北京交通大學

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Lab3

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 ： 2025-04-01

```
22281188-江家玮-数据库Lab3/
├── 22281188-江家玮-数据库Lab3.md        # Markdown格式实验报告
├── 22281188-江家玮-数据库Lab3.pdf       # PDF格式实验报告
├── DBLab3_Data_Insert.sql              # 插入脚本
├── DBLab3_22281188.sql                 # 建表脚本
├── DBLab3_Work5_22281188.sql           # 加入索引后脚本
```

SQL&表、索引、查询&关系模型设计完善

作业一 数据库建表

1.1 航空公司航班查询业务模型

我在实验二的`实体型名1（属性1，…，属性n）`：

1. **航空公司（航空公司ID，名称，总部位置，联系方式，航线数量）**
2. **航班（航班ID，航空公司ID，航线，起飞时间，抵达时间，起飞地点，目的地，座位总数，已预订座位数，机型）**
3. **航线（航线ID，航司ID，起点，终点，飞行时长）**
4. **乘客（乘客ID，姓名，身份证号，联系电话，电子邮件，常旅客编号）**
5. **预订（预订ID，乘客ID，航班ID，座位类型，预订日期，票价，支付状态）**
6. **支付（支付ID，预订ID，支付方式，支付金额，支付时间，支付状态）**
7. **机票（机票ID，航班ID，票号，座位号，乘客ID，票价）**
8. **航班状态（航班ID，状态，更新时间，延误原因，取消原因）**
9. **机场（机场ID，机场名称，机场位置，航班数量）**

1.1.1 航空公司（Airline）

项目	值
表名	航空公司
数据库用户	root
主键	航空公司ID
外键	(无)
排序字段	航空公司ID
索引字段	航空公司ID, 名称

字段名称	数据类型	允许为空	唯一	默认值	约束条件
航空公司ID	INT	N	Y		主键
名称	VARCHAR(100)	N	N		
总部位置	VARCHAR(255)	Y	N		
联系方式	VARCHAR(50)	Y	N		
航线数量	INT	Y	N	0	

1.1.2 航班 (Flight)

项目	值
表名	航班
数据库用户	root
主键	航班ID
外键	航空公司ID (-> 航空公司.航空公司ID), 航线ID (-> 航线.航线ID)
排序字段	航班ID, 起飞时间
索引字段	航班ID, 航空公司ID, 航线ID, 起飞地点, 目的地, 起飞时间

字段名称	数据类型	允许为空	唯一	默认值	约束条件
航班ID	INT	N	Y		主键
航空公司ID	INT	N	N		外键 (references 航空公司)
航线ID	INT	N	N		外键 (references 航线)
起飞时间	DATETIME	N	N		
抵达时间	DATETIME	N	N		
起飞地点	VARCHAR(100)	N	N		
目的地	VARCHAR(100)	N	N		
座位总数	INT	N	N		
已预订座位数	INT	Y	N	0	
机型	VARCHAR(50)	Y	N		

1.1.3 航线 (Route)

项目	值
表名	航线
数据库用户	root
主键	航线ID
外键	航司ID (-> 航空公司.航空公司ID)
排序字段	航线ID
索引字段	航线ID, 航司ID, 起点, 终点

字段名称	数据类型	允许为空	唯一	默认值	约束条件
航线ID	INT	N	Y		主键
航司ID	INT	N	N		外键 (references 航空公司)
起点	VARCHAR(100)	N	N		
终点	VARCHAR(100)	N	N		
飞行时长	VARCHAR(20)	Y	N		

1.1.4 乘客 (Passenger)

项目	值
表名	乘客
数据库用户	root
主键	乘客ID
外键	(无)
排序字段	乘客ID
索引字段	乘客ID, 身份证号, 联系电话, 电子邮件, 常旅客编号

字段名称	数据类型	允许为空	唯一	默认值	约束条件
乘客ID	INT	N	Y		主键
姓名	VARCHAR(100)	N	N		
身份证号	VARCHAR(18)	N	Y		UNIQUE
联系电话	VARCHAR(20)	N	N		
电子邮件	VARCHAR(100)	Y	Y		UNIQUE
常旅客编号	VARCHAR(50)	Y	Y		UNIQUE

注：身份证号、电子邮件、常旅客编号要求唯一，已标记为 Y 和 UNIQUE 约束。

1.1.5 预订 (Booking)

项目	值
表名	预订
数据库用户	root
主键	预订ID
外键	乘客ID (-> 乘客.乘客ID), 航班ID (-> 航班.航班ID)
排序字段	预订ID, 预订日期
索引字段	预订ID, 乘客ID, 航班ID

字段名称	数据类型	允许为空	唯一	默认值	约束条件
预订ID	INT	N	Y		主键
乘客ID	INT	N	N		外键 (references 乘客)
航班ID	INT	N	N		外键 (references 航班)
座位类型	VARCHAR(20)	Y	N		
预订日期	DATETIME	N	N		
票价	DECIMAL(10, 2)	N	N		
支付状态	VARCHAR(20)	N	N		(e.g., '待支付', '已支付')

1.1.6 支付 (Payment)

项目	值
表名	支付
数据库用户	root
主键	支付ID
外键	预订ID (-> 预订.预订ID)
排序字段	支付ID, 支付时间
索引字段	支付ID, 预订ID

字段名称	数据类型	允许为空	唯一	默认值	约束条件
支付ID	INT	N	Y		主键
预订ID	INT	N	N		外键 (references 预订)
支付方式	VARCHAR(50)	N	N		
支付金额	DECIMAL(10, 2)	N	N		
支付时间	DATETIME	N	N		
支付状态	VARCHAR(20)	N	N		(e.g., '成功', '失败')

1.1.7 机票 (Ticket)

项目	值
表名	机票
数据库用户	root
主键	机票ID
外键	航班ID (-> 航班.航班ID), 乘客ID (-> 乘客.乘客ID)
排序字段	机票ID
索引字段	机票ID, 票号, 航班ID, 乘客ID

字段名称	数据类型	允许为空	唯一	默认值	约束条件
机票ID	INT	N	Y		主键
航班ID	INT	N	N		外键 (references 航班)
票号	VARCHAR(50)	N	Y		UNIQUE
座位号	VARCHAR(10)	N	N		
乘客ID	INT	N	N		外键 (references 乘客)
票价	DECIMAL(10, 2)	N	N		

注：票号是唯一的，已标记为 Y 和 UNIQUE 约束。

1.1.8 航班状态 (FlightStatus)

项目	值
表名	航班状态
数据库用户	root
主键	航班ID
外键	航班ID (-> 航班.航班ID)
排序字段	更新时间
索引字段	航班ID

字段名称	数据类型	允许为空	唯一	默认值	约束条件
航班ID	INT	N	Y		主键, 外键 (references 航班)
状态	VARCHAR(50)	N	N		(e.g., '准点', '延误', '取消')
更新时间	DATETIME	N	N		
延误原因	VARCHAR(255)	Y	N		
取消原因	VARCHAR(255)	Y	N		

注：这里我是假设航班状态表存储每个航班的`当前`状态，因此航班ID是主键。如果需要存储状态变更历史，则需要不同的主键设计（例如，添加一个自增ID或使用 (航班ID, 更新时间) 作为`复合主键`）。

1.1.9 机场 (Airport)

项目	值
表名	机场
数据库用户	root
主键	机场ID
外键	(无)
排序字段	机场ID
索引字段	机场ID, 机场名称

字段名称	数据类型	允许为空	唯一	默认值	约束条件
机场ID	INT	N	Y		主键
机场名称	VARCHAR(100)	N	N		
机场位置	VARCHAR(255)	N	N		
航班数量	INT	Y	N	0	

注：由于机场表中的"航班数量"可能是冗余或衍生数据，实际设计中可能不直接存储，而是可以通过查询航班表动态计算。

作业二 建表语句

根据以上定义，写出各表的建表语句，并在你选的关系型数据库平台上建立各个表，请将建表语句统一写在扩展名为sql的文件中，构建一个建库脚本文本，命名要求为：`DBLab3_22281188.sql`

我的实验过程为：

2.1 建表代码

首先将这个代码，保存为文本文档（`.txt`）

```
-- ============================================================
-- DBLab3_22281188.sql
-- Database Schema Creation Script for Airline Database
-- Target Platform: Generic SQL (Tested primarily with MySQL syntax)
-- Author: [Jiawei Jiang/Beijing Jiaotong University]
-- Date: 2025-04-01
-- ============================================================

-- 可选：如果不存在，则取消注释下面语句，即可建表
-- CREATE DATABASE IF NOT EXISTS AirlineDB CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
-- USE AirlineDB;
DROP TABLE IF EXISTS FlightStatus;
DROP TABLE IF EXISTS Ticket;
DROP TABLE IF EXISTS Payment;
```

```sql
DROP TABLE IF EXISTS Booking;
DROP TABLE IF EXISTS Flight;
DROP TABLE IF EXISTS Route;
DROP TABLE IF EXISTS Passenger;
DROP TABLE IF EXISTS Airport;
DROP TABLE IF EXISTS Airline;

-- ============================================================
-- Table: Airline (航空公司)
-- ============================================================
CREATE TABLE Airline (
    airline_id INT PRIMARY KEY 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'
) COMMENT='航空公司信息表';

-- ============================================================
-- Table: Airport (机场)
-- ============================================================
CREATE TABLE Airport (
    airport_id INT PRIMARY KEY 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)'
) COMMENT='机场信息表';

-- ============================================================
-- Table: Passenger (乘客)
-- ============================================================
CREATE TABLE Passenger (
    passenger_id INT PRIMARY KEY NOT NULL COMMENT '乘客ID',
    name VARCHAR(100) NOT NULL COMMENT '姓名',
    id_card_number VARCHAR(18) NOT NULL UNIQUE COMMENT '身份证号',
    phone_number VARCHAR(20) NOT NULL COMMENT '联系电话',
    email VARCHAR(100) UNIQUE COMMENT '电子邮件',
    frequent_flyer_number VARCHAR(50) UNIQUE COMMENT '常旅客编号'
) COMMENT='乘客信息表';

-- ============================================================
-- Table: Route (航线)
-- ============================================================
CREATE TABLE Route (
    route_id INT PRIMARY KEY NOT NULL COMMENT '航线ID',
```

```sql
    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)',
    FOREIGN KEY (airline_id) REFERENCES Airline(airline_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- Example FK constraints behavior
) COMMENT='航线信息表';

-- ============================================================
-- Table: Flight (航班)
-- ============================================================
CREATE TABLE Flight (
    flight_id INT PRIMARY KEY 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 (可能冗余, 可从Route获取)',
    destination_location VARCHAR(100) NOT NULL COMMENT '目的地 (可能冗余, 可从Route获取)',
    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',
    FOREIGN KEY (airline_id) REFERENCES Airline(airline_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (route_id) REFERENCES Route(route_id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) COMMENT='航班信息表';

-- ============================================================
-- Table: Booking (预订)
-- ============================================================
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY 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 (e.g., Pending, Paid, Cancelled)',
    FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id)
```

```
        ON DELETE CASCADE ON UPDATE CASCADE, -- If passenger deleted, maybe delete booking? Adjust as needed.
    FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- Can't delete flight if bookings exist
) COMMENT='预订信息表';

-- ============================================================
-- Table: Payment (支付)
-- ============================================================
CREATE TABLE Payment (
    payment_id INT PRIMARY KEY 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)',
    payment_amount DECIMAL(10, 2) NOT NULL COMMENT '支付金额',
    payment_time DATETIME NOT NULL COMMENT '支付时间',
    payment_status VARCHAR(20) NOT NULL DEFAULT 'Success' COMMENT '支付状态 (e.g., Success, Failed)',
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- Can't delete booking if payment exists (usually)
) COMMENT='支付信息表';

-- ============================================================
-- Table: Ticket (机票)
-- ============================================================
CREATE TABLE Ticket (
    ticket_id INT PRIMARY KEY NOT NULL COMMENT '机票ID',
    flight_id INT NOT NULL COMMENT '航班ID (外键)',
    ticket_number VARCHAR(50) NOT NULL UNIQUE COMMENT '票号',
    seat_number VARCHAR(10) NOT NULL COMMENT '座位号',
    passenger_id INT NOT NULL COMMENT '乘客ID (外键)',
    price DECIMAL(10, 2) NOT NULL COMMENT '票价 (可能冗余, 可从Booking获取)',
    FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- Don't delete passenger if they have tickets? Or set NULL? Adjust as needed.
) COMMENT='机票信息表';

-- ============================================================
-- Table: FlightStatus (航班状态)
-- ============================================================
```

```sql
CREATE TABLE FlightStatus (
    flight_id INT PRIMARY KEY NOT NULL COMMENT '航班ID (主键/外键)',
    status VARCHAR(50) NOT NULL COMMENT '状态 (e.g., On Time, Delayed, Cancelled)',
    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',
    FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
        ON DELETE CASCADE ON UPDATE CASCADE -- If flight is deleted, its status is removed.
) COMMENT='航班状态表 (通常存储当前状态)';

-- ============================================================
-- Add Indexes for performance (Optional but recommended)
-- ============================================================
-- Add indexes on foreign keys and frequently queried columns
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);
ALTER TABLE Booking ADD INDEX idx_booking_flight (flight_id);
ALTER TABLE Payment ADD INDEX idx_payment_booking (booking_id);
ALTER TABLE Ticket ADD INDEX idx_ticket_flight (flight_id);
ALTER TABLE Ticket ADD INDEX idx_ticket_passenger (passenger_id);
ALTER TABLE Passenger ADD INDEX idx_passenger_phone (phone_number);
-- Note: Primary keys are usually automatically indexed. Unique keys are also indexed.

-- ============================================================
-- End of Script
-- ============================================================
```

2.2 终端输入

```
mysql -u root -p < C:\Users\37623\Desktop\DBLab3_22281188.sql
```

而后登录

```
mysql -u root -p
```

```
C:\Users\37623>mysql -u root -p < C://Users/37623/Desktop/DBLab3_22281188.sql
Enter password: ********

C:\Users\37623>mysql -u root -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
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.
```

再输入：

```
SOURCE C://Users/37623/Desktop/DBLab3_22281188.sql;
```

```
Database changed
Query OK, 0 rows affected (0.02 sec)                Query OK, 0 rows affected (0.03 sec)
                                                    Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.02 sec)

                                                    Query OK, 0 rows affected (0.03 sec)
Query OK, 0 rows affected (0.03 sec)                Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.02 sec)
                                                    Query OK, 0 rows affected (0.03 sec)
Query OK, 0 rows affected (0.02 sec)                Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.02 sec)
                                                    Query OK, 0 rows affected (0.02 sec)
Query OK, 0 rows affected (0.02 sec)                Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.04 sec)                Query OK, 0 rows affected (0.03 sec)
                                                    Records: 0  Duplicates: 0  Warnings: 0
Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected, 3 warnings (0.03 sec)    Query OK, 0 rows affected (0.02 sec)
                                                    Records: 0  Duplicates: 0  Warnings: 0
Query OK, 0 rows affected, 1 warning (0.02 sec)

Query OK, 0 rows affected, 4 warnings (0.04 sec)    Query OK, 0 rows affected (0.03 sec)
                                                    Records: 0  Duplicates: 0  Warnings: 0
Query OK, 0 rows affected, 4 warnings (0.03 sec)

                                                    Query OK, 0 rows affected (0.03 sec)
Query OK, 0 rows affected, 8 warnings (0.04 sec)    Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected, 6 warnings (0.03 sec)
                                                    Query OK, 0 rows affected (0.02 sec)
Query OK, 0 rows affected, 7 warnings (0.03 sec)    Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected, 3 warnings (0.04 sec)
                                                    Query OK, 0 rows affected (0.02 sec)
Query OK, 0 rows affected, 4 warnings (0.03 sec)    Records: 0  Duplicates: 0  Warnings: 0
```

2.3 检查结果

2.3.1 查看警告信息

因为脚本刚刚执行完，则可以看到最近一条指令的警告信息

```
SHOW WARNINGS;
```

```
mysql> SHOW WARNINGS;
Empty set (0.00 sec)
```

2.3.2 确认数据库已被选择

```
SELECT DATABASE();
```

会显示 `AirlineDB`，如下图：

2.3.3 查看已创建的表

```
SHOW TABLES;
```

可看到脚本中定义的所有表名列表（`Airline`，`Airport`，`Booking`，`Flight`，`FlightStatus`，`Passenger`，`Payment`，`Route`，`Ticket` 共 9 个表）。

2.3.4 检查具体表的结构

我随机挑选几个重要的表，检查它们的列定义、数据类型、主键、外键、是否允许 NULL .....使用 `DESCRIBE` 或 `DESC` 命令：

```
DESCRIBE Airline;
DESC Flight;
DESC Booking;
```

```
mysql> DESCRIBE Airline;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| airline_id   | int          | NO   | PRI | NULL    |       |
| name         | varchar(100) | NO   |     | NULL    |       |
| hq_location  | varchar(255) | YES  |     | NULL    |       |
| contact_info | varchar(50)  | YES  |     | NULL    |       |
| route_count  | int          | YES  |     | 0       |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)

mysql> DESC Flight;
+----------------------+--------------+------+-----+---------+-------+
| Field                | Type         | Null | Key | Default | Extra |
+----------------------+--------------+------+-----+---------+-------+
| flight_id            | int          | NO   | PRI | NULL    |       |
| airline_id           | int          | NO   | MUL | NULL    |       |
| route_id             | int          | NO   | MUL | NULL    |       |
| departure_time       | datetime     | NO   | MUL | NULL    |       |
| arrival_time         | datetime     | NO   |     | NULL    |       |
| departure_location   | varchar(100) | NO   |     | NULL    |       |
| destination_location | varchar(100) | NO   |     | NULL    |       |
| total_seats          | int          | NO   |     | NULL    |       |
| booked_seats         | int          | YES  |     | 0       |       |
| aircraft_model       | varchar(50)  | YES  |     | NULL    |       |
+----------------------+--------------+------+-----+---------+-------+
10 rows in set (0.00 sec)

mysql> DESC Booking;
+----------------+---------------+------+-----+---------+-------+
| Field          | Type          | Null | Key | Default | Extra |
+----------------+---------------+------+-----+---------+-------+
| booking_id     | int           | NO   | PRI | NULL    |       |
| passenger_id   | int           | NO   | MUL | NULL    |       |
| flight_id      | int           | NO   | MUL | NULL    |       |
| seat_type      | varchar(20)   | YES  |     | NULL    |       |
| booking_date   | datetime      | NO   |     | NULL    |       |
| price          | decimal(10,2) | NO   |     | NULL    |       |
| payment_status | varchar(20)   | NO   |     | Pending |       |
+----------------+---------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

作业三 掌握数据插入方法

掌握选用的关系型数据库的控制台插入数据的不同方法（执行数据批量插入脚本、窗口界面表格式手工录入、命令行交互式录入），实际填入测试数据，以验证你所设计的数据模型的合理性和完整性，注意验证三种完整性约束。

3.1 执行批量数据插入脚本（`.sql`文件）

可以很高效的插入大量数据，我创建了一个`DBLab3_Data_Insert.sql`|| *注：中文会报错！因此尽量写英文*

```sql
-- DBLab3_Data_Insert.sql

SET NAMES 'utf8mb4';
USE AirlineDB;

-- Clear existing data (optional but recommended before running inserts)
DELETE FROM FlightStatus;
DELETE FROM Ticket;
```

```sql
DELETE FROM Payment;
DELETE FROM Booking;
DELETE FROM Flight;
DELETE FROM Route;
DELETE FROM Passenger;
DELETE FROM Airport;
DELETE FROM Airline;


-- Insert Airlines (Parent Table)
INSERT INTO Airline (airline_id, name, hq_location, contact_info, route_count) VALUES
(1, 'China Eastern Airlines', 'Shanghai', '95530', 0), -- Replaced '东方航空', '上海'
(2, 'China Southern Airlines', 'Guangzhou', '95539', 0); -- Replaced '南方航空', '广州'

-- Insert Airports
INSERT INTO Airport (airport_id, airport_name, location, flight_count) VALUES
(101, 'Shanghai Hongqiao International Airport', 'Shanghai, China', 0), -- Replaced Chinese names/locations
(102, 'Guangzhou Baiyun International Airport', 'Guangzhou, China', 0),
(103, 'Beijing Capital International Airport', 'Beijing, China', 0);

-- Insert Passengers
INSERT INTO Passenger (passenger_id, name, id_card_number, phone_number, email, frequent_flyer_number) VALUES
(1001, 'Jiang Jiawei', '310101199001011234', '13800138000', 'byjiaweijiang@gmail.com', 'MU123456'), -- Used Pinyin, fixed comma typo
(1002, 'Xiao Jiang', '440101199202022345', '13900139000', '22281188@bjtu.edu.cn', 'CZ654321'), -- Used Pinyin alias
(1003, 'Da Jiang', '110101198803033456', '13700137000', 'jiangpig0130@gmail.com', NULL); -- Used Pinyin alias

-- Insert Routes (Child of Airline)
INSERT INTO Route (route_id, airline_id, origin, destination, duration) VALUES
(201, 1, 'Shanghai Hongqiao International Airport', 'Beijing Capital International Airport', '2h 10m'), -- CE Shanghai->Beijing
(202, 2, 'Guangzhou Baiyun International Airport', 'Shanghai Hongqiao International Airport', '2h 00m'); -- CS Guangzhou->Shanghai

```

```sql
-- Insert Flights (Child of Airline and Route)
-- Make sure departure/arrival locations match route, and IDs exist
INSERT INTO Flight (flight_id, airline_id, route_id, departure_time, arrival_time, departure_location, destination_location, total_seats, aircraft_model) VALUES
(3001, 1, 201, '2025-07-15 08:00:00', '2025-07-15 10:10:00', 'Shanghai Hongqiao International Airport', 'Beijing Capital International Airport', 180, '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, 'B737');

-- Insert Bookings (Child of Passenger and Flight)
INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type, booking_date, price, payment_status) VALUES
(4001, 1001, 3001, 'Economy', '2025-06-10 14:30:00', 750.00, 'Paid'), -- Jiang Jiawei booked flight 3001
(4002, 1002, 3002, 'Business', '2025-06-11 10:00:00', 1800.00, 'Paid'); -- Xiao Jiang booked flight 3002
-- (Add a booking with 'Pending' status to test defaults/later updates)
INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type, booking_date, price) VALUES
(4003, 1003, 3001, 'Economy', '2025-06-12 11:00:00', 780.00); -- Da Jiang booked flight 3001, status defaults to 'Pending'

-- Insert Payments (Child of Booking)
-- Ensure booking_id exists and matches booking details
INSERT INTO Payment (payment_id, booking_id, payment_method, payment_amount, payment_time, payment_status) VALUES
(5001, 4001, 'Alipay', 750.00, '2025-06-10 14:35:00', 'Success'), -- Corresponds to Jiang Jiawei booking
(5002, 4002, 'Credit Card', 1800.00, '2025-06-11 10:05:00', 'Success'); -- Corresponds to Xiao Jiang booking

-- Insert Tickets (Child of Flight and Passenger)
-- Ensure flight_id, passenger_id exist and match booking/payment details
INSERT INTO Ticket (ticket_id, flight_id, ticket_number, seat_number, passenger_id, price) VALUES
(6001, 3001, 'TKT-MU-12345', '15A', 1001, 750.00), -- Jiang Jiawei ticket
(6002, 3002, 'TKT-CZ-67890', '03B', 1002, 1800.00); -- Xiao Jiang ticket

```

```sql
-- Insert Flight Status (Child of Flight)
-- Ensure flight_id exists
INSERT INTO FlightStatus (flight_id, status, update_time,
delay_reason, cancellation_reason) VALUES
(3001, 'On Time', '2025-07-15 07:00:00', NULL, NULL),
(3002, 'On Time', '2025-07-15 08:30:00', NULL, NULL);
```

而后在终端输入：

```
mysql -u root -p AirlineDB <
C://Users/37623/Desktop/DBLab3_Data_Insert.sql
```

```
C:\Users\37623>mysql --default-character-set=utf8mb4 -u root -p AirlineDB < C:\Users\37623\Desktop\DBLab3_Data_Insert.sql
Enter password: ********

C:\Users\37623>mysql -u root -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 24
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.
```

而后检查结果,如下：

```sql
USE AirlineDB;      -- 切换表格

SELECT * FROM Airline;
SELECT * FROM Passenger WHERE passenger_id = 1001;
SELECT * FROM Airport;
SELECT * FROM Booking;
SELECT * FROM FlightStatus;
```

```
mysql> USE AirlineDB;
Database changed
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 |
+------------+-----------------------+-------------+--------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Passenger WHERE passenger_id = 1001;
+--------------+--------------+--------------------+--------------+-------------------------+----------------------+
| passenger_id | name         | id_card_number     | phone_number | email                   | frequent_flyer_number |
+--------------+--------------+--------------------+--------------+-------------------------+----------------------+
|         1001 | Jiang Jiawei | 310101199001011234 | 13800138000  | byjiaweijiang@gmail.com | MU123456              |
+--------------+--------------+--------------------+--------------+-------------------------+----------------------+
1 row in set (0.00 sec)

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)

mysql> SELECT * FROM Booking;
+------------+--------------+-----------+-----------+---------------------+---------+----------------+
| booking_id | passenger_id | flight_id | seat_type | booking_date        | price   | payment_status |
+------------+--------------+-----------+-----------+---------------------+---------+----------------+
|       4001 |         1001 |      3001 | Economy   | 2025-06-10 14:30:00 |  750.00 | Paid           |
|       4002 |         1002 |      3002 | Business  | 2025-06-11 10:00:00 | 1800.00 | Paid           |
|       4003 |         1003 |      3001 | Economy   | 2025-06-12 11:00:00 |  780.00 | Pending        |
+------------+--------------+-----------+-----------+---------------------+---------+----------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM FlightStatus;
+-----------+---------+---------------------+--------------+---------------------+
| flight_id | status  | update_time         | delay_reason | cancellation_reason |
+-----------+---------+---------------------+--------------+---------------------+
|      3001 | On Time | 2025-07-15 07:00:00 | NULL         | NULL                |
|      3002 | On Time | 2025-07-15 08:30:00 | NULL         | NULL                |
+-----------+---------+---------------------+--------------+---------------------+
2 rows in set (0.00 sec)
```

而后检查三种完整性，通过分别注释在运行结果可看得：

```sql
-- Referential Integrity Test
-- insert a booking for a non-existent passenger (passenger_id=9999)
-- INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type, booking_date, price) VALUES
-- (4999, 9999, 3001, 'Economy', '2025-06-13 10:00:00', 700.00);

-- Unique Constraint Test
-- Attempt to insert a passenger with an existing id_card_number
-- INSERT INTO Passenger (passenger_id, name, id_card_number, phone_number, email) VALUES
-- (1004, 'Zhao Liu', '310101199001011234', '13600136000', 'zhaoliu@example.com'); -- Used Pinyin

-- FAIL (NOT NULL Constraint Test)
-- Attempt to insert a passenger with NULL name
-- INSERT INTO Passenger (passenger_id, name, id_card_number, phone_number, email) VALUES
-- (1005, NULL, '123456789012345678', '13500135000', 'nullname@example.com');

```

```
16  -- FAIL (Domain/Data Type Test)
17  -- Attempt to insert non-numeric value into price
18  -- INSERT INTO Booking (booking_id, passenger_id, flight_id,
    seat_type, booking_date, price) VALUES
19  -- (4998, 1003, 3002, 'Economy', '2025-06-14 09:00:00', 'Expensive');
```

运行第一条语句，成功运行，无错误。

运行第二条语句测试`Unique`：会发现由于`身份证ID`重合，则会报错,因为之前建表的限制是UNIQUE键

```
ERROR 1062 (23000): Duplicate entry '310101199001011234' for key 'passenger.id_card_number'
```

运行第三条语句，出现报错提示，因为把name设置为NULL，但是在之前建表时候是：`name VARCHAR(100) NOT NULL COMMENT '姓名',`

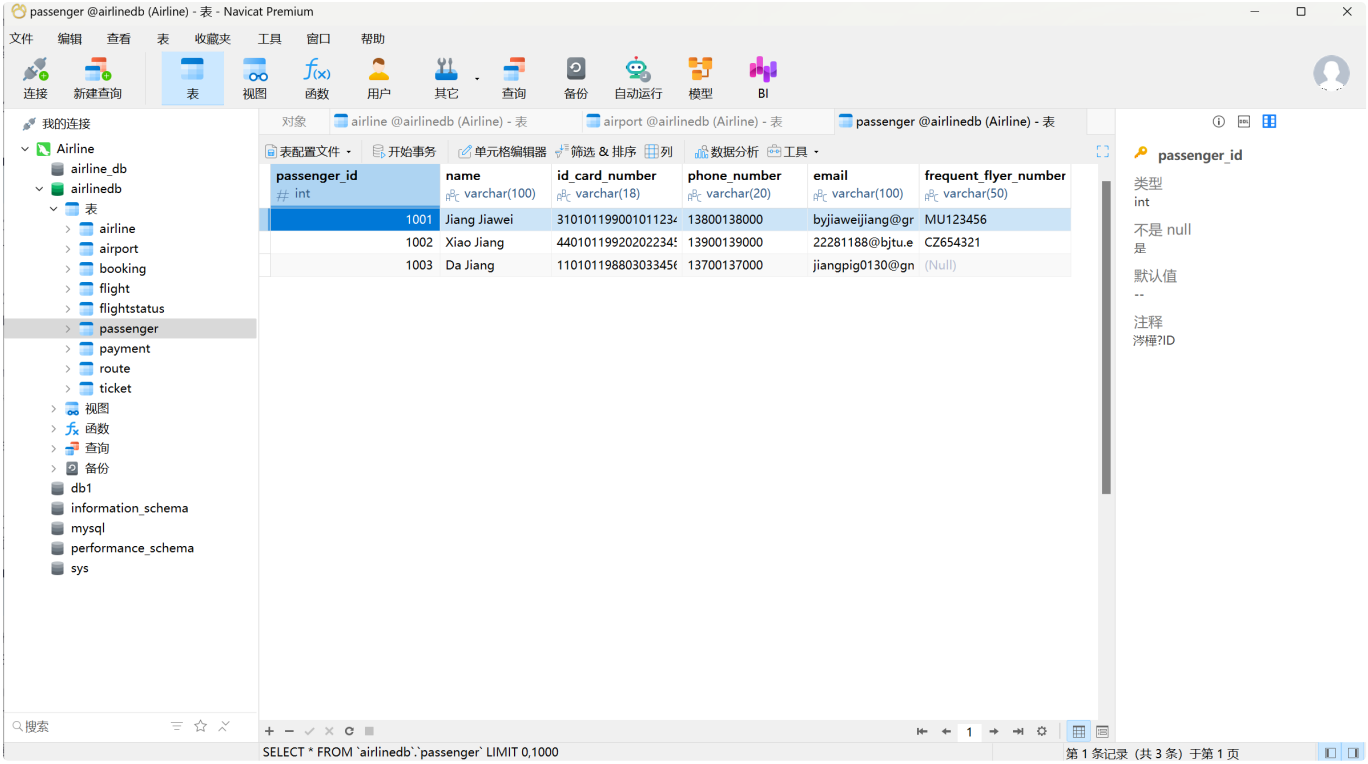
```
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

ERROR 1048 (23000): Column 'name' cannot be null
```

运行第四条语句，出现报错提示，因为价格是要数值格式，而不能输入expensive，这种string。

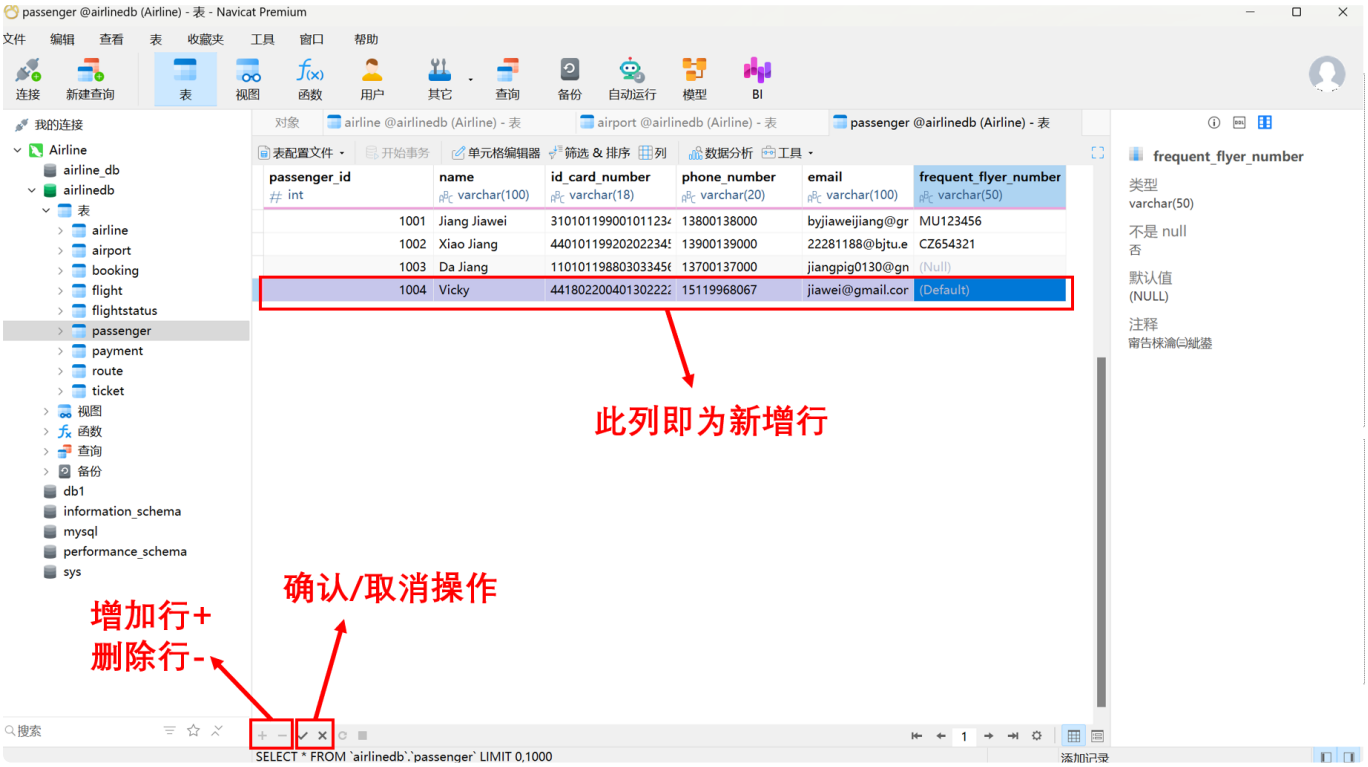
```
ERROR 1366 (HY000): Incorrect decimal value: 'Expensive' for column 'price' at row 1
```

3.2 窗口界面表格格式手动输入（GUI）

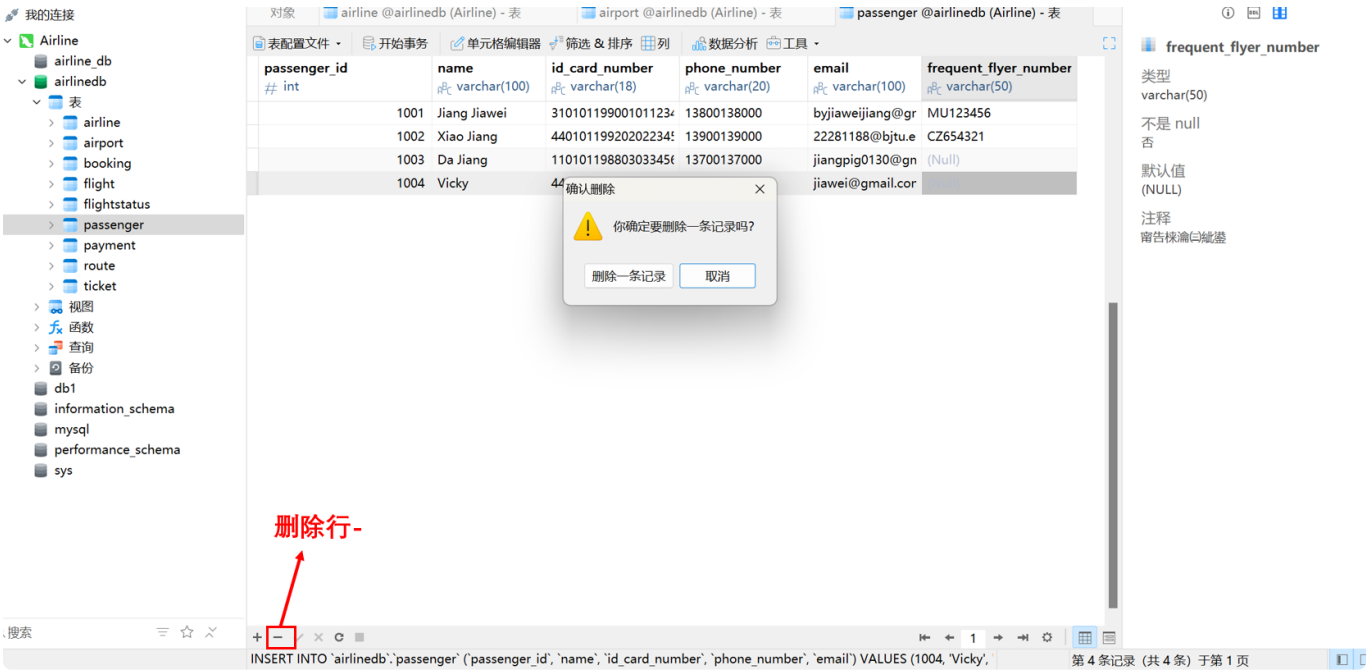
我使用`Navicat`，进行窗口界面格式的手动输入：例如我想在我的乘客的表里面新增加数据



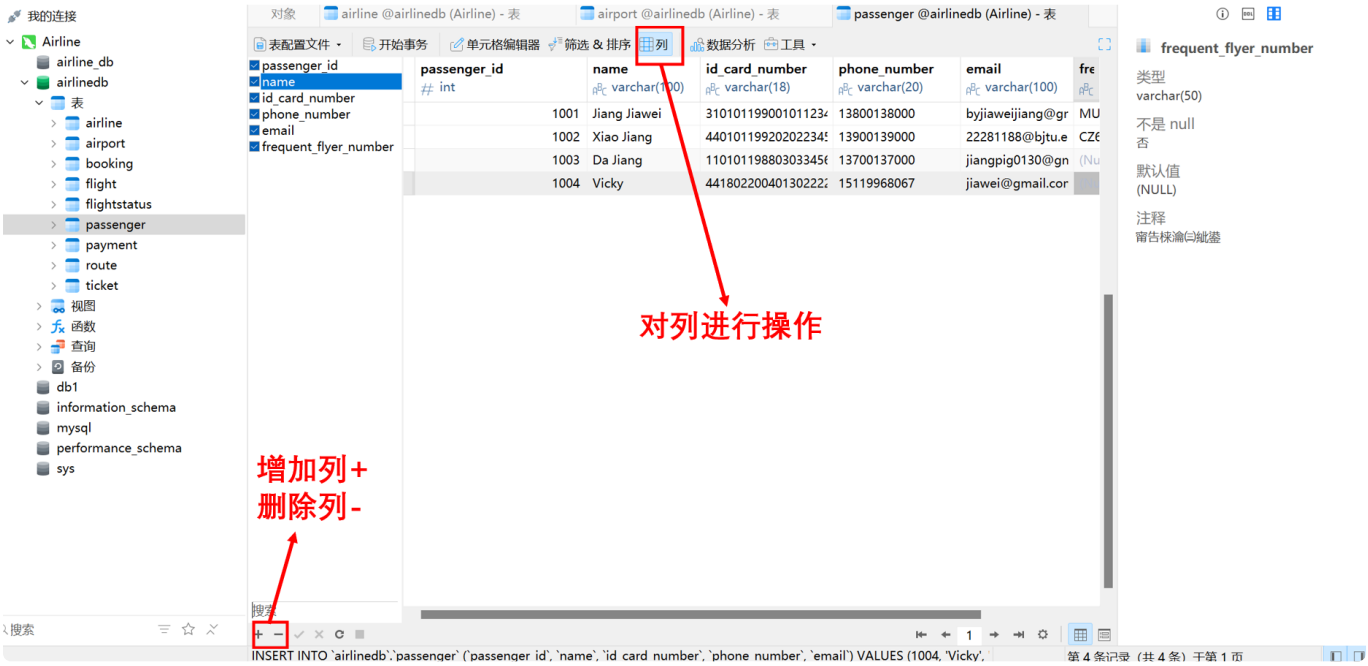
而后我新增了`Vicky`的乘客，是实现直接在窗口内即可完成输入，按左下角的`+`为输入



如果需要删除一行，则按左下角的`-`即可



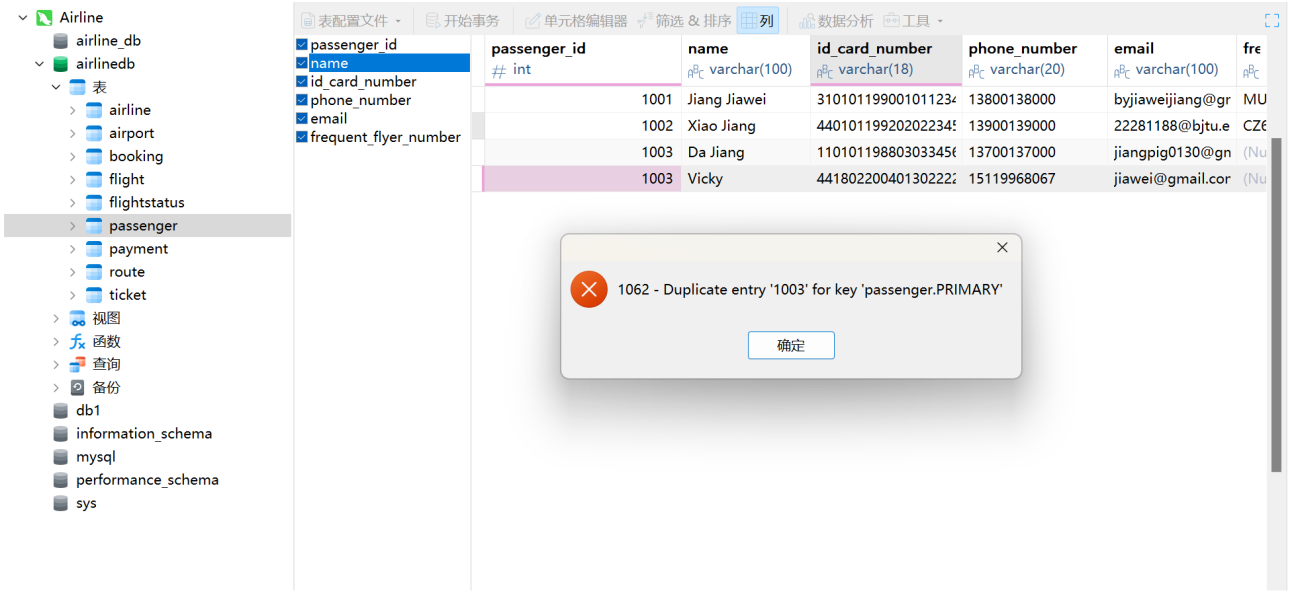
刚刚是插入了 `行`，也就是不同的元组，然后接下来也可以对 `列` 进行操作，增删属性：



接下来检验完整性：

实体完整性：

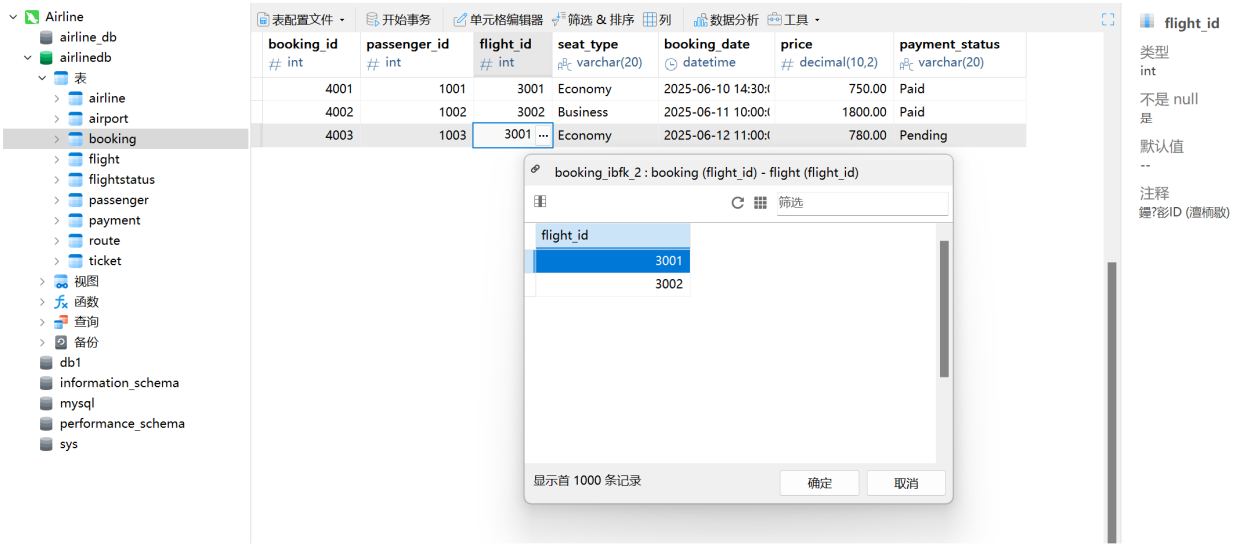
例如我把 `passenge_id` 从 `1004` 改为 `1003` 则会报错，因为此为主键，有且仅有一个！



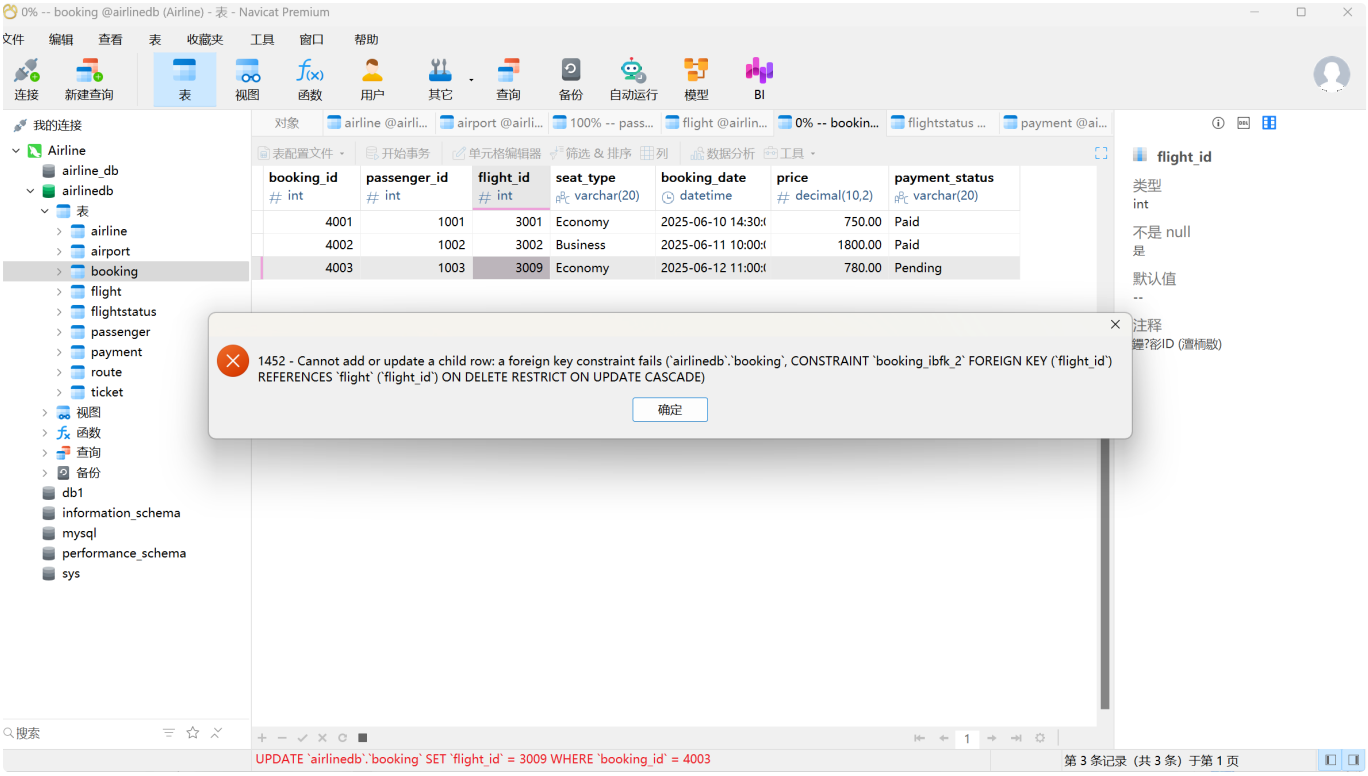
参照完整性：

例如我在`booking`的表中修改`flight_id`，由于`flight_id`是外键，参照`flight`表，因此一定是要是`filght`表里面的`flight_id`，否则就违反了参照完整性

从图中可知，`flight`表的`flight_id`只有`3001, 3002`



然后我修改为`3009`，不属于`flight`表的`flight_id`，报错！



很明显，这种可视化的窗口软件，相比较与终端命令行交互以及直接写一整个sql的代码，会更加容易上手和直观。

3.3 命令行交互式输入

登录后`INSERT INTO`直接在命令行客户端中输入语句，对于插入单行或精确测试约束非常有用。`mysql>`

选择数据库：

```
USE AirlineDB;
```

而后输入`INSERT INTO`语句：

- **a) 有效插入（基线）：**

```
INSERT INTO Passenger (passenger_id, name, id_card_number,
phone_number, email) VALUES (1004, 'zhaoliu', '110102199504044567',
'13600136000', 'zhaoliu@example.com');
-- 期待输出：OK
```

```
mysql> USE AirlineDB;
Database changed
mysql> INSERT INTO Passenger (passenger_id, name, id_card_number, phone_number, email) VALUES
 (1004, 'zhaoliu', '110102199504044567', '13600136000', 'zhaoliu@example.com');
Query OK, 1 row affected (0.01 sec)
```

- **b）测试实体完整性（重复PK）：**

```sql
INSERT INTO Passenger (passenger_id, name, id_card_number,
phone_number, email) VALUES (1004, 'qianqi', '110103199605055678',
'13500135000', 'qianqi@example.com');
-- 期待输出是Error: 主键重复输入
```

```
mysql> INSERT INTO Passenger (passenger_id, name, id_card_number, phone_number, email) VALUES
 (1004, 'qianqi', '110103199605055678', '13500135000', 'qianqi@example.com');
ERROR 1062 (23000): Duplicate entry '1004' for key 'passenger.PRIMARY'
```

- **c）测试唯一约束（重复`id_card_number`）：**

```sql
INSERT INTO Passenger (passenger_id, name, id_card_number,
phone_number, email) VALUES (1005, 'Sunba', '110102199504044567',
'13400134000', 'sunba@example.com');
-- 期待输出是Error
```

```
mysql> INSERT INTO Passenger (passenger_id, name, id_card_number, phone_number, email) VALUES
 (1005, 'Sunba', '110102199504044567', '13400134000', 'sunba@example.com');
ERROR 1062 (23000): Duplicate entry '110102199504044567' for key 'passenger.id_card_number'
```

- **d）测试域完整性（NOT NULL）：**

```sql
INSERT INTO Airport (airport_id, airport_name, location) VALUES
(105, NULL, 'China Xi an');
-- 期待输出是Error
```

```
mysql> INSERT INTO Airport (airport_id, airport_name, location) VALUES (105, NULL, 'China Xi
an');
ERROR 1048 (23000): Column 'airport_name' cannot be null
```

- **e）测试参照完整性（无效FK）：**
  SQL

```sql
INSERT INTO Flight (flight_id, airline_id, route_id,
departure_time, arrival_time, departure_location,
destination_location, total_seats) VALUES (3003, 99, 201, '2025-07-
16 10:00:00', '2025-07-16 12:10:00', 'Shanghai', 'Beijing', 150);
-- 期待输出是Error
```

```
mysql> INSERT INTO Flight (flight_id, airline_id, route_id, departure_time, arrival_time, dep
arture_location, destination_location, total_seats) VALUES (3003, 99, 201, '2025-07-16 10:00:
00', '2025-07-16 12:10:00', 'Shanghai', 'Beijing', 150);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`airlin
edb`.`flight`, CONSTRAINT `flight_ibfk_1` FOREIGN KEY (`airline_id`) REFERENCES `airline` (`a
irline_id`) ON DELETE RESTRICT ON UPDATE CASCADE)
```

- **f）测试域完整性（数据类型）：**

```sql
INSERT INTO Airline (airline_id, name, route_count) VALUES (3, 'International Airport', 'Many');
-- 期待输出是Error: many不符合数据类型
```

```
mysql> INSERT INTO Airline (airline_id, name, route_count) VALUES (3, 'International Airport'
, 'Many');
ERROR 1366 (HY000): Incorrect integer value: 'Many' for column 'route_count' at row 1
```

作业四 查询

请设计一些简单的单表查询、多表连接查询语句，查询表中的内容，并截图证明。

4.1 单表查询

这些查询每次只从一个表中检索数据。

1.查询所有乘客信息（查询所有乘客信息）

- **描述：**显示`Passenger`表中的所有记录。

```sql
SELECT * FROM Passenger;
```

```
mysql> SELECT * FROM Passenger;
+--------------+---------+--------------------+--------------+--------------------+----------------------+
| passenger_id | name    | id_card_number     | phone_number | email              | frequent_flyer_number |
+--------------+---------+--------------------+--------------+--------------------+----------------------+
|         1004 | zhaoliu | 110102199504044567 | 13600136000  | zhaoliu@example.com | NULL                 |
+--------------+---------+--------------------+--------------+--------------------+----------------------+
1 row in set (0.00 sec)
```

2.查询特定航空公司的信息（查询特定航空公司信息）

- **描述:**查找`airline_id`为 1 的中央的详细信息。

```sql
SELECT * FROM Airline WHERE airline_id = 1;
```

```
mysql> SELECT * FROM Airline WHERE airline_id = 1;
+------------+-----------------------+-------------+-------------+-------------+
| airline_id | name                  | hq_location | contact_info | route_count |
+------------+-----------------------+-------------+-------------+-------------+
|          1 | China Eastern Airlines | Shanghai    | 95530        |           0 |
+------------+-----------------------+-------------+-------------+-------------+
```

3.查询所有机场并按名称排序（查询所有机场，按名称排序）

- **描述：**显示`Airport`表中的所有机场，并按机场字母名称顺序排序。

```sql
SELECT * FROM Airport ORDER BY airport_name ASC;
```

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

4. 查询特定航班的状态（查询特定航班的状态）

- **描述:**查找`flight_id`为 3001 的航班的当前状态。

```sql
SELECT * FROM FlightStatus WHERE flight_id = 3001;
```

```
mysql> SELECT * FROM FlightStatus WHERE flight_id = 3001;
+-----------+---------+---------------------+--------------+---------------------+
| flight_id | status  | update_time         | delay_reason | cancellation_reason |
+-----------+---------+---------------------+--------------+---------------------+
|      3001 | On Time | 2025-07-15 07:00:00 | NULL         | NULL                |
+-----------+---------+---------------------+--------------+---------------------+
1 row in set (0.00 sec)
```

5. 查询票价某一值的预订记录（查询价格高于某一值的预订记录）

- **描述:**查找`Booking`表中速率（`price`）大于 1000 的所有预订记录。

```sql
SELECT booking_id, passenger_id, flight_id, price, payment_status
FROM Booking
WHERE price > 1000;
```

```
mysql> SELECT booking_id, passenger_id, flight_id, price, payment_status
    -> FROM Booking
    -> WHERE price > 1000;
+------------+--------------+-----------+---------+----------------+
| booking_id | passenger_id | flight_id | price   | payment_status |
+------------+--------------+-----------+---------+----------------+
|       4002 |         1002 |      3002 | 1800.00 | Paid           |
+------------+--------------+-----------+---------+----------------+
1 row in set (0.00 sec)
```

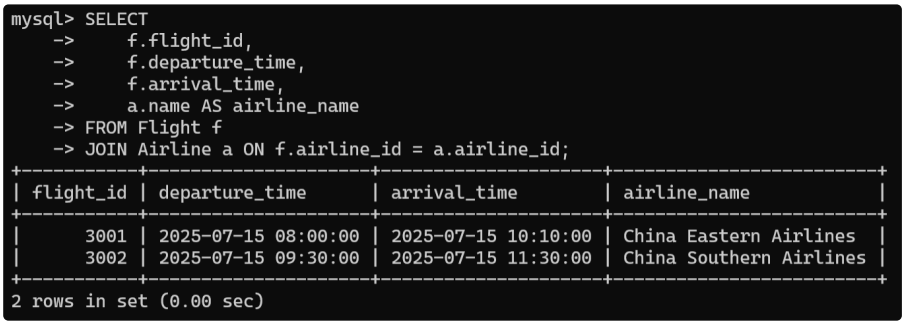
4.2 多表连接查询

这些查询使用 组合来自两个或多个相关表的数据`JOIN`。

6. 查询航班及其所属航空公司名称（查询航班及其对应航空公司名称）

- **描述:**显示每个航班的 ID、航班时间、起飞时间以及机场的名称。需要连接`Flight`和`Airline`表。

```sql
SELECT
    f.flight_id,
    f.departure_time,
    f.arrival_time,
    a.name AS airline_name
FROM Flight f
JOIN Airline a ON f.airline_id = a.airline_id;
```

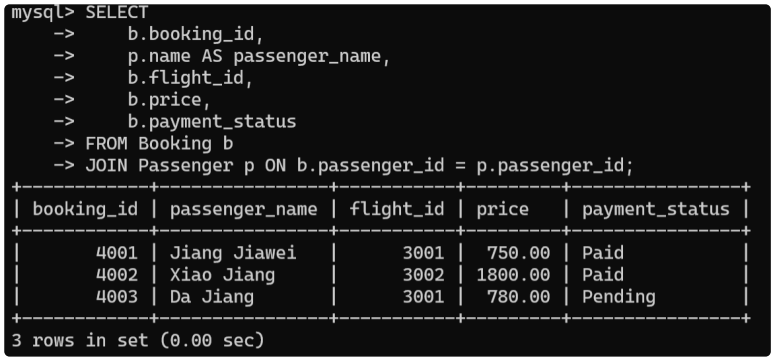


```
mysql> SELECT
    ->     f.flight_id,
    ->     f.departure_time,
    ->     f.arrival_time,
    ->     a.name AS airline_name
    -> FROM Flight f
    -> JOIN Airline a ON f.airline_id = a.airline_id;
+-----------+---------------------+---------------------+-----------------------+
| flight_id | departure_time      | arrival_time        | airline_name          |
+-----------+---------------------+---------------------+-----------------------+
|      3001 | 2025-07-15 08:00:00 | 2025-07-15 10:10:00 | China Eastern Airlines  |
|      3002 | 2025-07-15 09:30:00 | 2025-07-15 11:30:00 | China Southern Airlines |
+-----------+---------------------+---------------------+-----------------------+
2 rows in set (0.00 sec)
```

7.查询预订信息及对应的乘客姓名和航班号（用乘客姓名和航班号查询预订信息）

- **描述:**显示每个预订的 ID、乘客姓名、航班 ID 和票价。需要连接`Booking`和`Passenger`表。

```sql
SELECT
    b.booking_id,
    p.name AS passenger_name,
    b.flight_id,
    b.price,
    b.payment_status
FROM Booking b
JOIN Passenger p ON b.passenger_id = p.passenger_id;
```



```
mysql> SELECT
    ->     b.booking_id,
    ->     p.name AS passenger_name,
    ->     b.flight_id,
    ->     b.price,
    ->     b.payment_status
    -> FROM Booking b
    -> JOIN Passenger p ON b.passenger_id = p.passenger_id;
+------------+----------------+-----------+---------+----------------+
| booking_id | passenger_name | flight_id | price   | payment_status |
+------------+----------------+-----------+---------+----------------+
|       4001 | Jiang Jiawei   |      3001 |  750.00 | Paid           |
|       4002 | Xiao Jiang     |      3002 | 1800.00 | Paid           |
|       4003 | Da Jiang       |      3001 |  780.00 | Pending        |
+------------+----------------+-----------+---------+----------------+
3 rows in set (0.00 sec)
```

8.查询特定乘客的所有票务信息（查询特定乘客的所有票务信息）

- **描述:**查找乘客 'Jiang Jiawei' (`passenger_id` = 1001) 的所有占用号码和座位号。需要连接`Ticket`和`Passenger`表。

```sql
SELECT
    t.ticket_number,
    t.seat_number,
    t.flight_id,
    t.price
FROM Ticket t
JOIN Passenger p ON t.passenger_id = p.passenger_id
WHERE p.name = 'Jiang Jiawei';
-- 也可以转变为用ID: WHERE t.passenger_id = 1001;
```

```
mysql> SELECT
    ->     t.ticket_number,
    ->     t.seat_number,
    ->     t.flight_id,
    ->     t.price
    -> FROM Ticket t
    -> JOIN Passenger p ON t.passenger_id = p.passenger_id
    -> WHERE p.name = 'Jiang Jiawei';
+---------------+-------------+-----------+--------+
| ticket_number | seat_number | flight_id | price  |
+---------------+-------------+-----------+--------+
| TKT-MU-12345  | 15A         |      3001 | 750.00 |
+---------------+-------------+-----------+--------+
1 row in set (0.00 sec)
```

9.查询特定航班的详细航线信息和机场（查询特定航班的详细航线信息及航空公司）

- **描述:**显示航班 3002 的航空公司名称、航线起点、终点站和航班时长。需要连接`Flight`, `Route`, 和`Airline`表。

```sql
SELECT
    f.flight_id,
    a.name AS airline_name,
    r.origin,
    r.destination,
    r.duration
FROM Flight f
JOIN Route r ON f.route_id = r.route_id
JOIN Airline a ON f.airline_id = a.airline_id
WHERE f.flight_id = 3002;
```

```
mysql> SELECT
    ->     f.flight_id,
    ->     a.name AS airline_name,
    ->     r.origin,
    ->     r.destination,
    ->     r.duration
    -> FROM Flight f
    -> JOIN Route r ON f.route_id = r.route_id
    -> JOIN Airline a ON f.airline_id = a.airline_id
    -> WHERE f.flight_id = 3002;
+-----------+-------------------------+---------------------------------------+---------------------------------------+----------+
| flight_id | airline_name            | origin                                | destination                           | duration |
+-----------+-------------------------+---------------------------------------+---------------------------------------+----------+
|      3002 | China Southern Airlines | Guangzhou Baiyun International Airport | Shanghai Hongqiao International Airport | 2h 00m   |
+-----------+-------------------------+---------------------------------------+---------------------------------------+----------+
1 row in set (0.00 sec)
```

作业五 索引

请尝试练习在某些表上建立唯一索引和聚集索引的方法，并将建索引的语句写入建库脚本中。

5.1 唯一索引（唯一索引）

- **定义**：唯一索引是一个确保索引列/列中的所有值都是**唯一**的索引。这意味着在创建了唯一索引的列/组上，不允许有两行数据具有相同的值（but....对于允许 NULL 的列，通常允许多个 NULL 值【也就是和主键的区别，当然逐渐是特殊的唯一索引】）。
- **目的:**
  - **数据唯一性**：目的是强制数据的唯一性，防止插入重复记录（例如，确保没有两个用户的电子邮件地址相同，或者没有两个入侵的身份证号相同）。
  - **查询性能**：和普通索引一样，唯一索引也能*加快数据检索速度*，特别是当根据索引列进行精确查找时。
- **特点**：
  - 一个表可以有**多个**唯一索引。
  - 主键（Primary Key）约束上就是一个**特殊的唯一索引**（并且要求列值不能为 NULL）。
  - 例如我在`Passenger`表中为`id_card_number`,,设置的`email`约束，以及在表中为设置的约束，实际上都创建了唯一的索引。`frequent_flyer_number UNIQUE` `Ticket ticket_number UNIQUE`
  - 我们后面用`CREATE UNIQUE INDEX idx_unique_airline_name ON Airline (name);`创建的也是唯一一个索引。

5.2 聚集索引（Clustered Index）

- **定义**：聚集索引规定了表中数据行的**物理存储顺序**。表中数据行会聚集按照索引键的顺序进行物理排序和存储。（其实也就是类似于字典的索引，通过字母啥的快速定位一个大类，而后在精确查找）
- **目的:**
  - **查询性能**：对于基于聚集索引键的范围查询（例如，查找某个日期范围内的所有订单）和排序操作，性能通常非常高，因为相关数据物理存储在一起。点查（精确查找）也非常快。
  - **组织数据**：定义了表的基础物理结构。
- **特点**：
  - 一个表**只能有一个**聚集的索引，因为数据行的物理顺序存储只能有一个。
  - 在 MySQL中：
    - **主键（Primary Key）默认是聚集索引**。当定义了主键时，InnoDB就会使用主键来组织表的物理存储。
    - 如果你没有定义主键，InnoDB 会查找第一个`UNIQUE NOT NULL`索引作为聚集索引。
    - 如果两者都没有，InnoDB会内部生成一个聚集索引（基于行ID）。
    - 因此，在我的脚本中，**每个表的主键（`airline_id`、`airport_id`、`passenger_id`等）已经自动充当了该表的聚集索引**。我也就不需要也不能再单独创建它们聚集索引。
  - 聚集索引的维护（插入、删除、更新聚集键值）可能会比非聚集索引有更高的开销，因为它可能需要移动物理数据来保持顺序。

特性	唯一索引 (唯一索引)	聚集索引（Clustered Index）
主要目的	保证数据唯一性，加速查找	定义数据物理存储顺序，加速范围查找和排序
数量/表	可以有多个	**只有一个**
区别	通常是独立于数据的索引结构，包含键值和指针	**决定数据的物理存储顺序**，索引键和数据行在一起
与主键关系	主键是一种特殊的唯一索引（且 NOT NULL）	在 MySQL/InnoDB 中，**主键是默认的聚集索引**
关注点	价值是唯一性 (逻辑)	数据的物理排序 (物理)

例如我在代码中的`./DBLab3_Work5_22281188.sql`

```
-- ============================================================
-- 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（当然也不可能）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。

-- such as: 添加唯一索引以确保航空公司名称唯一
-- 那么这就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);
ALTER TABLE Booking ADD INDEX idx_booking_flight (flight_id);
ALTER TABLE Payment ADD INDEX idx_payment_booking (booking_id);
ALTER TABLE Ticket ADD INDEX idx_ticket_flight (flight_id);
```

```sql
ALTER TABLE Ticket ADD INDEX idx_ticket_passenger (passenger_id);
ALTER TABLE Passenger ADD INDEX idx_passenger_phone (phone_number);
-- AND: 主键通常会自动被索引。在 CREATE TABLE 中定义的唯一键
-- (如 id_card_number, email, ticket_number) 也会自动被索引。
```

作业六 大数据仿真方案

若某个表中涉及百万甚至千万级以上的数据，请提出仿真这些数据的方案，并在实验报告加以叙述。

6.1 目标表与数据规模

首先先估算，而后才更好的开展，我认为以下表最有可能达到百万级以上的数据量，应该是本次仿真的主要目标：

- `Booking`**(预订表)**：核心交易表，记录每次订票行为，数据量增长最快。**目标规模：1000万 ~ 5000万条。**
- `Ticket`**(接收表):**与成功预订和支付相关，通常与接收`Booking`表规模接近。**目标规模：1000万 ~ 5000万条。**
- `Payment`**(支付表):**记录与预订相关的支付行为。**目标规模：1000万 ~ 5000万条**(可能略多于成功预订数，包含失败尝试)。
- `Passenger`**(上升表):**累计用户数，可能达到百万级。**目标规模：100万 ~ 500万条。**
- `Flight` **（航班表）**：如果包含历史航班数据，数据量会很大。**目标规模：100万~500万条**（假设模拟几年的数据）。

其他表如`Airline`、`Airport`、`Route`、`FlightStatus`数据量相对较小，可按实际情况生成远程数据（百/千/万级....）。

6.2 数据特征与依赖关系定义

生成的数据必须符合表结构定义中的约束（主键、外键、唯一、非空、数据类型）并要求模拟真实世界的数据分布和关联：

- **主键 (PK):**
  - 必须唯一且非空。
  - 生成：对于`INT`类型主键，可利用程序控制从一个增量的起始值开始**递增**生成，保证唯一性。对于需要特定格式的ID（如票号`ticket_number`），按高级模式（如

“TKT-[AZ]{2}-[0-9]{7}”）生成唯一字符串。

- **外键 (FK):**
  - 必须引用父表中已存在的主键值。
  - 生成策略：在生成子表数据时，需要从对应的父表中**随机**或**按某种方式分配**（例如，热门航线/航班被预订更多）一些有效的父表主键值。例如，生成`Booking`记录时，`passenger_id`需要从`Passenger`表已存在的ID中一些，`flight_id`需从`Flight`表已存在的ID中存在一些。必须先生成父表数据。
- 数据分布模拟:
  - **姓名（`Passenger.name`):**使用常见的中文姓氏和姓名组合规则，或使用拼音/中文名，避免简单递增（如“上升1”、“上升2”）。可借助姓名生成库或预定义列表。
  - **身份证号/电话合法/邮箱（`Passenger`):**按格式生成唯一的或允许部分重复（邮箱）的字符串。
  - **日期时间（`Flight`，`Booking`，`Payment`):**在合理的时间范围内生成。例如，航班起降时间在一年到未来半年；预订过去日期通常早于航班起降日期；支付时间紧随预订日期。确保时间逻辑正确（起降时间）。
  - **地点/机场名称：**从`Airport`表中出现。
  - **价格（`Booking`，`Ticket`，`Payment`):**在合理范围内生成随机数（如 100-10000），可根据类型（`seat_type`）或路线（`Route`）设置不同的价格区间。确保关联表（如`Booking.price`和`Payment.payment_amount`）金额一致。
  - **状态字段（`Booking.payment_status`，`Payment.payment_status`，`FlightStatus.status`):**按默认比例生成，如 90% 的预订为“付费”，95% 的支付为“成功”，大部分航班状态为“准点”。
- **数据一致性：**保证关联数据间的逻辑状态的逻辑一致性，如一张`Ticket`记录对应一个“付费”的`Booking`记录，且乘客ID和航班ID一致。

6.3 仿真方法选择

> 我认为脚本编程 + CSV 文件 + 数据库批量加载的可实施可能性最大，相对最简单。

1. **编程语言选择：**使用Python等。Python拥有丰富的数据处理和生成库（如`Faker`用于生成模拟数据，`pandas`用于数据处理，`random`用于随机选择等）以及数据库连接库（`mysql-connector-python`等）。
2. 数据生成:
   - 编写脚本，首先根据规则生成父表数据并存入内存或临时数据库/文件，获取其主键ID列表。

- 同时，循环生成子表数据。在生成每一条子表记录时，根据外键约束，从内存中或数据库中查询/随机一些有效的父表主键ID。
- 使用`Faker`等库生成姓名、地址、日期、文本等模拟数据。
- 将生成的大量数据**写入CSV文件**，每个表对应一个或多个CSV文件。

3. 数据加载:
   - 使用目标数据库提供的
     批量加载工具
     将CSV文件中的数据高速导入数据库表中。
     - **MySQL**：使用`LOAD DATA INFILE 'path/to/your_table.csv' INTO TABLE YourTable FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n' IGNORE 1 ROWS;`（参数根据实际CSV格式调整）。这是MySQL中加载大量数据**效率最高**的方式。
   - **优点**：生成数据逻辑与数据库加载分离，灵活性高；利用数据库批量加载工具，速度远超逐条`INSERT`。

代码附录

`DBLab3_22281188.sql`

```sql
-- ============================================================
-- DBLab3_22281188.sql
-- 航空公司数据库模式创建脚本
-- 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]  -- 请确认作者信息是否需要保留或修改
-- 日期: 2025-04-01
-- ============================================================

-- 选: 如果数据库不存在则创建
-- CREATE DATABASE IF NOT EXISTS AirlineDB CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
-- USE AirlineDB;

-- 按依赖反序删除表 (如果存在)
DROP TABLE IF EXISTS FlightStatus;
DROP TABLE IF EXISTS Ticket;
DROP TABLE IF EXISTS Payment;
DROP TABLE IF EXISTS Booking;
```

```sql
DROP TABLE IF EXISTS Flight;
DROP TABLE IF EXISTS Route;
DROP TABLE IF EXISTS Passenger;
DROP TABLE IF EXISTS Airport;
DROP TABLE IF EXISTS Airline;

-- ============================================================
-- Table: Airline (航空公司)
-- ============================================================
CREATE TABLE Airline (
    airline_id INT PRIMARY KEY 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'
) COMMENT='航空公司信息表';

-- ============================================================
-- Table: Airport (机场)
-- ============================================================
CREATE TABLE Airport (
    airport_id INT PRIMARY KEY 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)'
) COMMENT='机场信息表';

-- ============================================================
-- Table: Passenger (乘客)
-- ============================================================
CREATE TABLE Passenger (
    passenger_id INT PRIMARY KEY NOT NULL COMMENT '乘客ID',
    name VARCHAR(100) NOT NULL COMMENT '姓名',
    id_card_number VARCHAR(18) NOT NULL UNIQUE COMMENT '身份证号',
    phone_number VARCHAR(20) NOT NULL COMMENT '联系电话',
    email VARCHAR(100) UNIQUE COMMENT '电子邮件',
    frequent_flyer_number VARCHAR(50) UNIQUE COMMENT '常旅客编号'
) COMMENT='乘客信息表';

-- ============================================================
-- Table: Route (航线)
-- ============================================================
CREATE TABLE Route (
    route_id INT PRIMARY KEY NOT NULL COMMENT '航线ID',
    airline_id INT NOT NULL COMMENT '航空公司ID (外键)',
```

```sql
    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)',
    FOREIGN KEY (airline_id) REFERENCES Airline(airline_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- Example FK constraints behavior
) COMMENT='航线信息表';

-- ============================================================
-- Table: Flight (航班)
-- ============================================================
CREATE TABLE Flight (
    flight_id INT PRIMARY KEY 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 (可能冗余, 可从Route获取)',
    destination_location VARCHAR(100) NOT NULL COMMENT '目的地 (可能冗余, 可从Route获取)',
    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',
    FOREIGN KEY (airline_id) REFERENCES Airline(airline_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (route_id) REFERENCES Route(route_id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) COMMENT='航班信息表';

-- ============================================================
-- Table: Booking (预订)
-- ============================================================
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY 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 (e.g., Pending, Paid, Cancelled)',
    FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id)
        ON DELETE CASCADE ON UPDATE CASCADE, -- If passenger deleted, maybe delete booking? Adjust as needed.
```

```sql
    FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- Can't delete flight if bookings exist
) COMMENT='预订信息表';

-- ============================================================
-- Table: Payment (支付)
-- ============================================================
CREATE TABLE Payment (
    payment_id INT PRIMARY KEY 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)',
    payment_amount DECIMAL(10, 2) NOT NULL COMMENT '支付金额',
    payment_time DATETIME NOT NULL COMMENT '支付时间',
    payment_status VARCHAR(20) NOT NULL DEFAULT 'Success' COMMENT '支付状态 (e.g., Success, Failed)',
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- Can't delete booking if payment exists (usually)
) COMMENT='支付信息表';

-- ============================================================
-- Table: Ticket (机票)
-- ============================================================
CREATE TABLE Ticket (
    ticket_id INT PRIMARY KEY NOT NULL COMMENT '机票ID',
    flight_id INT NOT NULL COMMENT '航班ID (外键)',
    ticket_number VARCHAR(50) NOT NULL UNIQUE COMMENT '票号',
    seat_number VARCHAR(10) NOT NULL COMMENT '座位号',
    passenger_id INT NOT NULL COMMENT '乘客ID (外键)',
    price DECIMAL(10, 2) NOT NULL COMMENT '票价 (可能冗余, 可从Booking获取)',
    FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- Don't delete passenger if they have tickets? Or set NULL? Adjust as needed.
) COMMENT='机票信息表';

-- ============================================================
-- Table: FlightStatus (航班状态)
-- ============================================================
CREATE TABLE FlightStatus (
    flight_id INT PRIMARY KEY NOT NULL COMMENT '航班ID (主键/外键)',
```

```
    status VARCHAR(50) NOT NULL COMMENT '状态 (e.g., On Time, Delayed, Cancelled)',
    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',
    FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
        ON DELETE CASCADE ON UPDATE CASCADE -- If flight is deleted, its status is removed.
) COMMENT='航班状态表 (通常存储当前状态)';


-- 外键和一些频繁访问的列
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);
ALTER TABLE Booking ADD INDEX idx_booking_flight (flight_id);
ALTER TABLE Payment ADD INDEX idx_payment_booking (booking_id);
ALTER TABLE Ticket ADD INDEX idx_ticket_flight (flight_id);
ALTER TABLE Ticket ADD INDEX idx_ticket_passenger (passenger_id);
ALTER TABLE Passenger ADD INDEX idx_passenger_phone (phone_number);

-- ============================================================
-- END!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
-- ============================================================
```

DBLab3_Data_Insert.sql

```
-- ============================================================
-- DBLab3_22281188.sql
-- 航空公司数据库模式创建脚本
-- 目标平台: 航空公司数据库示例数据插入脚本
-- 作者: [江家玮/北京交通大学]  -- 请确认作者信息是否需要保留或修改
-- 日期: 2025-04-01
-- ============================================================

-- 确保正在使用正确的数据库
SET NAMES 'utf8mb4';
USE AirlineDB;

-- 清除现有数据
DELETE FROM FlightStatus;
```

```sql
DELETE FROM Ticket;
DELETE FROM Payment;
DELETE FROM Booking;
DELETE FROM Flight;
DELETE FROM Route;
DELETE FROM Passenger;
DELETE FROM Airport;
DELETE FROM Airline;


-- 插入航空公司 (父表)
INSERT INTO Airline (airline_id, name, hq_location, contact_info, route_count) VALUES
(1, 'China Eastern Airlines', 'Shanghai', '95530', 0), -- 替换了 '东方航空', '上海'
(2, 'China Southern Airlines', 'Guangzhou', '95539', 0); -- 替换了 '南方航空', '广州'

-- 插入机场
INSERT INTO Airport (airport_id, airport_name, location, flight_count) VALUES
(101, 'Shanghai Hongqiao International Airport', 'Shanghai, China', 0), -- 替换了中文名称/位置
(102, 'Guangzhou Baiyun International Airport', 'Guangzhou, China', 0),
(103, 'Beijing Capital International Airport', 'Beijing, China', 0);

-- 插入乘客
INSERT INTO Passenger (passenger_id, name, id_card_number, phone_number, email, frequent_flyer_number) VALUES
(1001, 'Jiang Jiawei', '310101199001011234', '13800138000', 'byjiaweijiang@gmail.com', 'MU123456'), -- 使用了拼音，修正了逗号拼写错误
(1002, 'Xiao Jiang', '440101199202022345', '13900139000', '22281188@bjtu.edu.cn', 'CZ654321'), -- 使用了拼音别名
(1003, 'Da Jiang', '110101198803033456', '13700137000', 'jiangpig0130@gmail.com', NULL); -- 使用了拼音别名

-- 插入航线 (航空公司的子表)
INSERT INTO Route (route_id, airline_id, origin, destination, duration) VALUES
(201, 1, 'Shanghai Hongqiao International Airport', 'Beijing Capital International Airport', '2h 10m'), -- 东航 上海->北京
(202, 2, 'Guangzhou Baiyun International Airport', 'Shanghai Hongqiao International Airport', '2h 00m'); -- 南航 广州->上海

```

```sql
-- 插入航班 (航空公司和航线的子表)
-- 确保起飞/到达地点与航线匹配，且ID存在
INSERT INTO Flight (flight_id, airline_id, route_id, departure_time, arrival_time, departure_location, destination_location, total_seats, aircraft_model) VALUES
(3001, 1, 201, '2025-07-15 08:00:00', '2025-07-15 10:10:00', 'Shanghai Hongqiao International Airport', 'Beijing Capital International Airport', 180, '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, 'B737');

-- 插入预订 (乘客和航班的子表)
INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type, booking_date, price, payment_status) VALUES
(4001, 1001, 3001, 'Economy', '2025-06-10 14:30:00', 750.00, 'Paid'), -- 江家玮预订了航班3001
(4002, 1002, 3002, 'Business', '2025-06-11 10:00:00', 1800.00, 'Paid'); -- 小江江预订了航班3002
-- (添加一个状态为'Pending'的预订以测试默认值/后续更新)
INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type, booking_date, price) VALUES
(4003, 1003, 3001, 'Economy', '2025-06-12 11:00:00', 780.00); -- 大江江预订了航班3001, 状态默认为'Pending'

-- 插入支付 (预订的子表)
-- 确保 booking_id 存在且与预订详情匹配
INSERT INTO Payment (payment_id, booking_id, payment_method, payment_amount, payment_time, payment_status) VALUES
(5001, 4001, 'Alipay', 750.00, '2025-06-10 14:35:00', 'Success'), -- 对应江家玮的预订
(5002, 4002, 'Credit Card', 1800.00, '2025-06-11 10:05:00', 'Success'); -- 对应小江江的预订

-- 插入机票 (航班和乘客的子表)
-- 确保 flight_id, passenger_id 存在且与预订/支付详情匹配
INSERT INTO Ticket (ticket_id, flight_id, ticket_number, seat_number, passenger_id, price) VALUES
(6001, 3001, 'TKT-MU-12345', '15A', 1001, 750.00), -- 江家玮的机票
(6002, 3002, 'TKT-CZ-67890', '03B', 1002, 1800.00); -- 小江江的机票

-- 插入航班状态 (航班的子表)
-- 确保 flight_id 存在
INSERT INTO FlightStatus (flight_id, status, update_time, delay_reason, cancellation_reason) VALUES
```

```
(3001, 'On Time', '2025-07-15 07:00:00', NULL, NULL),
(3002, 'On Time', '2025-07-15 08:30:00', NULL, NULL);

-- ============================================================
-- 测试数据完整性约束

-- 参照完整性测试
-- 尝试插入一个不存在的乘客 (passenger_id=9999) 的预订
-- INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type, booking_date, price) VALUES
-- (4999, 9999, 3001, 'Economy', '2025-06-13 10:00:00', 700.00);

-- 唯一约束测试
-- 尝试插入一个具有现有身份证号的乘客
-- INSERT INTO Passenger (passenger_id, name, id_card_number, phone_number, email) VALUES
-- (1004, 'Zhao Liu', '310101199001011234', '13600136000', 'zhaoliu@example.com'); -- 使用了拼音

-- 失败 (NOT NULL 约束测试)
-- 尝试插入一个姓名为NULL的乘客
-- INSERT INTO Passenger (passenger_id, name, id_card_number, phone_number, email) VALUES
-- (1005, NULL, '123456789012345678', '13500135000', 'nullname@example.com');

-- 失败 (域/数据类型测试)
-- 尝试向价格插入非数值
-- INSERT INTO Booking (booking_id, passenger_id, flight_id, seat_type, booking_date, price) VALUES
-- (4998, 1003, 3002, 'Economy', '2025-06-14 09:00:00', 'Expensive');
```

DBLab3_Work5_22281188.sql

```
-- ============================================================
-- DBLab3_22281188.sql
-- 航空公司数据库模式创建脚本
-- 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]  -- 请确认作者信息是否需要保留或修改
-- 日期: 2025-04-01
-- ============================================================

-- 选: 如果数据库不存在则创建
```

```sql
-- CREATE DATABASE IF NOT EXISTS AirlineDB CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
-- USE AirlineDB;

-- 按依赖反序删除表 (如果存在)
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;
DROP TABLE IF EXISTS Airport;
DROP TABLE IF EXISTS Airline;

-- ============================================================
-- 表: Airline (航空公司)
-- ============================================================
CREATE TABLE Airline (
    airline_id INT PRIMARY KEY 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'
) COMMENT='航空公司信息表';

-- ============================================================
-- 表: Airport (机场)
-- ============================================================
CREATE TABLE Airport (
    airport_id INT PRIMARY KEY 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)'
) COMMENT='机场信息表';

-- ============================================================
-- 表: Passenger (乘客)
-- ============================================================
CREATE TABLE Passenger (
    passenger_id INT PRIMARY KEY NOT NULL COMMENT '乘客ID',
    name VARCHAR(100) NOT NULL COMMENT '姓名',
    id_card_number VARCHAR(18) NOT NULL UNIQUE COMMENT '身份证号',
    phone_number VARCHAR(20) NOT NULL COMMENT '联系电话',
    email VARCHAR(100) UNIQUE COMMENT '电子邮件',
```

```sql
    frequent_flyer_number VARCHAR(50) UNIQUE COMMENT '常旅客编号'
) COMMENT='乘客信息表';

-- ============================================================
-- 表: Route (航线)
-- ============================================================
CREATE TABLE Route (
    route_id INT PRIMARY KEY 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)',
    FOREIGN KEY (airline_id) REFERENCES Airline(airline_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- 外键约束行为的example
) COMMENT='航线信息表';

-- ============================================================
-- 表: Flight (航班)
-- ============================================================
CREATE TABLE Flight (
    flight_id INT PRIMARY KEY 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 (可能冗余, 可从Route获取)',
    destination_location VARCHAR(100) NOT NULL COMMENT '目的地 (可能冗余, 可从Route获取)',
    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',
    FOREIGN KEY (airline_id) REFERENCES Airline(airline_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (route_id) REFERENCES Route(route_id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) COMMENT='航班信息表';

-- ============================================================
-- 表: Booking (预订)
-- ============================================================
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY 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)',
```

```sql
    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 (e.g., Pending, Paid, Cancelled)',
    FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id)
        ON DELETE CASCADE ON UPDATE CASCADE, -- 如果乘客被删除，也许删除其预订？按需调整。
    FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- 如果存在预订，则无法删除航班
) COMMENT='预订信息表';

-- ============================================================
-- 表: Payment (支付)
-- ============================================================
CREATE TABLE Payment (
    payment_id INT PRIMARY KEY 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)',
    payment_amount DECIMAL(10, 2) NOT NULL COMMENT '支付金额',
    payment_time DATETIME NOT NULL COMMENT '支付时间',
    payment_status VARCHAR(20) NOT NULL DEFAULT 'Success' COMMENT '支付状态 (e.g., Success, Failed)',
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
        ON DELETE RESTRICT ON UPDATE CASCADE -- 如果存在支付记录，通常无法删除预订
) COMMENT='支付信息表';

-- ============================================================
-- 表: Ticket (机票)
-- ============================================================
CREATE TABLE Ticket (
    ticket_id INT PRIMARY KEY NOT NULL COMMENT '机票ID',
    flight_id INT NOT NULL COMMENT '航班ID (外键)',
    ticket_number VARCHAR(50) NOT NULL UNIQUE COMMENT '票号',
    seat_number VARCHAR(10) NOT NULL COMMENT '座位号',
    passenger_id INT NOT NULL COMMENT '乘客ID (外键)',
    price DECIMAL(10, 2) NOT NULL COMMENT '票价 (可能冗余，可从Booking获取)',
    FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id)
```

```
        ON DELETE RESTRICT ON UPDATE CASCADE -- 如果乘客有机票，是否禁止删除？或设为NULL？按需调整。
) COMMENT='机票信息表';

-- ============================================================
-- 表: FlightStatus (航班状态)
-- ============================================================
CREATE TABLE FlightStatus (
    flight_id INT PRIMARY KEY NOT NULL COMMENT '航班ID (主键/外键)',
    status VARCHAR(50) NOT NULL COMMENT '状态 (e.g., On Time, Delayed, Cancelled)',
    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',
    FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
        ON DELETE CASCADE ON UPDATE CASCADE -- 如果航班被删除，其状态记录也被移除。
) COMMENT='航班状态表 (通常存储当前状态)';


-- ============================================================
-- 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。

-- 示例: 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);
ALTER TABLE Booking ADD INDEX idx_booking_flight (flight_id);
ALTER TABLE Payment ADD INDEX idx_payment_booking (booking_id);
```

```sql
ALTER TABLE Ticket ADD INDEX idx_ticket_flight (flight_id);
ALTER TABLE Ticket ADD INDEX idx_ticket_passenger (passenger_id);
ALTER TABLE Passenger ADD INDEX idx_passenger_phone (phone_number);
-- AND: 主键通常会自动被索引。在 CREATE TABLE 中定义的唯一键
-- (如 id_card_number, email, ticket_number) 也会自动被索引。

-- ============================================================
-- END!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
-- ============================================================
```