 u

139 INNER JOIN 留言关系 l ON u.id = l.user_id

140 LEFT JOIN 收藏关系 s ON u.id = s.user_id

141 LEFT JOIN 视频实体 v ON s.video_id = v.id

142 WHERE l.description = '喜欢看动作电影'

143 AND (v.starring NOT LIKE '%演员2%' OR v.starring IS NULL)

144 LIMIT 0, 1000;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	name
▶	用户1

(3). 查询留言为“喜欢看动作电影”的用户中，收藏的影视中没有评分低于6的视频的用户名字。

• SQL语句

```
SELECT DISTINCT u.name
FROM 用户实体 u
INNER JOIN 留言关系 l ON u.id = l.user_id
LEFT JOIN 收藏关系 s ON u.id = s.user_id
LEFT JOIN 视频实体 v ON s.video_id = v.id
WHERE l.description = '喜欢看动作电影'
AND (v.score >= 6 OR v.score IS NULL)
LIMIT 0, 1000;
```

• 查询结果

145 -- 3.查询留言为“喜欢看动作电影”的用户中，收藏的影视中没有评分低于6的视频的用户名字。

146 • SELECT DISTINCT u.name

147 FROM 用户实体 u

148 INNER JOIN 留言关系 l ON u.id = l.user_id

149 LEFT JOIN 收藏关系 s ON u.id = s.user_id

150 LEFT JOIN 视频实体 v ON s.video_id = v.id

151 WHERE l.description = '喜欢看动作电影'

152 AND (v.score >= 6 OR v.score IS NULL)

153 LIMIT 0, 1000;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	name
▶	用户1

四.数据库表结构内容图

结构	内容
----	----

结构

内容

tables

Tables_in_影视管理

收藏关系

用户实体

留言关系

视频实体

用户实体

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
password	varchar(255)	YES		NULL	

	id	name	password
	1	用户1	password1
	2	用户2	password2
	3	用户3	password3
	NULL	NULL	NULL

视频实体

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
type	varchar(255)	YES		NULL	
score	decimal(5,2)	YES		NULL	
starring	varchar(255)	YES		NULL	
date	date	YES		NULL	
description	text	YES		NULL	
path	varchar(255)	YES		NULL	

	id	name	type	score	starring	date	description	path
	1	电影1	动作	8.50	演员1, 演员2	2023-01-01	电影1的描述	/path/to/电影1
	2	电影2	爱情	7.80	演员3, 演员4	2022-12-15	电影2的描述	/path/to/电影2
	3	电影3	喜剧	9.20	演员5, 演员6	2023-03-20	电影3的描述	/path/to/电影3
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

收藏关系

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
user_id	int	YES	MUL	NULL	
video_id	int	YES	MUL	NULL	

	id	user_id	video_id
	1	1	1
	2	1	2
	3	2	3
	NULL	NULL	NULL

留言关系

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
user_id	int	YES	MUL	NULL	
video_id	int	YES	MUL	NULL	
description	text	YES		NULL	
date	date	YES		NULL	

	id	user_id	video_id	description	date
	7	1	1	喜欢看动作电影	2024-04-01
	8	2	2	喜欢看爱情电影	2024-03-28
	9	3	3	喜欢看喜剧电影	2024-03-25
	NULL	NULL	NULL	NULL	NULL