北京交通大学

BEIJING JIAOTONG UNIVERSITY

北京交通大学

课程名称：数据库系统原理

实验题目 ： 数据库系统原理Lab2

学号 ： 22281188

姓名 ： 江家玮

班级 ： 计科2204班

指导老师 ： 刘真老师

报告日期 ： 2025-04-19

```
22281188-江家玮-数据库Lab5/
├── DBLab4_Permissions_22281188.sql     # 用户创建与权限管理SQL脚本
├── DBLab4_UPDATE2LAB3.sql              # 更新Lab3脚本
├── 用户密码重置业务流程图.pdf            # 密码重置流程图 PDF 版
├── 用户密码重置业务流程图.vsdx             # 密码重置流程图 Visio 源文件
├── 用户修改密码业务流程图.vsdx             # 密码修改流程图 Visio 源文件
└── 用户修改密码业务流程图.pdf           # 密码修改流程图 PDF 版
├── 22281188-江家玮-数据库Lab5.pdf       # 实验报告
```

一、视图设计与操作

请针对你在作业2中所设计的用户界面的数据需求

1）至少设计并创建行列子集视图、带表达式的视图和分组视图各一个，并将代码加入自己的建库脚本中。

我在作业2中设计的数据库模型是关于“航空公司航班查询业务”的，包含了 `Airlines`，`Routes`，`Flights`，`Passengers`，`Bookings`，`Payments`，`Tickets`，`Flight_Status`，`Airports` 这些表。

下面是模型设计的视图：

- **行列子集视图:**
  - **目的:** 创建一个只显示中国航空公司名称和联系方式的视图。这限制了行（只选总部在中国 'China' 的）和列（只选 `name` 和 `contact_info`）。

```sql
-- 创建视图：只显示中国航空公司的名称和联系方式
CREATE VIEW Chinese_Airlines_Contact AS
SELECT
    name,
    contact_info
FROM
    Airlines
WHERE
    headquarters_location LIKE '%China%';
```

- **带表达式的视图 (View with Expression):**
  - **目的:** 创建一个显示每个航班的总座位数、已预订座位数以及计算出的可用座位数的视图。这里包含了一个表达式 `(total_seats - reserved_seats)`。

```sql
-- 创建视图：显示航班座位可用情况
CREATE VIEW Flight_Availability AS
SELECT
    flight_id,
    total_seats,
    booked_seats,
    (total_seats - booked_seats) AS available_seats
FROM
    Flight;
```

- **分组视图 (Grouped View):**
  - **目的:** 创建一个按航空公司分组，统计每个航空公司运营的航班数量的视图。这需要使用 `GROUP BY` 和聚合函数 `COUNT()`。

```sql
-- 创建视图：统计各航空公司的航班数量
CREATE VIEW Airline_Flight_Count AS
SELECT
    a.name AS airline_name,
    COUNT(f.flight_id) AS number_of_flights -- 统计航班数
FROM
    Airline a
JOIN
    Flight f ON a.airline_id = f.airline_id
GROUP BY
    a.name; -- 按航空公司名称分组
```

2）编写测试代码，通过视图实现数据查询操作

以下是查询上面创建的三个视图的测试代码：

```sql
-- 查询中国航空公司的联系方式视图
SELECT * FROM Chinese_Airlines_Contact;

-- 查询航班座位可用情况视图
SELECT * FROM Flight_Availability;

-- 查询各航空公司的航班数量视图
SELECT * FROM Airline_Flight_Count;

-- 条件查询
-- 查询可用座位超过100个的航班信息
SELECT * FROM Flight_Availability WHERE available_seats > 100;
```

```sql

-- 查询航班数量超过3个的航空公司
SELECT * FROM Airline_Flight_Count WHERE number_of_flights > 3;
```

```
mysql> -- 查询航班座位可用情况视图
mysql> SELECT * FROM Flight_Availability;
+-----------+-------------+-------------+-----------------+
| flight_id | total_seats | booked_seats | available_seats |
+-----------+-------------+-------------+-----------------+
|      3001 |         180 |           0 |             180 |
|      3002 |         220 |           0 |             220 |
+-----------+-------------+-------------+-----------------+
2 rows in set (0.00 sec)

mysql>
mysql> -- 查询各航空公司的航班数量视图
mysql> SELECT * FROM Airline_Flight_Count;
+-------------------------+-------------------+
| airline_name            | number_of_flights |
+-------------------------+-------------------+
| China Eastern Airlines  |                 1 |
| China Southern Airlines |                 1 |
+-------------------------+-------------------+
2 rows in set (0.00 sec)

mysql>
mysql> -- 条件查询
mysql> -- 查询可用座位超过100个的航班信息
mysql> SELECT * FROM Flight_Availability WHERE available_seats > 100;
+-----------+-------------+-------------+-----------------+
| flight_id | total_seats | booked_seats | available_seats |
+-----------+-------------+-------------+-----------------+
|      3001 |         180 |           0 |             180 |
|      3002 |         220 |           0 |             220 |
+-----------+-------------+-------------+-----------------+
2 rows in set (0.00 sec)

mysql>
mysql> -- 查询航班数量超过3个的航空公司
mysql> SELECT * FROM Airline_Flight_Count WHERE number_of_flights > 3;
Empty set (0.00 sec)
```

3）对不同类别视图尝试进行数据修改和删除操作，分析其原因

- `Chinese_Airlines_Contact` **(行列子集视图):**
  - 尝试修改:

```sql
-- 尝试更新 'Airline C' 的联系方式
UPDATE Chinese_Airlines_Contact
SET contact_info = '888-888-8888'
WHERE name = 'Airline C';
```

原本的视图：

```
mysql> SELECT * FROM Chinese_Airlines_Contact;
+--------------+--------------+
| name         | contact_info |
+--------------+--------------+
| Airline C    | 123-456-7891 |
| 中国南方航空 | 95539        |
+--------------+--------------+
2 rows in set (0.00 sec)
```

执行完语句后：

```
mysql> -- 尝试更新 'Airline C' 的联系方式
mysql> UPDATE Chinese_Airlines_Contact
    -> SET contact_info = '888-888-8888'
    -> WHERE name = 'Airline C';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Chinese_Airlines_Cont
+--------------+--------------+
| name         | contact_info |
+--------------+--------------+
| Airline C    | 888-888-8888 |
| 中国南方航空 | 95539        |
+--------------+--------------+
2 rows in set (0.00 sec)
```

- 尝试删除:

```sql
-- 尝试删除 'Airline C'
DELETE FROM Chinese_Airlines_Contact
WHERE name = 'Airline C';
```

执行完语句后：

```
mysql> SELECT * FROM Chinese_Airlines_Contact;
+--------------+--------------+
| name         | contact_info |
+--------------+--------------+
| Airline C    | 888-888-8888 |
| 中国南方航空 | 95539        |
+--------------+--------------+
2 rows in set (0.00 sec)

mysql> -- 尝试删除 'Airline C'
mysql> DELETE FROM Chinese_Airlines_Contact
    -> WHERE name = 'Airline C';
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Chinese_Airlines_Contact;
+--------------+--------------+
| name         | contact_info |
+--------------+--------------+
| 中国南方航空 | 95539        |
+--------------+--------------+
1 row in set (0.00 sec)
```

- **能否实现: 可以实现** (Update 和 Delete)。

- **原因分析:** 这个视图只基于单个基表 (`Airlines`)，并且选择的列 (`name`, `contact_info`) 直接映射到基表的列。没有使用聚合函数、`GROUP BY`、`DISTINCT` 或表达式。因此，对视图的修改/删除操作可以明确地、无歧义地转换为对基表 `Airlines` 的相应操作。*当然，删除操作能否成功还取决于外键约束（例如，如果 `Airline c` 在 `Flights` 或 `Routes` 表中有引用，且设置了 `ON DELETE RESTRICT` 或 `NO ACTION`，则删除会失败）。*

- **`Flight_Availability` (带表达式的视图):**
  - 尝试修改 (非表达式列):

```sql
-- 尝试更新 flight_id 为 1 的已预订座位数
UPDATE Flight_Availability
SET booked_seats = 60
WHERE flight_id = 1;
```

```
mysql> -- 尝试更新 flight_id 为 1 的已预订座位数
mysql> UPDATE Flight_Availability
    -> SET booked_seats = 60
    -> WHERE flight_id = 1;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

  - 尝试修改 (表达式列):

```sql
-- 尝试更新视图中的计算列 (失败)
UPDATE Flight_Availability
SET available_seats = 130 -- 试图直接修改计算结果
WHERE flight_id = 101;
```

```
mysql> -- 尝试更新视图中的计算列 (预期失败)
mysql> UPDATE Flight_Availability
    -> SET available_seats = 130 -- 试图直接修改计算结果
    -> WHERE flight_id = 101;
ERROR 1348 (HY000): Column 'available_seats' is not updatable
```

  - 尝试删除:

```sql
DELETE FROM Flight_Availability
WHERE flight_id = 1;
```

```
mysql> -- 尝试通过视图删除数据
mysql> DELETE FROM Flight_Availability
    -> WHERE flight_id = 1;
Query OK, 0 rows affected (0.00 sec)
```

  - 能否实现:
    - 修改**非表达式列** (`reserved_seats`): **可以实现。**

- 修改**表达式列**（`available_seats`）：**不能实现。**
- 删除：**通常可以实现。**

- 原因分析:
  这个视图虽然包含表达式列

```
booked_seats
```

  但它仍然基于单个基表Flights

  - 对于 `UPDATE` 操作，如果更新的是基表中的列（如 `booked_seats`），并且该更新不会引起歧义，数据库系统（如 MySQL）通常允许这种操作。但是，不能直接更新由表达式计算得出的列（`available_seats`），因为数据库不知道如何将对这个计算结果的修改反向应用到基表的原始列上。
  - 对于 `DELETE` 操作，因为视图中的每一行仍然唯一对应基表 `Flights` 中的一行（由 `flight_id` 标识），所以删除操作通常是允许的（同样，需要考虑外键约束）。

- `Airline_Flight_Count` **(分组视图):**
  - **UPDATE (修改):** 尝试更新 '中国南方航空' 的航班数量（这逻辑上就不可能直接实现）。

```sql
-- 尝试更新分组视图中的聚合结果 (失败)
UPDATE Airline_Flight_Count
SET number_of_flights = 10 -- 试图修改 COUNT(*) 的结果
WHERE airline_name = '中国南方航空';
```

```
mysql> -- 尝试更新分组视图中的聚合结果 (失败)
mysql> UPDATE Airline_Flight_Count
    -> SET number_of_flights = 10 -- 试图修改 COUNT(*) 的结果
    -> WHERE airline_name = '中国南方航空';
ERROR 1288 (HY000): The target table Airline_Flight_Count of the UPDATE is not updatable
```

  UPDATE: 无法实现 (Failure)。

  - 原因:
    这个视图非常复杂，不满足可更新视图的任何基本条件：

    1. **基于多表:** 它连接了 `Airline` 和 `Flight` 两个表。
    2. **包含聚合函数:** 它使用了 `COUNT(f.flight_id)` 来计算航班数量。
    3. **包含 GROUP BY:** 它对结果进行了分组。 数据库完全无法理解如何将“把航班数量设置为 10”这个操作反向应用到基表（`Airline` 或 `Flight`）上。是应该在 `Flight` 表中为 '中国南方航空' 增加或删除航班记录吗？具体是哪些航班？这个操作是完全不明确、不可逆的。因

此，SQL 标准和所有主流数据库都不允许更新包含聚合函数或 `GROUP BY` 的视图。则会收到一个明确的错误，指出该视图不可更新。

- **DELETE (删除):** 尝试通过视图删除 '中国南方航空' 的统计记录。

```sql
-- 尝试通过分组视图删除数据 (失败)
DELETE FROM Airline_Flight_Count
WHERE airline_name = '中国南方航空';
```

```
mysql> -- 尝试通过分组视图删除数据 (失败)
mysql> DELETE FROM Airline_Flight_Count
    -> WHERE airline_name = '中国南方航空';
ERROR 1288 (HY000): The target table Airline_Flight_Count of the DELETE is not updatable
```

DELETE: 无法实现 (Failure)。

- **原因:** 与 UPDATE 类似，删除分组视图中的一行记录也是不明确的。删除 '中国南方航空' 这一行统计数据，是指要删除 `Airline` 表中名为 '中国南方航空' 的记录，还是要删除 `Flight` 表中所有属于 '中国南方航空' 的航班记录？数据库无法确定意图。涉及多表连接、聚合函数、`GROUP BY` 的视图通常是不可删除的。

4）在实验室报告中总结视图的主要作用

1. **简化数据操作:** 视图可以将复杂的查询（如多表连接、计算字段、数据筛选）封装起来，用户只需要查询简单的视图即可获取所需数据，隐藏了底层表的复杂结构和查询逻辑。
2. **提高数据安全性:** 视图可以限制用户访问数据的范围。通过创建只包含特定列或特定行（符合某些条件的行）的视图，可以向不同用户或应用程序暴露基表的不同子集，阻止用户访问敏感或不相关的数据。例如，只允许票务代理查看航班信息，而不允许查看航空公司的财务数据。
3. **实现逻辑数据独立性:** 视图提供了一个稳定的数据接口。即使底层基表的结构发生变化（例如，添加/删除列、重命名列、拆分表），只要视图的定义能够被相应修改以保持其对外接口不变，那么依赖该视图的应用程序就无需修改，从而提高了系统的可维护性。
4. **提供定制化的数据视图:** 可以为不同的用户或应用场景创建不同的视图，每个视图提供最符合其特定需求的数据展现方式。例如，为管理层创建一个包含汇总统计信息的视图，为客服创建一个包含详细乘客预订信息的视图。

二、用户口令管理设计题

用户注册与认证管理是数据库系统的基本功能，请为你在作业#2中选择的具体案例设计一个用户注册与认证中的口令管理解决方案，要求如下

1. **用户口令采用密文存储，采用不可逆加密算法加密，查资料了解现有的不可逆算法。**
2. **设计用户修改密码流程，用VISO画出流程图**
3. **设计用户密码重置业务流程（可以包含线下流程），用VISO画出业务流程图**

2.1 密码密文存储与不可逆加密算法

核心原则: 为了保障用户账户安全，用户的原始密码**绝对不能**以明文形式存储在数据库中。必须使用**不可逆加密算法**（也称为哈希函数）将用户密码转换成一段固定长度的、不可逆推回原始密码的字符串（哈希值），然后将这个哈希值存储在数据库中。

不可逆加密算法介绍:

哈希函数是一种将任意长度的数据（密码）映射为固定长度数据的算法。理想的密码哈希函数具有以下特点：

- **单向性 (不可逆):** 从哈希值 H(密码) 无法或极难反向计算出原始密码。这是与可逆加密（如 AES）最根本的区别。
- **确定性:** 对同一个输入（密码），总是产生相同的哈希值。
- **雪崩效应:** 输入的微小变化（密码改动一个字母）会导致输出的哈希值发生巨大、不可预测的变化。
- **抗碰撞性:** 难以找到两个不同的输入（密码）产生相同的哈希值。对于密码哈希，更重要的是**抗原像攻击**（给定哈希值找密码）和**抗第二原像攻击**（给定密码找另一个密码产生相同哈希值）。
- **计算缓慢 (可选但推荐):** 对于密码哈希，故意设计得计算起来比较慢（需要消耗一定的 CPU 或内存资源）可以有效抵抗暴力破解和彩虹表攻击。

现有的不可逆加密算法:

- **MD5, SHA-1: 已不安全，绝对不应使用。**它们计算速度过快，且已被发现存在严重的碰撞漏洞，容易被破解。
- **SHA-2 (SHA-256, SHA-512):** 比 MD5/SHA-1 安全得多，是通用的哈希算法。但它们同样设计得很快，如果单独使用（没有加盐或足够多的迭代），仍然容易受到暴力破解攻击。如果必须使用，**必须**配合强大的**盐值 (Salt)** 和**密钥派生函数 (KDF)**（如 PBKDF2）。

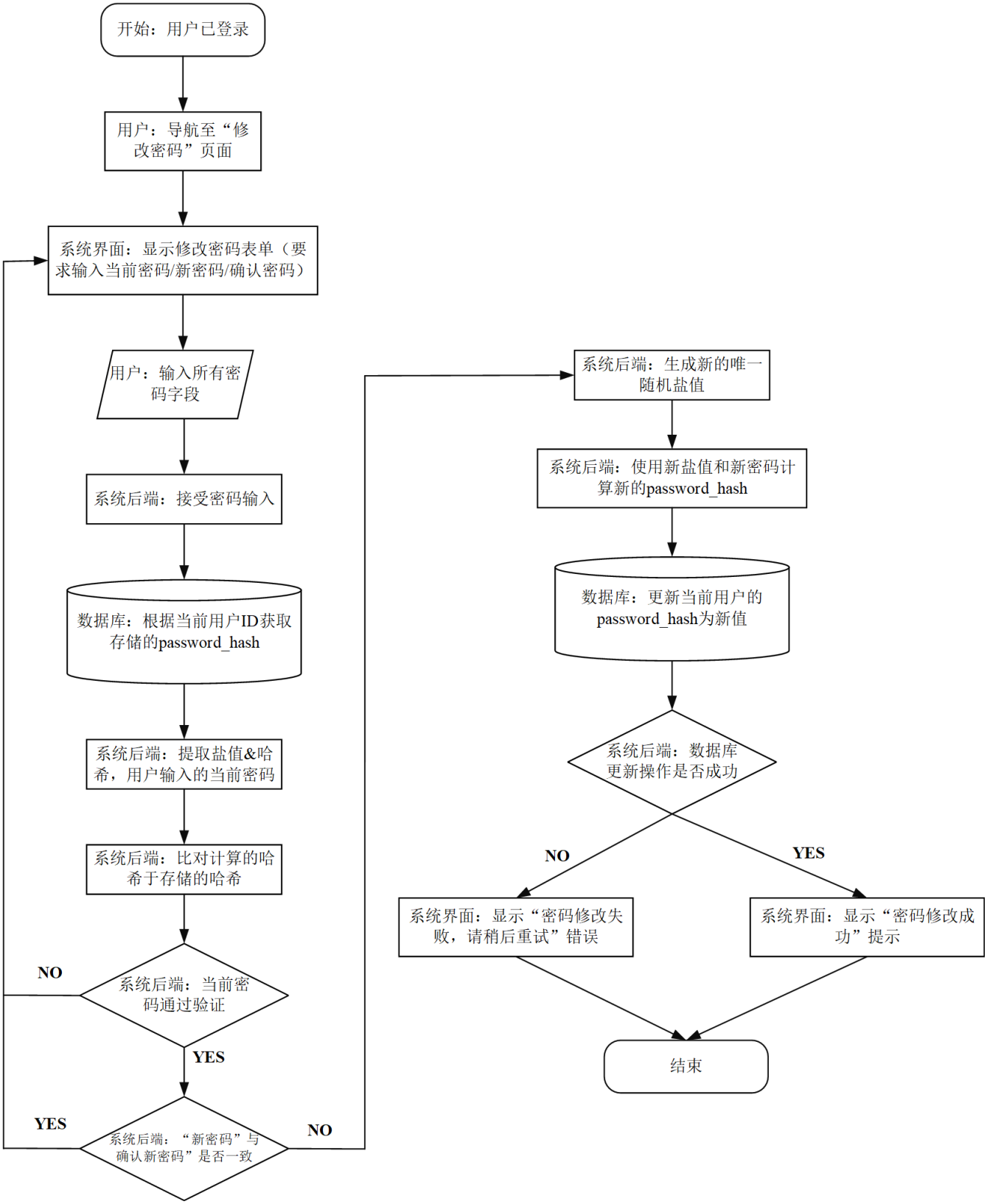
- Argon2:目前最优算法 。它是 2015 年密码哈希竞赛 (Password Hashing Competition) 的获胜者。
    - **优点:** Argon2 提供了高度的灵活性，可以调整时间成本（CPU 消耗）、内存成本和并行度。它对各种攻击（包括 GPU 破解、侧信道攻击）都有很好的抵抗力。它有几种变体，**Argon2id** (混合模式) 通常被认为是最佳选择，结合了 Argon2d 和 Argon2i 的优点。同样内置盐值和自适应调整。

关键技术：加盐 (Salting)

- **什么是盐 (Salt)?** 盐是一个为每个用户密码生成的、**唯一**的、随机的字符串。
- **为什么需要盐?**:
    - 防止**彩虹表攻击**: 如果不加盐，两个使用相同密码的用户将拥有相同的哈希值。攻击者可以预先计算常用密码的哈希值（构成彩虹表），然后快速查找匹配。加盐后，即使密码相同，由于盐值不同，最终的哈希值也完全不同，彩虹表失效。
    - 增加破解难度: 即使两个用户密码一样，攻击者也需要分别为每个用户（因为盐不同）进行破解尝试。
- **如何使用盐?** 在对用户密码进行哈希计算**之前**，将该用户**独有的盐值**与密码结合（通常是拼接）。
- **存储盐值:** 生成的盐值必须和密码哈希一起存储在数据库中。盐值不需要保密，它是公开的。在验证密码时需要用到它。

2.2 设计用户修改密码流程，用VISO画出流程图

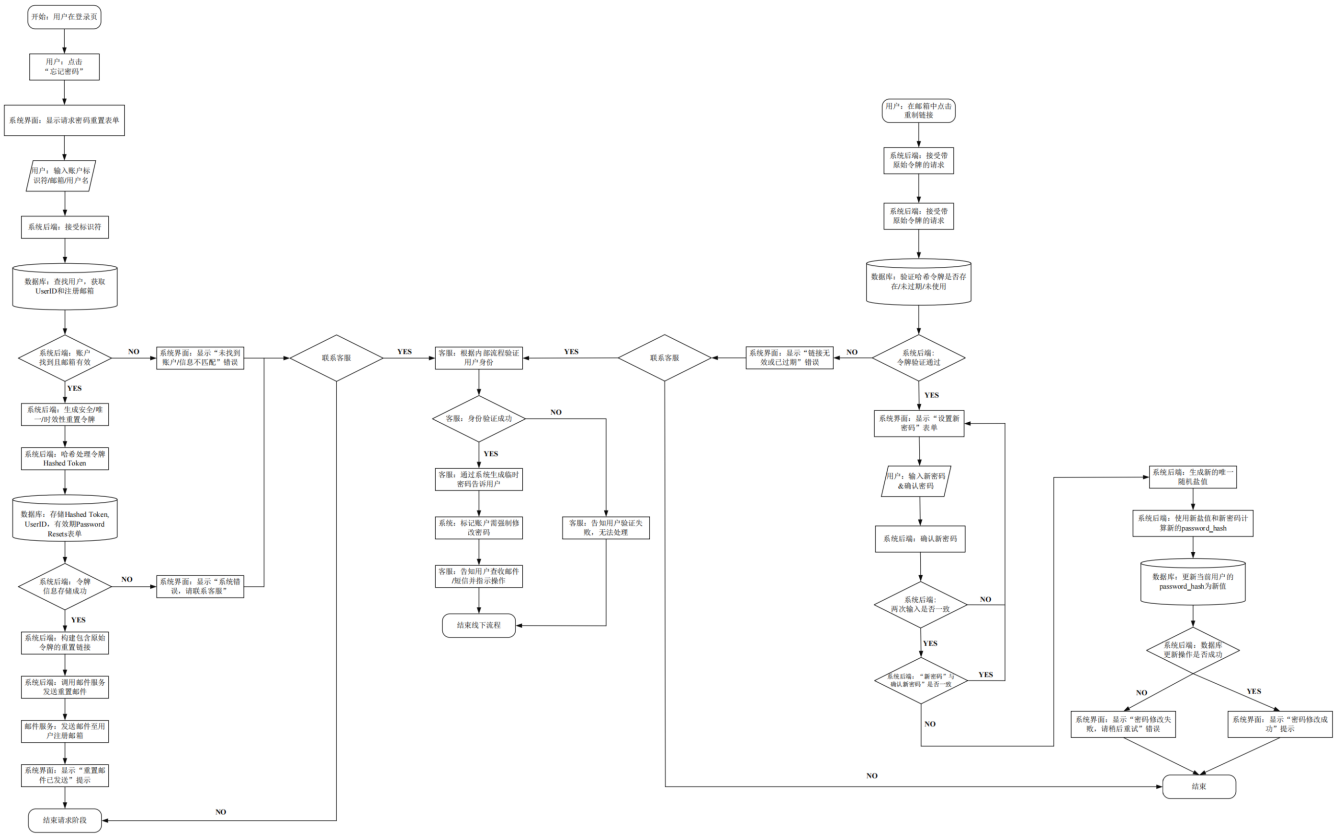
当用户登录系统后，可以通过“修改密码”功能来更新自己的账户密码。用户需要先输入当前的旧密码以验证身份，然后输入两次相同的新密码。系统会先校验旧密码是否正确，再检查两次输入的新密码是否一致。所有验证通过后，系统会为新密码生成一个全新的、唯一的盐值，并使用安全哈希算法计算出新的密码哈希值，最后将这个包含新盐值的新哈希更新到数据库中，替换掉旧的哈希，并向用户提示修改成功或失败。


开始：用户已登录
用户：导航至“修改密码”页面
系统界面：显示修改密码表单（要求输入当前密码/新密码/确认密码）
用户：输入所有密码字段
系统后端：接受密码输入
数据库：根据当前用户ID获取存储的password_hash
系统后端：提取盐值&哈希，用户输入的当前密码
系统后端：比对计算的哈希于存储的哈希
系统后端：当前密码通过验证
NO
YES
系统后端：“新密码”与确认新密码”是否一致
YES
NO
系统后端：生成新的唯一随机盐值
系统后端：使用新盐值和新密码计算新的password_hash
数据库：更新当前用户的password_hash为新值
系统后端：数据库更新操作是否成功
NO
YES
系统界面：显示“密码修改失败，请稍后重试”错误
系统界面：显示“密码修改成功”提示
结束

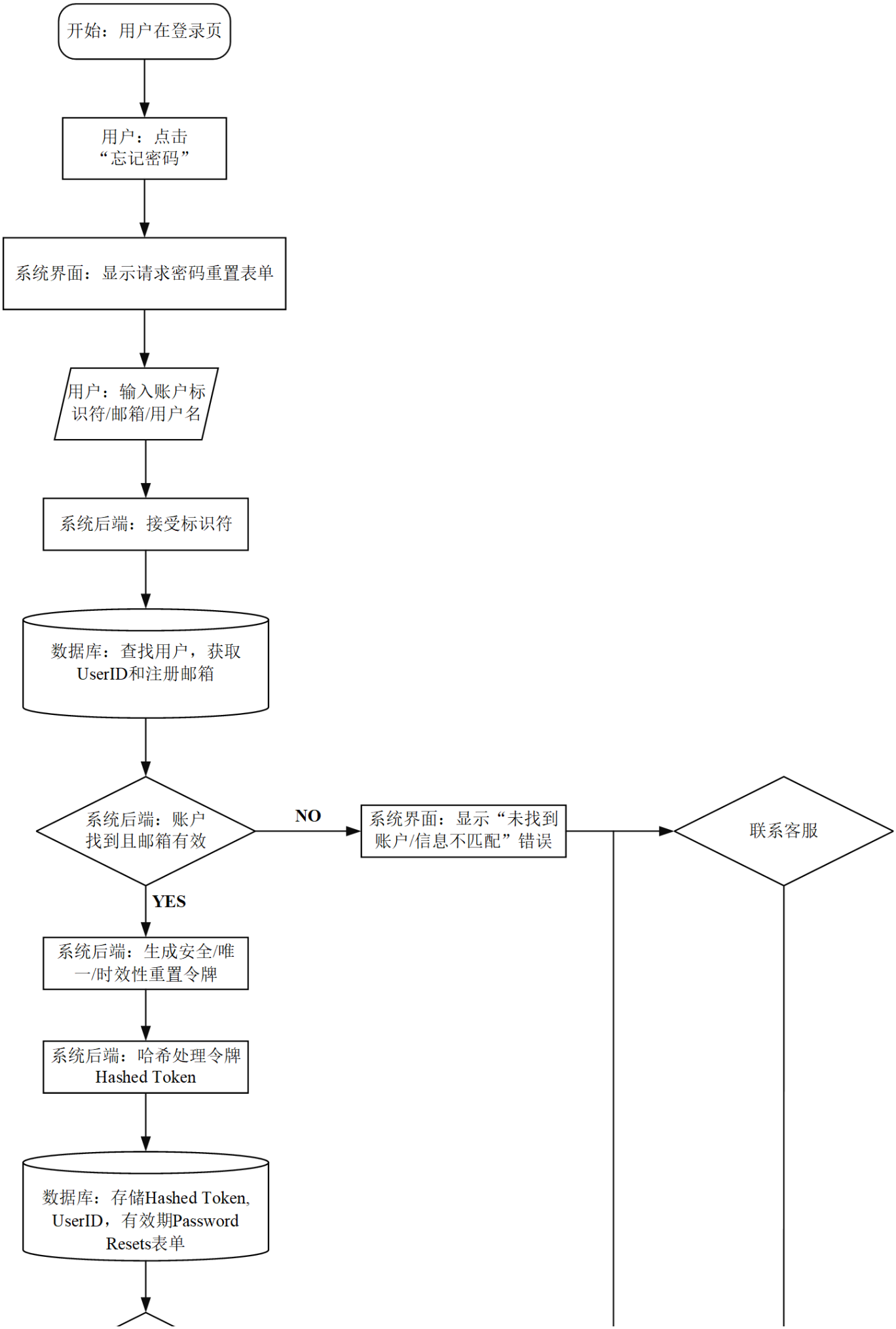
2.3 设计用户密码重置业务流程,用VISO画出业务流程图

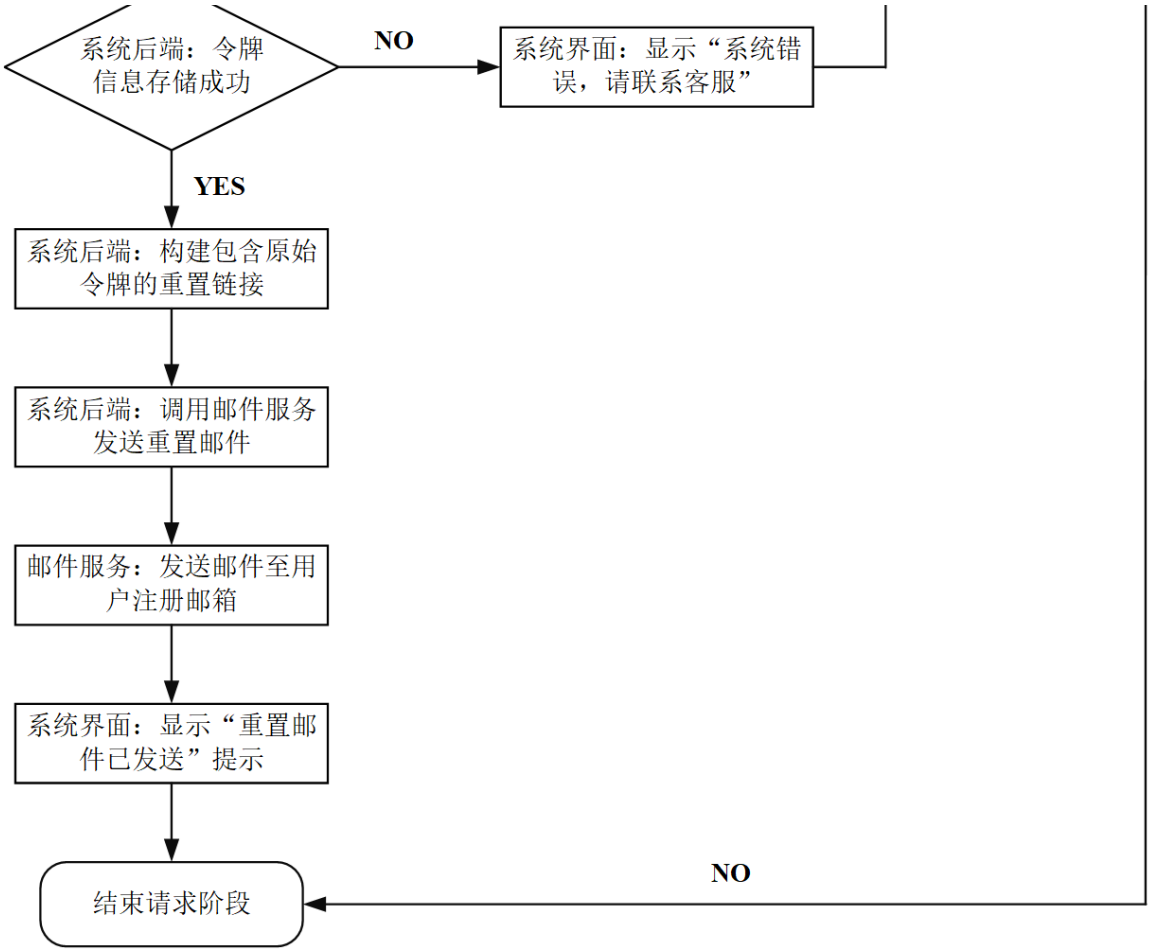
在用户密码重置业务流程中，我一共设计了三个阶段：

用户在忘记密码无法登录时，可以启动密码重置流程。用户首先在登录页面点击“忘记密码”，并输入注册时使用的邮箱或用户名等标识信息。系统验证账户存在后，会生成一个一次性、有时效性的安全重置令牌，将其哈希值存储在数据库，并将包含原始令牌的重置链接发送到用户的注册邮箱。用户点击邮件中的链接后，系统会验证令牌的有效性（是否存在、未过期、未使用）。验证通过，用户即可进入设置新密码页面（无需输入旧密码），输入并确认新密码。系统校验新密码通过后，同样会生成新盐值、计算新哈希，并更新数据库中的密码记录，同时使刚才使用的重置令牌失效。如果在线方式失败或用户无法访问邮箱，流程通常还包含联系客服进行线下身份验证和协助重置的选项。

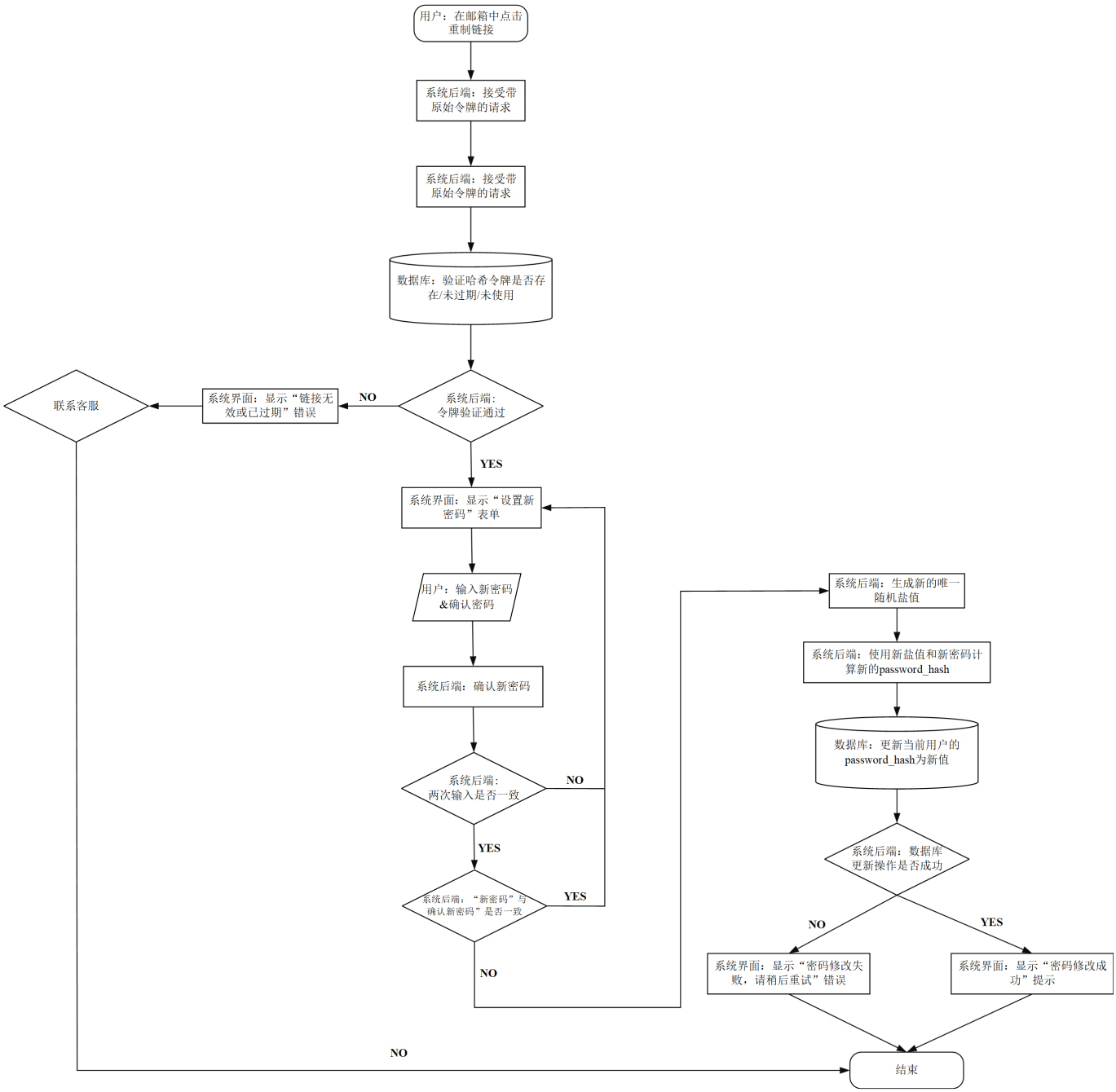


阶段一：用户请求重置（在线 - 邮件）

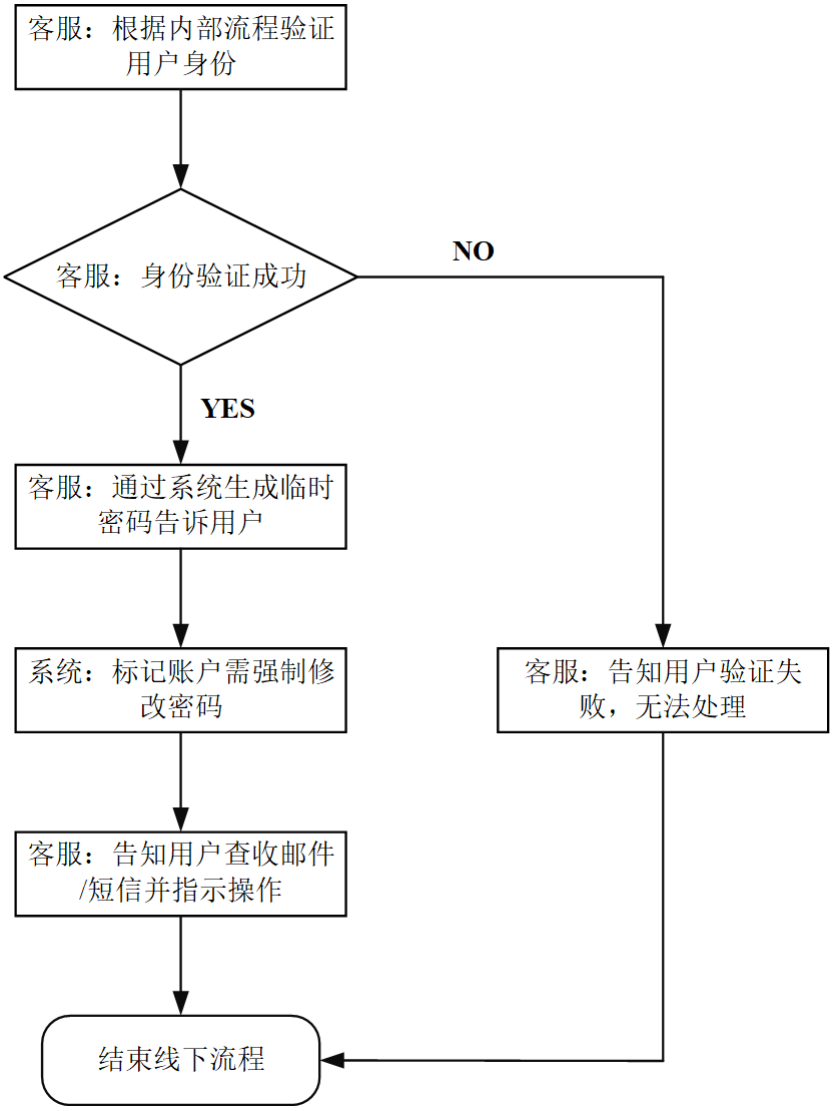

开始：用户在登录页
用户：点击“忘记密码”
系统界面：显示请求密码重置表单
用户：输入账户标识符/邮箱/用户名
系统后端：接受标识符
数据库：查找用户，获取UserID和注册邮箱
系统后端：账户找到且邮箱有效
NO
系统界面：显示“未找到账户/信息不匹配”错误
联系客服
YES
系统后端：生成安全/唯一/时效性重置令牌
系统后端：哈希处理令牌Hashed Token
数据库：存储Hashed Token, UserID，有效期Password Resets表单



阶段二：用户点击链接设置新密码（在线 - 邮件）


用户：在邮箱中点击重制链接
系统后端：接受带原始令牌的请求
系统后端：接受带原始令牌的请求
数据库：验证哈希令牌是否存在/未过期/未使用
系统后端：令牌验证通过
NO
系统界面：显示“链接无效或已过期”错误
联系客服
YES
系统界面：显示“设置新密码”表单
用户：输入新密码&确认密码
系统后端：确认新密码
系统后端：两次输入是否一致
NO
YES
系统后端：“新密码”与确认新密码”是否一致
YES
NO
系统后端：生成新的唯一随机盐值
系统后端：使用新盐值和新密码计算新的password_hash
数据库：更新当前用户的password_hash为新值
系统后端：数据库更新操作是否成功
NO
YES
系统界面：显示“密码修改失败，请稍后重试”错误
系统界面：显示“密码修改成功”提示
NO
结束

分支：线下处理流程



三、用户授权操作题

为你在作业#2中选择的具体案例增加若干个数据库用户，并对各个用户进行相应的授权操作，至少应完成如下操作：

3.1 创建用户

```
-- ----------------------------------------
-- 1) 创建用户
-- ----------------------------------------
-- 创建几个不同角色的用户，密码

-- 创建一个票务代理用户
CREATE USER 'ticket_agent'@'localhost' IDENTIFIED BY 'AgentPass123!';

-- 创建一个数据分析师用户
CREATE USER 'data_analyst'@'localhost' IDENTIFIED BY
'AnalystPass456!';

-- 创建一个航班管理员用户
CREATE USER 'flight_manager'@'localhost' IDENTIFIED BY
'ManagerPass789!';
```

3.2 为不同的用户授予和收回表级SELECT, INSERT, UPDATE, DELETE, ALL PRIVILEGES权限

3.2.1 为 ticket_agent 授予权限

```
-- -- 为 ticket_agent 授予权限 --
-- 允许查询航班和乘客信息
GRANT SELECT ON AirlineDB.Flight TO 'ticket_agent'@'localhost';
GRANT SELECT ON AirlineDB.Passenger TO 'ticket_agent'@'localhost';
-- 允许创建和查询预订信息
GRANT SELECT, INSERT ON AirlineDB.Booking TO
'ticket_agent'@'localhost';
-- 允许更新预订的支付状态
GRANT UPDATE ON AirlineDB.Booking TO 'ticket_agent'@'localhost';
```

3.2.2 为 data_analyst 授予权限

```sql
-- -- 为 data_analyst 授予权限 --
-- 允许查询数据库中的所有表 (只读权限)
GRANT SELECT ON AirlineDB.* TO 'data_analyst'@'localhost'; -- * 代表所有表
```

3.2.3 为 flight_manager 授予权限

```sql
-- -- 为 flight_manager 授予权限 --
-- 允许管理航班和航班状态
GRANT SELECT, INSERT, UPDATE ON AirlineDB.Flight TO 'flight_manager'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON AirlineDB.FlightStatus TO 'flight_manager'@'localhost';
-- 允许查询相关信息
GRANT SELECT ON AirlineDB.Airline TO 'flight_manager'@'localhost';
GRANT SELECT ON AirlineDB.Route TO 'flight_manager'@'localhost';
-- 授予对 Airport 表的所有权限-----> ALL PRIVILEGES权限
GRANT ALL PRIVILEGES ON AirlineDB.Airport TO 'flight_manager'@'localhost';
```

3.2.4 演示收回权限

```sql
-- 假设不再允许票务代理直接更新整个 Booking 表，收回 UPDATE 权限
REVOKE UPDATE ON AirlineDB.Booking FROM 'ticket_agent'@'localhost';

-- 收回 flight_manager 对 Airport 表的 DELETE 权限 (从 ALL PRIVILEGES 中移除)
REVOKE DELETE ON AirlineDB.Airport FROM 'flight_manager'@'localhost';
```

3.3 为不同的用户授予和收回属性列级的SELECT, INSERT, UPDATE的权限

```sql
-- -- 为 ticket_agent 授予列级权限 --
-- 允许票务代理仅查询乘客的姓名和电话号码
GRANT SELECT (passenger_id, name, phone_number) ON AirlineDB.Passenger TO 'ticket_agent'@'localhost';
```

```
-- (注意：如果之前授予了整个表的 SELECT，需要先 REVOKE 整个表的 SELECT，再 GRANT 列级 SELECT)
-- 因此，我们先收回之前对 Passenger 表的 SELECT 权限
REVOKE SELECT ON AirlineDB.Passenger FROM 'ticket_agent'@'localhost';
-- 再授予列级 SELECT
GRANT SELECT (passenger_id, name, phone_number) ON AirlineDB.Passenger TO 'ticket_agent'@'localhost';

-- 允许票务代理仅更新 Booking 表的支付状态
-- (如果之前收回了表级 UPDATE，现在可以授予列级 UPDATE)
GRANT UPDATE (payment_status) ON AirlineDB.Booking TO 'ticket_agent'@'localhost';

-- -- 演示收回列级权限 --
-- 收回票务代理更新 Booking 表支付状态的权限
REVOKE UPDATE (payment_status) ON AirlineDB.Booking FROM 'ticket_agent'@'localhost';
```

3.4 创建角色，并把角色授予用户或收回用户的某个角色

```
-- -- 创建角色 --
CREATE ROLE 'ReadOnlyRole'@'localhost';
CREATE ROLE 'BookingAgentRole'@'localhost';
CREATE ROLE 'FlightAdminRole'@'localhost';

-- -- 为角色授权 --
-- 为只读角色授予所有表的 SELECT 权限
GRANT SELECT ON AirlineDB.* TO 'ReadOnlyRole'@'localhost';

-- 为预订代理角色授权
GRANT SELECT ON AirlineDB.Flight TO 'BookingAgentRole'@'localhost';
GRANT SELECT (passenger_id, name, phone_number, email) ON AirlineDB.Passenger TO 'BookingAgentRole'@'localhost'; -- 授予查询乘客部分信息的权限
GRANT SELECT, INSERT ON AirlineDB.Booking TO 'BookingAgentRole'@'localhost';
GRANT UPDATE (payment_status) ON AirlineDB.Booking TO 'BookingAgentRole'@'localhost'; -- 允许更新支付状态
GRANT SELECT ON AirlineDB.Route TO 'BookingAgentRole'@'localhost'; -- 可能需要看航线信息
```

```sql

-- 为航班管理角色授权
GRANT SELECT, INSERT, UPDATE, DELETE ON AirlineDB.Flight TO 'FlightAdminRole'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON AirlineDB.FlightStatus TO 'FlightAdminRole'@'localhost';
GRANT SELECT ON AirlineDB.Airline TO 'FlightAdminRole'@'localhost';
GRANT SELECT ON AirlineDB.Route TO 'FlightAdminRole'@'localhost';
GRANT SELECT, UPDATE, INSERT, DELETE ON AirlineDB.Airport TO 'FlightAdminRole'@'localhost'; -- 之前授权给用户的 ALL PRIVILEGES 包含更多，这里给角色具体权限

-- -- 将角色授予用户 --
GRANT 'ReadOnlyRole'@'localhost' TO 'data_analyst'@'localhost';
GRANT 'BookingAgentRole'@'localhost' TO 'ticket_agent'@'localhost';
GRANT 'FlightAdminRole'@'localhost' TO 'flight_manager'@'localhost';

-- 设置用户的默认角色 (可选，登录后默认激活的角色)
-- SET DEFAULT ROLE 'ReadOnlyRole'@'localhost' TO 'data_analyst'@'localhost';
-- SET DEFAULT ROLE 'BookingAgentRole'@'localhost' TO 'ticket_agent'@'localhost';
-- SET DEFAULT ROLE 'FlightAdminRole'@'localhost' TO 'flight_manager'@'localhost';
-- 或者设置所有角色为默认
SET DEFAULT ROLE ALL TO 'data_analyst'@'localhost', 'ticket_agent'@'localhost', 'flight_manager'@'localhost';


-- -- 演示从用户收回角色 --
-- 假设 data_analyst 角色变更，不再需要只读权限
REVOKE 'ReadOnlyRole'@'localhost' FROM 'data_analyst'@'localhost';
```

3.5 上述过程均编写为单独脚本，实验结果如下

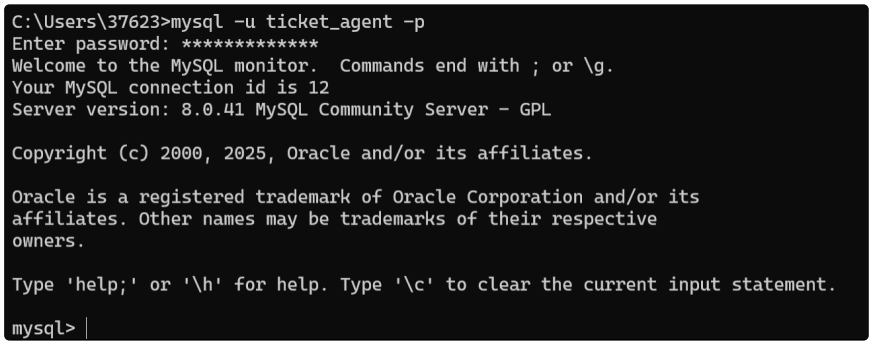
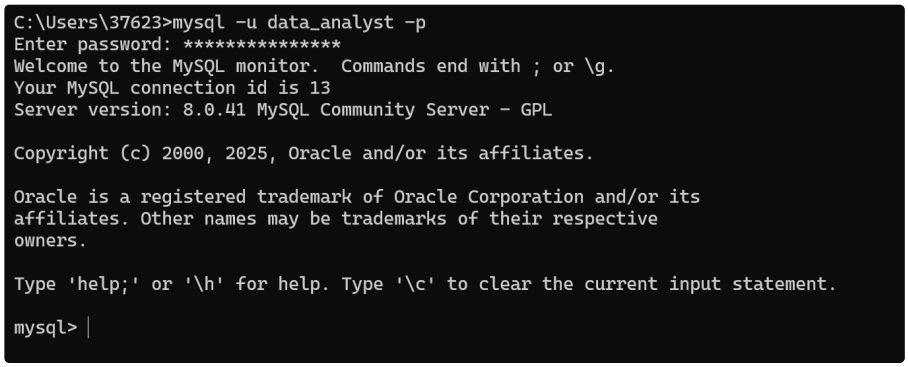
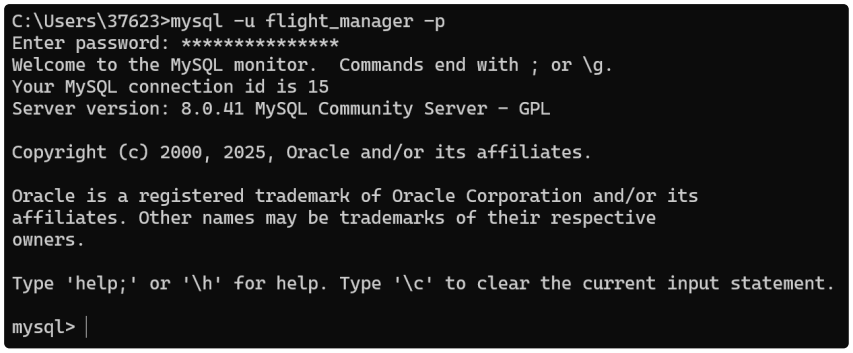
执行脚本

```
mysql -u root -p
mysql> SOURCE /path/to/DBLab4_Permissions_22281188.sql;
e.g. C:/Users/37623/Desktop/Lab5/DBLab4_Permissions_22281188.sql;
```

下面测试语句应该在mysql客户端实现：

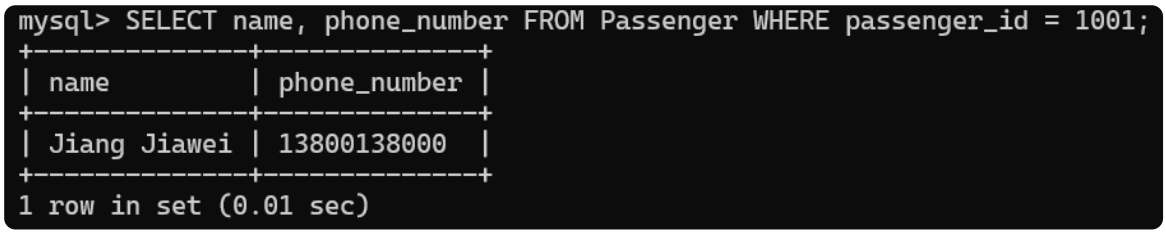
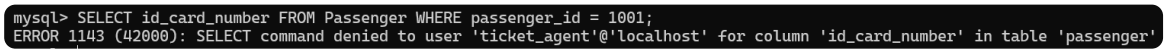
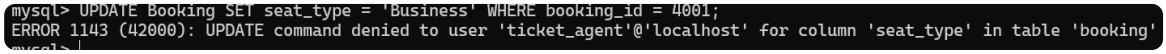
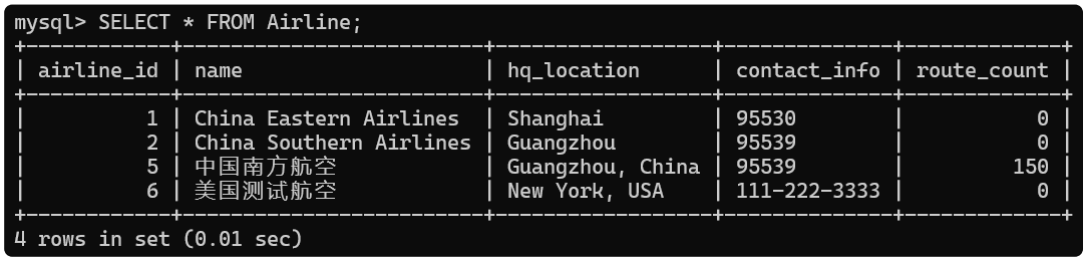
3.5.1登录测试

尝试分别使用 'ticket_agent'@'localhost', 'data_analyst'@'localhost', 'flight_manager'@'localhost' 和对应的密码登录 MySQL

- `mysql -u ticket_agent -p` (输入 AgentPass123!)
  - 
    ```
    C:\Users\37623>mysql -u ticket_agent -p
    Enter password: *************
    Welcome to the MySQL monitor.  Commands end with ; or \g.
    Your MySQL connection id is 12
    Server version: 8.0.41 MySQL Community Server - GPL

    Copyright (c) 2000, 2025, Oracle and/or its affiliates.

    Oracle is a registered trademark of Oracle Corporation and/or its
    affiliates. Other names may be trademarks of their respective
    owners.

    Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

    mysql>
    ```
- `mysql -u data_analyst -p` (输入 AnalystPass456!)
  - 
    ```
    C:\Users\37623>mysql -u data_analyst -p
    Enter password: ***************
    Welcome to the MySQL monitor.  Commands end with ; or \g.
    Your MySQL connection id is 13
    Server version: 8.0.41 MySQL Community Server - GPL

    Copyright (c) 2000, 2025, Oracle and/or its affiliates.

    Oracle is a registered trademark of Oracle Corporation and/or its
    affiliates. Other names may be trademarks of their respective
    owners.

    Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

    mysql>
    ```
- `mysql -u flight_manager -p` (输入 ManagerPass789!)
  - 
    ```
    C:\Users\37623>mysql -u flight_manager -p
    Enter password: ***************
    Welcome to the MySQL monitor.  Commands end with ; or \g.
    Your MySQL connection id is 15
    Server version: 8.0.41 MySQL Community Server - GPL

    Copyright (c) 2000, 2025, Oracle and/or its affiliates.

    Oracle is a registered trademark of Oracle Corporation and/or its
    affiliates. Other names may be trademarks of their respective
    owners.

    Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

    mysql>
    ```

3.5.2 权限测试 (登录后)

- **对于 ticket_agent:**
  - `USE AirlineDB;`
  - 尝试 `SELECT * FROM Flight;` (成功)

    - 
      ```
      mysql> USE AirlineDB;
      Database changed
      mysql> SELECT * FROM Flight;
      +-----------+------------+----------+---------------------+---------------------+-----------------------------------+-----------------------------------+-------------+-------------+----------------+
      | flight_id | airline_id | route_id | departure_time      | arrival_time        | departure_location                | destination_location              | total_seats | booked_seats | aircraft_model |
      +-----------+------------+----------+---------------------+---------------------+-----------------------------------+-----------------------------------+-------------+-------------+----------------+
      |      3001 |          1 |      201 | 2025-07-15 08:00:00 | 2025-07-15 10:10:00 | Shanghai Hongqiao International Airport | Beijing Capital International Airport |         180 |           0 | A320           |
      |      3002 |          2 |      202 | 2025-07-15 09:30:00 | 2025-07-15 11:30:00 | Guangzhou Baiyun International Airport | Shanghai Hongqiao International Airport |         220 |           0 | B737           |
      +-----------+------------+----------+---------------------+---------------------+-----------------------------------+-----------------------------------+-------------+-------------+----------------+
      2 rows in set (0.01 sec)
      ```
  - 尝试 `SELECT name, phone_number FROM Passenger WHERE passenger_id = 1001;` (成功)
    - 
      ```
      mysql> SELECT name, phone_number FROM Passenger WHERE passenger_id = 1001;
      +--------------+--------------+
      | name         | phone_number |
      +--------------+--------------+
      | Jiang Jiawei | 13800138000  |
      +--------------+--------------+
      1 row in set (0.01 sec)
      ```
  - 尝试 `SELECT id_card_number FROM Passenger WHERE passenger_id = 1001;` (失败 - 无此列权限)
    - 
      ```
      mysql> SELECT id_card_number FROM Passenger WHERE passenger_id = 1001;
      ERROR 1143 (42000): SELECT command denied to user 'ticket_agent'@'localhost' for column 'id_card_number' in table 'passenger'
      ```
  - 尝试 `SELECT * FROM Airline;` (失败 - 无此表权限)
    - 
      ```
      mysql> SELECT * FROM Airline;
      ERROR 1142 (42000): SELECT command denied to user 'ticket_agent'@'localhost' for table 'airline'
      ```
  - 尝试 `UPDATE Booking SET seat_type = 'Business' WHERE booking_id = 4001;` (失败 - 无此列 UPDATE 权限，除非重新授权)
    - 
      ```
      mysql> UPDATE Booking SET seat_type = 'Business' WHERE booking_id = 4001;
      ERROR 1143 (42000): UPDATE command denied to user 'ticket_agent'@'localhost' for column 'seat_type' in table 'booking'
      ```
  - 尝试 `DELETE FROM Flight WHERE flight_id = 3001;` (应失败 - 无 DELETE 权限)
    - 
      ```
      mysql> DELETE FROM Flight WHERE flight_id = 3001;
      ERROR 1142 (42000): DELETE command denied to user 'ticket_agent'@'localhost' for table 'flight'
      ```
- **对于 data_analyst:**
  - `USE AirlineDB;`
  - 尝试 `SELECT * FROM Flight;` (成功)
    - 
      ```
      mysql> USE AirlineDB;
      Database changed
      mysql> SELECT * FROM Flight;
      +-----------+------------+----------+---------------------+---------------------+-----------------------------------+-----------------------------------+-------------+-------------+----------------+
      | flight_id | airline_id | route_id | departure_time      | arrival_time        | departure_location                | destination_location              | total_seats | booked_seats | aircraft_model |
      +-----------+------------+----------+---------------------+---------------------+-----------------------------------+-----------------------------------+-------------+-------------+----------------+
      |      3001 |          1 |      201 | 2025-07-15 08:00:00 | 2025-07-15 10:10:00 | Shanghai Hongqiao International Airport | Beijing Capital International Airport |         180 |           0 | A320           |
      |      3002 |          2 |      202 | 2025-07-15 09:30:00 | 2025-07-15 11:30:00 | Guangzhou Baiyun International Airport | Shanghai Hongqiao International Airport |         220 |           0 | B737           |
      +-----------+------------+----------+---------------------+---------------------+-----------------------------------+-----------------------------------+-------------+-------------+----------------+
      2 rows in set (0.00 sec)
      ```
  - 尝试 `SELECT * FROM Airline;` (成功)
    - 
      ```
      mysql> SELECT * FROM Airline;
      +------------+--------------------------+-----------------+--------------+-------------+
      | airline_id | name                     | hq_location     | contact_info | route_count |
      +------------+--------------------------+-----------------+--------------+-------------+
      |          1 | China Eastern Airlines   | Shanghai        | 95530        |           0 |
      |          2 | China Southern Airlines  | Guangzhou       | 95539        |           0 |
      |          5 | 中国南方航空             | Guangzhou, China | 95539        |         150 |
      |          6 | 美国测试航空             | New York, USA   | 111-222-3333 |           0 |
      +------------+--------------------------+-----------------+--------------+-------------+
      4 rows in set (0.01 sec)
      ```

- 尝试 `UPDATE Flight SET booked_seats = 10 WHERE flight_id = 3001;` (失败)

```
mysql> UPDATE Flight SET booked_seats = 10 WHERE flight_id = 3001;
ERROR 1142 (42000): UPDATE command denied to user 'data_analyst'@'localhost' for table 'flight'
mysql>
```

- **对于 flight_manager:**
  - `USE AirlineDB;`
  - 尝试 `SELECT * FROM Flight;` (成功)

```
mysql> SELECT * FROM Flight;
+-----------+------------+----------+---------------------+---------------------+-----------------------------------+-----------------------------------+-------------+-------------+----------------+
| flight_id | airline_id | route_id | departure_time      | arrival_time        | departure_location                | destination_location              | total_seats | booked_seats | aircraft_model |
+-----------+------------+----------+---------------------+---------------------+-----------------------------------+-----------------------------------+-------------+-------------+----------------+
|      3001 |          1 |      201 | 2025-07-15 08:00:00 | 2025-07-15 10:10:00 | Shanghai Hongqiao International Airport | Beijing Capital International Airport |         180 |           0 | A320           |
|      3002 |          2 |      202 | 2025-07-15 09:30:00 | 2025-07-15 11:30:00 | Guangzhou Baiyun International Airport | Shanghai Hongqiao International Airport |         220 |           0 | B737           |
+-----------+------------+----------+---------------------+---------------------+-----------------------------------+-----------------------------------+-------------+-------------+----------------+
2 rows in set (0.00 sec)
```

  - 尝试 `UPDATE Flight SET booked_seats = 50 WHERE flight_id = 3001;` (成功)

```
mysql> UPDATE Flight SET booked_seats = 50 WHERE flight_id = 3001;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

  - 尝试 `SELECT * FROM Airport;` (成功)

```
mysql> SELECT * FROM Airport;
+------------+-----------------------------------------+------------------+---------------+
| airport_id | airport_name                            | location         | flight_count |
+------------+-----------------------------------------+------------------+---------------+
|        101 | Shanghai Hongqiao International Airport | Shanghai, China  |             0 |
|        102 | Guangzhou Baiyun International Airport  | Guangzhou, China |             0 |
|        103 | Beijing Capital International Airport   | Beijing, China   |             0 |
+------------+-----------------------------------------+------------------+---------------+
3 rows in set (0.00 sec)
```

  - 尝试 `DELETE FROM Airport WHERE airport_id = 103;` (成功)

```
mysql> DELETE FROM Airport WHERE airport_id = 103;
Query OK, 1 row affected (0.01 sec)
```

  - 尝试 `SELECT * FROM Passenger;` (失败 - 未授权)

```
mysql> SELECT * FROM Passenger;
ERROR 1142 (42000): SELECT command denied to user 'flight_manager'@'localhost' for table 'passenger'
mysql>
```

3.5.3 登录root用户查看权限

- `SHOW GRANTS FOR 'ticket_agent'@'localhost';`

```
mysql> SHOW GRANTS FOR 'ticket_agent'@'localhost';
+-----------------------------------------------------------------------------------------------------------------------------------+
| Grants for ticket_agent@localhost                                                                                                 |
+-----------------------------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `ticket_agent`@`localhost`                                                                                  |
| GRANT SELECT, INSERT ON `airlinedb`.`booking` TO `ticket_agent`@`localhost`                                                       |
| GRANT SELECT ON `airlinedb`.`flight` TO `ticket_agent`@`localhost`                                                                |
| GRANT SELECT (`name`, `passenger_id`, `phone_number`) ON `airlinedb`.`passenger` TO `ticket_agent`@`localhost`                    |
| GRANT `BookingAgentRole`@`localhost` TO `ticket_agent`@`localhost`                                                                |
+-----------------------------------------------------------------------------------------------------------------------------------+
5 rows in set (0.00 sec)
```

- `SHOW GRANTS FOR 'data_analyst'@'localhost';`

```
mysql> SHOW GRANTS FOR 'data_analyst'@'localhost';
+-----------------------------------------------------------------+
| Grants for data_analyst@localhost                               |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO `data_analyst`@`localhost`                |
| GRANT SELECT ON `airlinedb`.* TO `data_analyst`@`localhost`     |
+-----------------------------------------------------------------+
2 rows in set (0.00 sec)
```

- `SHOW GRANTS FOR 'flight_manager'@'localhost';`

```
mysql> SHOW GRANTS FOR 'flight_manager'@'localhost'
    -> ;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Grants for flight_manager@localhost                                                                                                                                   |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `flight_manager`@`localhost`                                                                                                                    |
| GRANT SELECT ON `airlinedb`.`airline` TO `flight_manager`@`localhost`                                                                                                 |
| GRANT SELECT, INSERT, UPDATE, CREATE, DROP, REFERENCES, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER ON `airlinedb`.`airport` TO `flight_manager`@`localhost`        |
| GRANT SELECT, INSERT, UPDATE ON `airlinedb`.`flight` TO `flight_manager`@`localhost`                                                                                  |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `airlinedb`.`flightstatus` TO `flight_manager`@`localhost`                                                                    |
| GRANT SELECT ON `airlinedb`.`route` TO `flight_manager`@`localhost`                                                                                                   |
| GRANT `FlightAdminRole`@`localhost` TO `flight_manager`@`localhost`                                                                                                   |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
7 rows in set (0.00 sec)
```

- `SHOW GRANTS FOR 'ReadOnlyRole'@'localhost';`

```
mysql> SHOW GRANTS FOR 'ReadOnlyRole'@'localhost';
+-----------------------------------------------------------------+
| Grants for ReadOnlyRole@localhost                               |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO `ReadOnlyRole`@`localhost`                |
| GRANT SELECT ON `airlinedb`.* TO `ReadOnlyRole`@`localhost`     |
+-----------------------------------------------------------------+
2 rows in set (0.00 sec)
```

- `SHOW GRANTS FOR 'ticket_agent'@'localhost' USING 'BookingAgentRole'@'localhost';` (查看通过角色获得的权限)

```
mysql> SHOW GRANTS FOR 'ticket_agent'@'localhost' USING 'BookingAgentRole'@'localhost';
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Grants for ticket_agent@localhost                                                                                                                                     |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `ticket_agent`@`localhost`                                                                                                                      |
| GRANT SELECT, INSERT, UPDATE (`payment_status`) ON `airlinedb`.`booking` TO `ticket_agent`@`localhost`                                                                |
| GRANT SELECT ON `airlinedb`.`flight` TO `ticket_agent`@`localhost`                                                                                                    |
| GRANT SELECT (`email`, `name`, `passenger_id`, `phone_number`) ON `airlinedb`.`passenger` TO `ticket_agent`@`localhost`                                               |
| GRANT SELECT ON `airlinedb`.`route` TO `ticket_agent`@`localhost`                                                                                                     |
| GRANT `BookingAgentRole`@`localhost` TO `ticket_agent`@`localhost`                                                                                                    |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
6 rows in set (0.00 sec)
```

四、数据库完整性实验题

完善作业#3中你的数据模型完整性设计，模型中至少应体现如下完整性要求：

1） 实体完整性

2） 参照完整性

3） 用户定义的完整性

4） 对部分完整性约束进行命名

将以上实现代码加入到作业#3 的脚本中并调试通过，用违反相应约束的语句尝试，以验证完整性约束实施效果。注意观察RDBMS平台对完整性的支持程度。

数据库完整性有：

实体完整性：确保每个表都有一个主键，且主键值唯一、非空，通过 `PRIMARY KEY` 和 `NOT NULL` 约束实现。

参照完整性: 确保外键列的值要么等于其引用的父表主键列中的某个值，要么为 `NULL` (如果外键允许为 NULL)。通过 `FOREIGN KEY` 约束实现。

用户定义的完整性：对数据施加特定业务规则的约束，例如值的范围、格式、集合等。常用 `CHECK`、`UNIQUE` (非主键唯一性)、`NOT NULL` (非主键非空)、`DEFAULT` 等实现。

约束命名 (Constraint Naming): 为约束（尤其是 `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`, `UNIQUE`）指定一个有意义的名称，便于管理和错误识别。

以下代码是修改LAB3的代码：

相应部分已在代码处注释增加

```
-- ============================================================
-- DBLab4_UPDATE2LAB3
-- 航空公司数据库模式创建脚本---------------->对于LAB3的更新
-- 目标平台: 通用 SQL (主要基于 MySQL 语法测试)
-- 作者: [江家玮/北京交通大学]
-- 日期: 2025-04-19
-- ============================================================

-- 不存在则创建
-- CREATE DATABASE IF NOT EXISTS AirlineDB CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
-- USE AirlineDB;

-- 按依赖反序删除表
DROP TABLE IF EXISTS FlightStatus;
DROP TABLE IF EXISTS Ticket;
DROP TABLE IF EXISTS Payment;
DROP TABLE IF EXISTS Booking;
DROP TABLE IF EXISTS Flight;
DROP TABLE IF EXISTS Route;
DROP TABLE IF EXISTS Passenger;
```

```sql
DROP TABLE IF EXISTS Airport;
DROP TABLE IF EXISTS Airline;

-- ============================================================
-- 表: Airline (航空公司)
-- ============================================================
CREATE TABLE Airline (
    airline_id INT NOT NULL COMMENT '航空公司ID',
    name VARCHAR(100) NOT NULL COMMENT '名称',
    hq_location VARCHAR(255) COMMENT '总部位置',
    contact_info VARCHAR(50) COMMENT '联系方式',
    route_count INT DEFAULT 0 COMMENT '航线数量',
    -- 1) 实体完整性 (命名主键约束)
    CONSTRAINT pk_airline PRIMARY KEY (airline_id),
    -- 3) 用户定义完整性 + 4) 约束命名 (航空公司名称唯一)
    CONSTRAINT uq_airline_name UNIQUE (name)
) COMMENT='航空公司信息表';

-- ============================================================
-- 表: Airport (机场)
-- ============================================================
CREATE TABLE Airport (
    airport_id INT NOT NULL COMMENT '机场ID',
    airport_name VARCHAR(100) NOT NULL COMMENT '机场名称',
    location VARCHAR(255) NOT NULL COMMENT '机场位置',
    flight_count INT DEFAULT 0 COMMENT '航班数量 (可能为动态数据)',
    -- 1) 实体完整性 (命名主键约束)
    CONSTRAINT pk_airport PRIMARY KEY (airport_id)
) COMMENT='机场信息表';

-- ============================================================
-- 表: Passenger (乘客)
-- ============================================================
CREATE TABLE Passenger (
    passenger_id INT NOT NULL COMMENT '乘客ID',
    name VARCHAR(100) NOT NULL COMMENT '姓名',
    id_card_number VARCHAR(18) NOT NULL COMMENT '身份证号',
    phone_number VARCHAR(20) NOT NULL COMMENT '联系电话',
    email VARCHAR(100) COMMENT '电子邮件',
    frequent_flyer_number VARCHAR(50) COMMENT '常旅客编号',
    -- 1) 实体完整性 (命名主键约束)
    CONSTRAINT pk_passenger PRIMARY KEY (passenger_id),
    -- 3) 用户定义完整性 + 4) 约束命名 (确保身份证号唯一)
    CONSTRAINT uq_passenger_idcard UNIQUE (id_card_number),
    -- 3) 用户定义完整性 + 4) 约束命名 (确保电子邮件唯一, 如果非空)
```

```sql
    CONSTRAINT uq_passenger_email UNIQUE (email),
    -- 3) 用户定义完整性 + 4) 约束命名 (确保常旅客编号唯一, 如果非空)
    CONSTRAINT uq_passenger_ffn UNIQUE (frequent_flyer_number)
) COMMENT='乘客信息表';

-- ============================================================
-- 表: Route (航线)
-- ============================================================
CREATE TABLE Route (
    route_id INT NOT NULL COMMENT '航线ID',
    airline_id INT NOT NULL COMMENT '航空公司ID (外键)',
    origin VARCHAR(100) NOT NULL COMMENT '起点',
    destination VARCHAR(100) NOT NULL COMMENT '终点',
    duration VARCHAR(20) COMMENT '飞行时长 (e.g., 2h 30m)',
    -- 1) 实体完整性 (命名主键约束)
    CONSTRAINT pk_route PRIMARY KEY (route_id),
    -- 2) 参照完整性 + 4) 约束命名 (外键关联 Airline 表)
    CONSTRAINT fk_route_airline FOREIGN KEY (airline_id) REFERENCES Airline(airline_id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) COMMENT='航线信息表';

-- ============================================================
-- 表: Flight (航班)
-- ============================================================
CREATE TABLE Flight (
    flight_id INT NOT NULL COMMENT '航班ID',
    airline_id INT NOT NULL COMMENT '航空公司ID (外键)',
    route_id INT NOT NULL COMMENT '航线ID (外键)',
    departure_time DATETIME NOT NULL COMMENT '起飞时间',
    arrival_time DATETIME NOT NULL COMMENT '抵达时间',
    departure_location VARCHAR(100) NOT NULL COMMENT '起飞地点',
    destination_location VARCHAR(100) NOT NULL COMMENT '目的地',
    total_seats INT NOT NULL COMMENT '座位总数',
    booked_seats INT DEFAULT 0 COMMENT '已预订座位数',
    aircraft_model VARCHAR(50) COMMENT '机型',
    -- 1) 实体完整性 (命名主键约束)
    CONSTRAINT pk_flight PRIMARY KEY (flight_id),
    -- 2) 参照完整性 + 4) 约束命名 (外键关联 Airline 表)
    CONSTRAINT fk_flight_airline FOREIGN KEY (airline_id) REFERENCES Airline(airline_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    -- 2) 参照完整性 + 4) 约束命名 (外键关联 Route 表)
    CONSTRAINT fk_flight_route FOREIGN KEY (route_id) REFERENCES Route(route_id)
```

```
        ON DELETE RESTRICT ON UPDATE CASCADE,
    -- 3) 用户定义完整性 + 4) 约束命名 (检查座位数)
    CONSTRAINT chk_flight_seats CHECK (booked_seats >= 0 AND
booked_seats <= total_seats AND total_seats > 0),
    -- 3) 用户定义完整性 + 4) 约束命名 (检查起飞到达时间)
    CONSTRAINT chk_flight_times CHECK (arrival_time > departure_time)
) COMMENT='航班信息表';

-- ============================================================
-- 表: Booking (预订)
-- ============================================================
CREATE TABLE Booking (
    booking_id INT NOT NULL COMMENT '预订ID',
    passenger_id INT NOT NULL COMMENT '乘客ID (外键)',
    flight_id INT NOT NULL COMMENT '航班ID (外键)',
    seat_type VARCHAR(20) COMMENT '座位类型 (e.g., Economy, Business)',
    booking_date DATETIME NOT NULL COMMENT '预订日期',
    price DECIMAL(10, 2) NOT NULL COMMENT '票价',
    payment_status VARCHAR(20) NOT NULL DEFAULT 'Pending' COMMENT '支
付状态',
    -- 1) 实体完整性 (命名主键约束)
    CONSTRAINT pk_booking PRIMARY KEY (booking_id),
    -- 2) 参照完整性 + 4) 约束命名 (外键关联 Passenger 表)
    CONSTRAINT fk_booking_passenger FOREIGN KEY (passenger_id)
REFERENCES Passenger(passenger_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    -- 2) 参照完整性 + 4) 约束命名 (外键关联 Flight 表)
    CONSTRAINT fk_booking_flight FOREIGN KEY (flight_id) REFERENCES
Flight(flight_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    -- 3) 用户定义完整性 + 4) 约束命名 (检查价格必须为正)
    CONSTRAINT chk_booking_price CHECK (price > 0),
    -- 3) 用户定义完整性 + 4) 约束命名 (检查支付状态必须在允许范围内)
    CONSTRAINT chk_booking_status CHECK (payment_status IN
('Pending', 'Paid', 'Cancelled'))
) COMMENT='预订信息表';

-- ============================================================
-- 表: Payment (支付)
-- ============================================================
CREATE TABLE Payment (
    payment_id INT NOT NULL COMMENT '支付ID',
    booking_id INT NOT NULL COMMENT '预订ID (外键)',
    payment_method VARCHAR(50) NOT NULL COMMENT '支付方式 (e.g., Credit
Card, Alipay)',
```

```
147     payment_amount DECIMAL(10, 2) NOT NULL COMMENT '支付金额',
148     payment_time DATETIME NOT NULL COMMENT '支付时间',
149     payment_status VARCHAR(20) NOT NULL DEFAULT 'Success' COMMENT '支
    付状态',
150     -- 1) 实体完整性 (命名主键约束)
151     CONSTRAINT pk_payment PRIMARY KEY (payment_id),
152     -- 2) 参照完整性 + 4) 约束命名 (外键关联 Booking 表)
153     CONSTRAINT fk_payment_booking FOREIGN KEY (booking_id) REFERENCES
    Booking(booking_id)
154         ON DELETE RESTRICT ON UPDATE CASCADE,
155     -- 3) 用户定义完整性 + 4) 约束命名 (检查支付金额为正)
156     CONSTRAINT chk_payment_amount CHECK (payment_amount > 0),
157     -- 3) 用户定义完整性 + 4) 约束命名 (检查支付状态)
158     CONSTRAINT chk_payment_status CHECK (payment_status IN
    ('Success', 'Failed', 'Processing')) -- 增加 'Processing' 状态
159 ) COMMENT='支付信息表';
160
161 -- ============================================================
162 -- 表: Ticket (机票)
163 -- ============================================================
164 CREATE TABLE Ticket (
165     ticket_id INT NOT NULL COMMENT '机票ID',
166     flight_id INT NOT NULL COMMENT '航班ID (外键)',
167     ticket_number VARCHAR(50) NOT NULL COMMENT '票号',
168     seat_number VARCHAR(10) NOT NULL COMMENT '座位号',
169     passenger_id INT NOT NULL COMMENT '乘客ID (外键)',
170     price DECIMAL(10, 2) NOT NULL COMMENT '票价',
171     -- 1) 实体完整性 (命名主键约束)
172     CONSTRAINT pk_ticket PRIMARY KEY (ticket_id),
173     -- 3) 用户定义完整性 + 4) 约束命名 (票号唯一)
174     CONSTRAINT uq_ticket_number UNIQUE (ticket_number),
175     -- 2) 参照完整性 + 4) 约束命名 (外键关联 Flight 表)
176     CONSTRAINT fk_ticket_flight FOREIGN KEY (flight_id) REFERENCES
    Flight(flight_id)
177         ON DELETE RESTRICT ON UPDATE CASCADE,
178     -- 2) 参照完整性 + 4) 约束命名 (外键关联 Passenger 表)
179     CONSTRAINT fk_ticket_passenger FOREIGN KEY (passenger_id)
    REFERENCES Passenger(passenger_id)
180         ON DELETE RESTRICT ON UPDATE CASCADE,
181      -- 3) 用户定义完整性 + 4) 约束命名 (检查价格必须为正)
182     CONSTRAINT chk_ticket_price CHECK (price >= 0) -- 允许价格为 0 (例
    如里程兑换?)
183 ) COMMENT='机票信息表';
184
185 -- ============================================================
```

```sql
-- 表: FlightStatus (航班状态)
-- ============================================================
CREATE TABLE FlightStatus (
    flight_id INT NOT NULL COMMENT '航班ID (主键/外键)',
    status VARCHAR(50) NOT NULL COMMENT '状态',
    update_time DATETIME NOT NULL COMMENT '更新时间',
    delay_reason TEXT COMMENT '延误原因',
    cancellation_reason TEXT COMMENT '取消原因',
    -- 1) 实体完整性 (命名主键约束) - 注意: 这里 flight_id 是主键, 也同时是外键
    CONSTRAINT pk_flightstatus PRIMARY KEY (flight_id),
    -- 2) 参照完整性 + 4) 约束命名 (外键关联 Flight 表)
    CONSTRAINT fk_flightstatus_flight FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    -- 3) 用户定义完整性 + 4) 约束命名 (检查状态值)
    CONSTRAINT chk_flightstatus_status CHECK (status IN ('Scheduled', 'On Time', 'Delayed', 'Cancelled', 'Boarding', 'Departed', 'Arrived'))
) COMMENT='航班状态表';


-- ============================================================
-- 添加索引以提高性能并强制约束
-- ============================================================

-- 关于聚集索引的说明 (MySQL/InnoDB):
-- 在 InnoDB 中, 表的主键自动作为聚集索引。
-- 表数据根据主键的顺序进行物理存储。
-- 因此, 在定义了主键的情况下, 不需要 (也不可能) 使用单独的 'CREATE CLUSTERED INDEX' 命令。
-- 所有上述表都有主键, 这些主键充当聚集索引, 组织了每个表的物理数据存储。

-- 例如添加唯一索引以确保航空公司名称唯一
-- 这强制约束了两个航空公司不能同名。
CREATE UNIQUE INDEX idx_unique_airline_name ON Airline (name);

-- 在外键和常用查询列上添加索引 (非唯一/辅助索引)
-- 这些索引可以提高查找和连接的查询性能。
ALTER TABLE Route ADD INDEX idx_route_airline (airline_id);
ALTER TABLE Flight ADD INDEX idx_flight_airline (airline_id);
ALTER TABLE Flight ADD INDEX idx_flight_route (route_id);
ALTER TABLE Flight ADD INDEX idx_flight_departure_time (departure_time);
ALTER TABLE Booking ADD INDEX idx_booking_passenger (passenger_id);
```

```
225  ALTER TABLE Booking ADD INDEX idx_booking_flight (flight_id);
226  ALTER TABLE Payment ADD INDEX idx_payment_booking (booking_id);
227  ALTER TABLE Ticket ADD INDEX idx_ticket_flight (flight_id);
228  ALTER TABLE Ticket ADD INDEX idx_ticket_passenger (passenger_id);
229  ALTER TABLE Passenger ADD INDEX idx_passenger_phone (phone_number);
230  -- AND: 主键通常会自动被索引。在 CREATE TABLE 中定义的唯一键
231  -- (如 id_card_number, email, ticket_number) 也会自动被索引。
232
233  -- ============================================================
234  -- END!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
235  -- ============================================================
```

而后进行测试:

4.1 实体完整性测试

尝试插入重复的主键值 (airline_id=1 已存在) ----->*失败*

```
1  INSERT INTO Airline (airline_id, name) VALUES (1, 'Duplicate Air');
```

```
mysql> USE AirlineDB
Database changed
mysql> INSERT INTO Airline (airline_id, name) VALUES (1, 'Duplicate Air');
ERROR 1062 (23000): Duplicate entry '1' for key 'airline.PRIMARY'
```

尝试插入 NULL 作为主键值 (如果列定义允许 NULL 会违反 PK 约束，但这里是 NOT NULL) ----->*失败*,not null 报错

```
1  INSERT INTO Airline (airline_id, name) VALUES (NULL, 'Null ID Air');
```

```
mysql> INSERT INTO Airline (airline_id, name) VALUES (NULL, 'Null ID Air');
ERROR 1048 (23000): Column 'airline_id' cannot be null
```

4.2 参照完整性测试

尝试插入 Flight 记录，其 airline_id 在 Airline 表中不存在----->*失败*

```sql
INSERT INTO Flight (flight_id, airline_id, route_id, departure_time,
arrival_time, departure_location, destination_location, total_seats,
aircraft_model) VALUES
(9001, 99, 201, '2025-09-01 10:00:00', '2025-09-01 12:00:00',
'Shanghai', 'Beijing', 150, 'A320');
```

```
mysql> INSERT INTO Flight (flight_id, airline_id, route_id, departure_time, arrival_time, departure_location, destination_location, total_seats, aircraft_model) VALUES
    -> (9001, 99, 201, '2025-09-01 10:00:00', '2025-09-01 12:00:00', 'Shanghai', 'Beijing', 150, 'A320');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`airlinedb`.`flight`, CONSTRAINT `flight_ibfk_1` FOREIGN KEY (`airline_id`) REFERENCES `airline` (`airline_id`) ON DELETE RESTRICT ON UPDATE CASCADE)
```

尝试删除被 Flight 表引用的 Airline 记录----->*失败*

```sql
DELETE FROM Airline WHERE airline_id = 1;
```

```
mysql> DELETE FROM Airline WHERE airline_id = 1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`airlinedb`.`flight`, CONSTRAINT `flight_ibfk_1` FOREIGN KEY (`airline_id`) REFERENCES `airline` (`airline_id`) ON DELETE RESTRICT ON UPDATE CASCADE)
```

4.3 用户定义完整性测试

尝试插入价格为负数的 Booking 记录 (违反 chk_booking_price)----->*失败*

```sql
INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type,
booking_date, price, payment_status) VALUES
(4999, 1001, 3001, 'Economy', '2025-08-01 10:00:00', -100.00,
'Pending');
```

```
mysql> INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type, booking_date, price, payment_status) VALUES
    -> (4999, 1001, 3001, 'Economy', '2025-08-01 10:00:00', -100.00, 'Pending');
ERROR 3819 (HY000): Check constraint 'chk_booking_price' is violated.
```

尝试插入支付状态无效的 Booking 记录 (违反 chk_booking_status)----->*失败*

```sql
INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type,
booking_date, price, payment_status) VALUES
(4998, 1002, 3002, 'Business', '2025-08-02 11:00:00', 1500.00,
'Processing'); -- 'Processing' 不在允许列表
```

```
mysql> INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type, booking_date, price, payment_status) VALUES
    -> (4998, 1002, 3002, 'Business', '2025-08-02 11:00:00', 1500.00, 'Processing'); -- 'Processing' 不在允许列表
ERROR 3819 (HY000): Check constraint 'chk_booking_status' is violated.
```

尝试插入到达时间早于出发时间的 Flight 记录 (违反 chk_flight_times)----->*失败*

```sql
INSERT INTO Flight (flight_id, airline_id, route_id, departure_time,
arrival_time, departure_location, destination_location, total_seats,
aircraft_model) VALUES
(9002, 1, 201, '2025-09-02 14:00:00', '2025-09-02 13:00:00',
'Shanghai', 'Beijing', 150, 'A320');
```

```
mysql> INSERT INTO Flight (flight_id, airline_id, route_id, departure_time, arrival_time, departure_location, destination_location, total_seats, aircraft_model) VALUES
    -> (9002, 1, 201, '2025-09-02 14:00:00', '2025-09-02 13:00:00', 'Shanghai', 'Beijing', 150, 'A320');
ERROR 3819 (HY000): Check constraint 'chk_flight_times' is violated.
```

4.4 用户定义完整性测试

尝试插入名称重复的 Airline 记录 (违反 uq_airline_name)----->*失败*

```sql
INSERT INTO Airline (airline_id, name) VALUES (3, 'China Eastern
Airlines');
```

```
mysql> INSERT INTO Airline (airline_id, name) VALUES (4, 'China Eastern Airlines');
ERROR 1062 (23000): Duplicate entry 'China Eastern Airlines' for key 'airline.uq_airline_name'
```

尝试插入身份证号重复的 Passenger 记录 (违反 uq_passenger_idcard)

```sql
INSERT INTO Passenger (passenger_id, name, id_card_number,
phone_number) VALUES
(1004, 'Test Dupe', '310101199001011234', '13300133000');
```

```
mysql> INSERT INTO Passenger (passenger_id, name, id_card_number, phone_number) VALUES
    -> (1004, 'Test Dupe', '310101199001011234', '13300133000');
ERROR 1062 (23000): Duplicate entry '310101199001011234' for key 'passenger.uq_passenger_idcard'
```