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Lab8

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 ： 2025-05-21

目录结构：

```
Lab8_22281188/
├── Lab8_Report_22281188.md          # Markdown 格式实验报告
├── DBLab8_Base_Schema.sql           # 基础表结构脚本（基于Lab3修改，移
除了部分UNIQUE约束以避免插入冲突）
├── DBLab8_Add_Index_A.sql           # 为 Passenger.name 属性添加索引
的脚本
├── scripts/                         # Python 脚本目录
|   ├── config.py                    # 数据库连接配置和测试参数
|   ├── data_generator.py            # 数据生成辅助脚本
|   ├── inserter.py                  # 数据插入脚本（记录耗时）
|   ├── querier.py                   # 数据查询脚本（记录耗时）
|   └── main_test_runner.py          # 概念性的测试编排脚本
├── results/                         # 实验结果存放目录
|   ├── no_index_insert_log.csv      # 无索引时插入操作耗时记录
|   ├── no_index_query_log.csv       # 无索引时查询操作耗时记录
|   ├── with_index_insert_log.csv    # 有索引时插入操作耗时记录
|   ├── with_index_query_log.csv     # 有索引时查询操作耗时记录
|   └── performance_charts/          # 生成的性能图表存放目录
├── backup/                          # 数据库备份文件存放目录
|   ├── airline_db_full_backup.sql
|   └── passenger_table_backup.sql
└── logs_analysis/                   # 日志分析相关文件
└── mysql_binlog_example.txt
```

实验八：索引与压力测试、备份与日志初步

一、实验要求环境

1. **操作系统：** macOS
2. **数据库管理系统：** MySQL
3. **编程语言：** Python 3.12
4. **Python库：**
    - `mysql-connector-python`: 用于Python程序与MySQL数据库进行交互。
    - `Faker`: 用于生成大量的模拟数据，以进行压力测试。
    - `pandas`, `matplotlib`, `seaborn`

`pip` 命令安装了这些必要的Python库

```
pip install mysql-connector-python Faker pandas matplotlib seaborn
```

二、实验目的

本次实验的核心目的在于深入理解和实践数据库管理中的几个关键方面：

1. **探究索引机制：** 通过对比在有索引和无索引的情况下，数据插入和查询操作的性能表现，直观感受索引对数据库性能的影响。
2. **学习压力测试：** 掌握对数据库进行压力测试的基本方法，模拟高并发场景，评估数据库在不同负载下的稳定性和处理能力。
3. **掌握备份与恢复：** 学习使用如 `mysqldump` 等工具对数据库进行单表备份和整库备份，并实践数据恢复过程，理解其重要性。
4. **初步了解数据库日志：** 学习数据库日志（特别是MySQL的二进制日志Binlog或通用查询日志General Query Log）的概念、配置和查看方法，理解其在数据追踪、审计和故障恢复中的作用。

三、运行过程&测试

1. **创建数据库和基础表结构:**
   - 执行 `DBLab8_Base_Schema.sql` 脚本来创建实验所需的表结构。此脚本基于Lab3的结构，但**已移除了 `Passenger` 表中 `email`, `id_card_number`, `frequent_flyer_number` 列的 `UNIQUE` 约束**，以避免在大量数据插入时因 `Faker` 生成重复数据而导致实验中断。

     ```
     # 命令行
     mysql -u your_mysql_user -p AirlineDB <
     DBLab8_Base_Schema.sql
     ```

运行实验

实验主要分为两部分：压力测试与索引实验，以及备份与日志初步实验。

第一部分：压力测试与索引实验

此部分包含两个主要场景：无索引和有索引。

重要: 在切换场景或重新运行同一场景前，需要清空 `Passenger` 表的数据 (`TRUNCATE TABLE Passenger;`) 并确保索引状态正确。

场景1: 无索引压力测试

1. **准备数据库 (无索引):**
   - 连接到 `AirlineDB`。
   - 确保 `Passenger` 表的 `name` 列上**没有**名为 `idx_passenger_name` 的索引。如果存在，请删除它：

     ```
     USE AirlineDB;
     DROP INDEX IF EXISTS idx_passenger_name ON Passenger;
     ```

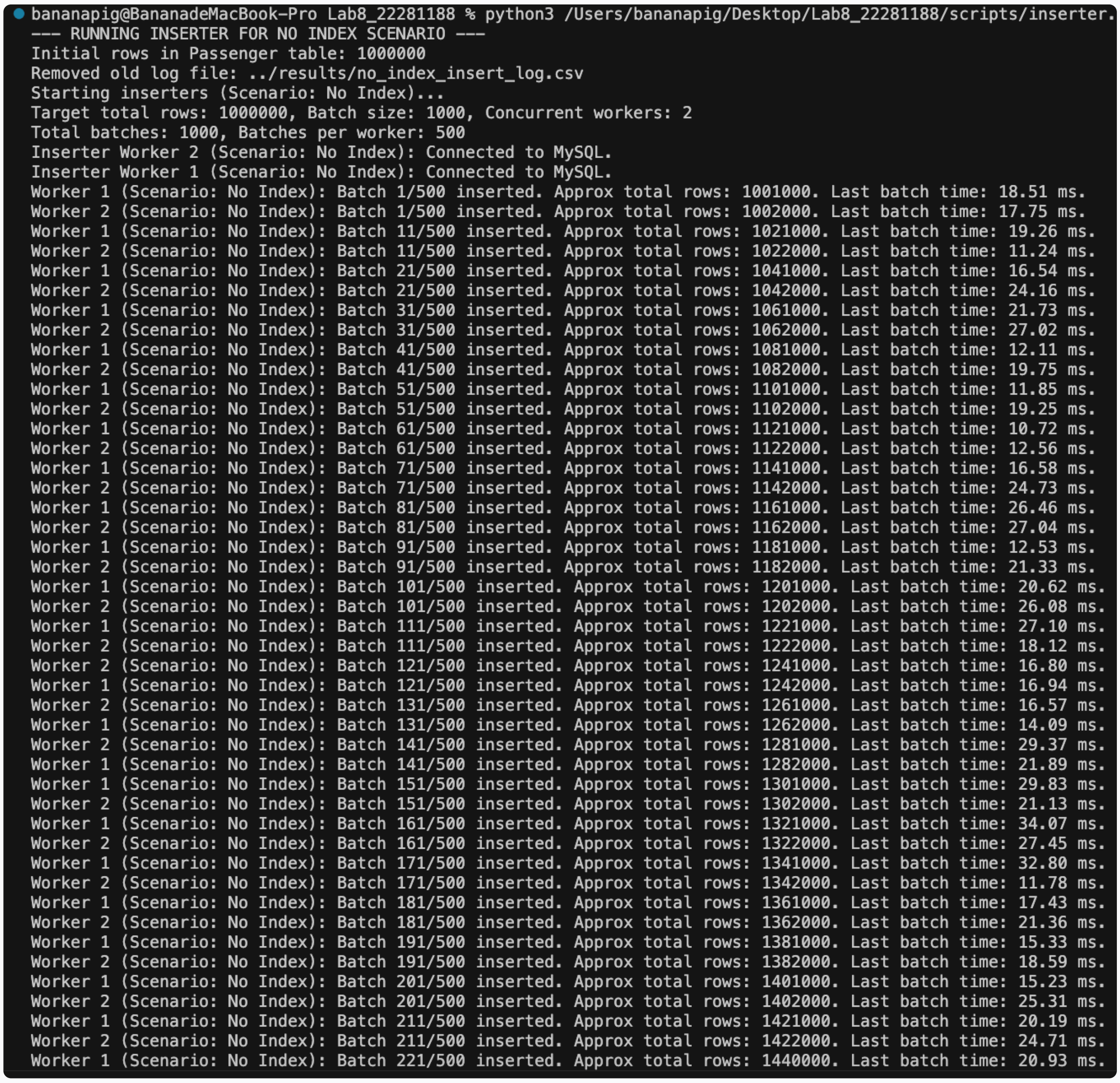
   - 清空 `Passenger` 表：

     ```
     TRUNCATE TABLE Passenger;
     ```

2. **运行插入脚本**:
    - 打开一个终端，导航到 `scripts/` 目录。
    - 修改 `inserter.py` 文件，确保 `if __name__ == '__main__':` 部分调用的是无索引场景：

      ```
      # inserter.py
      if __name__ == '__main__':
          print("--- RUNNING INSERTER FOR NO INDEX SCENARIO ---")
          run_inserters(use_index_scenario=False)
          # print("\n--- RUNNING INSERTER FOR WITH INDEX SCENARIO -
      --")
          # run_inserters(use_index_scenario=True)
      ```

    - 执行插入脚本： `python inserter.py`
    - 耗时数据将记录在 `results/no_index_insert_log.csv`。
    -

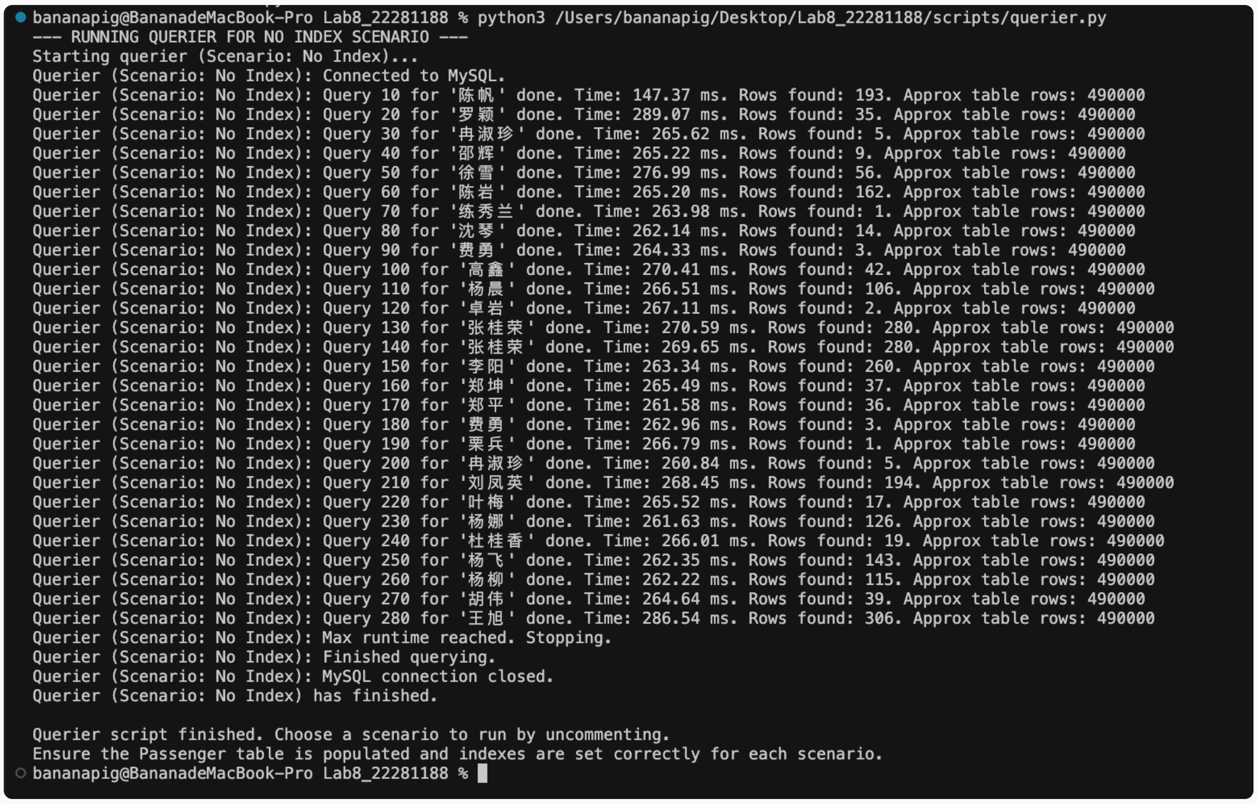


```
● bananapig@BananadeMacBook-Pro Lab8_22281188 % python3 /Users/bananapig/Desktop/Lab8_22281188/scripts/inserter.
  --- RUNNING INSERTER FOR NO INDEX SCENARIO ---
  Initial rows in Passenger table: 1000000
  Removed old log file: ../results/no_index_insert_log.csv
  Starting inserters (Scenario: No Index)...
  Target total rows: 1000000, Batch size: 1000, Concurrent workers: 2
  Total batches: 1000, Batches per worker: 500
  Inserter Worker 2 (Scenario: No Index): Connected to MySQL.
  Inserter Worker 1 (Scenario: No Index): Connected to MySQL.
  Worker 1 (Scenario: No Index): Batch 1/500 inserted. Approx total rows: 1001000. Last batch time: 18.51 ms.
  Worker 2 (Scenario: No Index): Batch 1/500 inserted. Approx total rows: 1002000. Last batch time: 17.75 ms.
  Worker 1 (Scenario: No Index): Batch 11/500 inserted. Approx total rows: 1021000. Last batch time: 19.26 ms.
  Worker 2 (Scenario: No Index): Batch 11/500 inserted. Approx total rows: 1022000. Last batch time: 11.24 ms.
  Worker 1 (Scenario: No Index): Batch 21/500 inserted. Approx total rows: 1041000. Last batch time: 16.54 ms.
  Worker 2 (Scenario: No Index): Batch 21/500 inserted. Approx total rows: 1042000. Last batch time: 24.16 ms.
  Worker 1 (Scenario: No Index): Batch 31/500 inserted. Approx total rows: 1061000. Last batch time: 21.73 ms.
  Worker 2 (Scenario: No Index): Batch 31/500 inserted. Approx total rows: 1062000. Last batch time: 27.02 ms.
  Worker 1 (Scenario: No Index): Batch 41/500 inserted. Approx total rows: 1081000. Last batch time: 12.11 ms.
  Worker 2 (Scenario: No Index): Batch 41/500 inserted. Approx total rows: 1082000. Last batch time: 19.75 ms.
  Worker 1 (Scenario: No Index): Batch 51/500 inserted. Approx total rows: 1101000. Last batch time: 11.85 ms.
  Worker 2 (Scenario: No Index): Batch 51/500 inserted. Approx total rows: 1102000. Last batch time: 19.25 ms.
  Worker 1 (Scenario: No Index): Batch 61/500 inserted. Approx total rows: 1121000. Last batch time: 10.72 ms.
  Worker 2 (Scenario: No Index): Batch 61/500 inserted. Approx total rows: 1122000. Last batch time: 12.56 ms.
  Worker 1 (Scenario: No Index): Batch 71/500 inserted. Approx total rows: 1141000. Last batch time: 16.58 ms.
  Worker 2 (Scenario: No Index): Batch 71/500 inserted. Approx total rows: 1142000. Last batch time: 24.73 ms.
  Worker 1 (Scenario: No Index): Batch 81/500 inserted. Approx total rows: 1161000. Last batch time: 26.46 ms.
  Worker 2 (Scenario: No Index): Batch 81/500 inserted. Approx total rows: 1162000. Last batch time: 27.04 ms.
  Worker 1 (Scenario: No Index): Batch 91/500 inserted. Approx total rows: 1181000. Last batch time: 12.53 ms.
  Worker 2 (Scenario: No Index): Batch 91/500 inserted. Approx total rows: 1182000. Last batch time: 21.33 ms.
  Worker 1 (Scenario: No Index): Batch 101/500 inserted. Approx total rows: 1201000. Last batch time: 20.62 ms.
  Worker 2 (Scenario: No Index): Batch 101/500 inserted. Approx total rows: 1202000. Last batch time: 26.08 ms.
  Worker 1 (Scenario: No Index): Batch 111/500 inserted. Approx total rows: 1221000. Last batch time: 27.10 ms.
  Worker 2 (Scenario: No Index): Batch 111/500 inserted. Approx total rows: 1222000. Last batch time: 18.12 ms.
  Worker 2 (Scenario: No Index): Batch 121/500 inserted. Approx total rows: 1241000. Last batch time: 16.80 ms.
  Worker 1 (Scenario: No Index): Batch 121/500 inserted. Approx total rows: 1242000. Last batch time: 16.94 ms.
  Worker 2 (Scenario: No Index): Batch 131/500 inserted. Approx total rows: 1261000. Last batch time: 16.57 ms.
  Worker 1 (Scenario: No Index): Batch 131/500 inserted. Approx total rows: 1262000. Last batch time: 14.09 ms.
  Worker 2 (Scenario: No Index): Batch 141/500 inserted. Approx total rows: 1281000. Last batch time: 29.37 ms.
  Worker 1 (Scenario: No Index): Batch 141/500 inserted. Approx total rows: 1282000. Last batch time: 21.89 ms.
  Worker 1 (Scenario: No Index): Batch 151/500 inserted. Approx total rows: 1301000. Last batch time: 29.83 ms.
  Worker 2 (Scenario: No Index): Batch 151/500 inserted. Approx total rows: 1302000. Last batch time: 21.13 ms.
  Worker 1 (Scenario: No Index): Batch 161/500 inserted. Approx total rows: 1321000. Last batch time: 34.07 ms.
  Worker 2 (Scenario: No Index): Batch 161/500 inserted. Approx total rows: 1322000. Last batch time: 27.45 ms.
  Worker 1 (Scenario: No Index): Batch 171/500 inserted. Approx total rows: 1341000. Last batch time: 32.80 ms.
  Worker 2 (Scenario: No Index): Batch 171/500 inserted. Approx total rows: 1342000. Last batch time: 11.78 ms.
  Worker 1 (Scenario: No Index): Batch 181/500 inserted. Approx total rows: 1361000. Last batch time: 17.43 ms.
  Worker 2 (Scenario: No Index): Batch 181/500 inserted. Approx total rows: 1362000. Last batch time: 21.36 ms.
  Worker 1 (Scenario: No Index): Batch 191/500 inserted. Approx total rows: 1381000. Last batch time: 15.33 ms.
  Worker 2 (Scenario: No Index): Batch 191/500 inserted. Approx total rows: 1382000. Last batch time: 18.59 ms.
  Worker 1 (Scenario: No Index): Batch 201/500 inserted. Approx total rows: 1401000. Last batch time: 15.23 ms.
  Worker 2 (Scenario: No Index): Batch 201/500 inserted. Approx total rows: 1402000. Last batch time: 25.31 ms.
  Worker 1 (Scenario: No Index): Batch 211/500 inserted. Approx total rows: 1421000. Last batch time: 20.19 ms.
  Worker 2 (Scenario: No Index): Batch 211/500 inserted. Approx total rows: 1422000. Last batch time: 24.71 ms.
  Worker 1 (Scenario: No Index): Batch 221/500 inserted. Approx total rows: 1440000. Last batch time: 20.93 ms.
```

3. **并发运行查询脚本**：
    - 在插入脚本运行时，打开**另一个**终端，导航到 `scripts/` 目录。
    - 修改 `querier.py` 文件，确保 `if __name__ == '__main__':` 部分调用的是无索引场景：

    ```python
    # querier.py
    if __name__ == '__main__':
        print("--- RUNNING QUERIER FOR NO INDEX SCENARIO ---")
        run_querier(use_index_scenario=False)
        # print("\n--- RUNNING QUERIER FOR WITH INDEX SCENARIO ---")
        # run_querier(use_index_scenario=True)
    ```

    - 执行查询脚本：`python querier.py`

- 耗时数据将记录在 `results/no_index_query_log.csv`。
- 

```
● bananapig@BananadeMacBook-Pro Lab8_22281188 % python3 /Users/bananapig/Desktop/Lab8_22281188/scripts/querier.py
  --- RUNNING QUERIER FOR NO INDEX SCENARIO ---
  Starting querier (Scenario: No Index)...
  Querier (Scenario: No Index): Connected to MySQL.
  Querier (Scenario: No Index): Query 10 for '陈帆' done. Time: 147.37 ms. Rows found: 193. Approx table rows: 490000
  Querier (Scenario: No Index): Query 20 for '罗颖' done. Time: 289.07 ms. Rows found: 35. Approx table rows: 490000
  Querier (Scenario: No Index): Query 30 for '冉淑珍' done. Time: 265.62 ms. Rows found: 5. Approx table rows: 490000
  Querier (Scenario: No Index): Query 40 for '邵辉' done. Time: 265.22 ms. Rows found: 9. Approx table rows: 490000
  Querier (Scenario: No Index): Query 50 for '徐雪' done. Time: 276.99 ms. Rows found: 56. Approx table rows: 490000
  Querier (Scenario: No Index): Query 60 for '陈岩' done. Time: 265.20 ms. Rows found: 162. Approx table rows: 490000
  Querier (Scenario: No Index): Query 70 for '练秀兰' done. Time: 263.98 ms. Rows found: 1. Approx table rows: 490000
  Querier (Scenario: No Index): Query 80 for '沈琴' done. Time: 262.14 ms. Rows found: 14. Approx table rows: 490000
  Querier (Scenario: No Index): Query 90 for '费勇' done. Time: 264.33 ms. Rows found: 3. Approx table rows: 490000
  Querier (Scenario: No Index): Query 100 for '高鑫' done. Time: 270.41 ms. Rows found: 42. Approx table rows: 490000
  Querier (Scenario: No Index): Query 110 for '杨晨' done. Time: 266.51 ms. Rows found: 106. Approx table rows: 490000
  Querier (Scenario: No Index): Query 120 for '卓岩' done. Time: 267.11 ms. Rows found: 2. Approx table rows: 490000
  Querier (Scenario: No Index): Query 130 for '张桂荣' done. Time: 270.59 ms. Rows found: 280. Approx table rows: 490000
  Querier (Scenario: No Index): Query 140 for '张桂荣' done. Time: 269.65 ms. Rows found: 280. Approx table rows: 490000
  Querier (Scenario: No Index): Query 150 for '李阳' done. Time: 263.34 ms. Rows found: 260. Approx table rows: 490000
  Querier (Scenario: No Index): Query 160 for '郑坤' done. Time: 265.49 ms. Rows found: 37. Approx table rows: 490000
  Querier (Scenario: No Index): Query 170 for '郑平' done. Time: 261.58 ms. Rows found: 36. Approx table rows: 490000
  Querier (Scenario: No Index): Query 180 for '费勇' done. Time: 262.96 ms. Rows found: 3. Approx table rows: 490000
  Querier (Scenario: No Index): Query 190 for '栗兵' done. Time: 266.79 ms. Rows found: 1. Approx table rows: 490000
  Querier (Scenario: No Index): Query 200 for '冉淑珍' done. Time: 260.84 ms. Rows found: 5. Approx table rows: 490000
  Querier (Scenario: No Index): Query 210 for '刘凤英' done. Time: 268.45 ms. Rows found: 194. Approx table rows: 490000
  Querier (Scenario: No Index): Query 220 for '叶梅' done. Time: 265.52 ms. Rows found: 17. Approx table rows: 490000
  Querier (Scenario: No Index): Query 230 for '杨娜' done. Time: 261.63 ms. Rows found: 126. Approx table rows: 490000
  Querier (Scenario: No Index): Query 240 for '杜桂香' done. Time: 266.01 ms. Rows found: 19. Approx table rows: 490000
  Querier (Scenario: No Index): Query 250 for '杨飞' done. Time: 262.35 ms. Rows found: 143. Approx table rows: 490000
  Querier (Scenario: No Index): Query 260 for '杨柳' done. Time: 262.22 ms. Rows found: 115. Approx table rows: 490000
  Querier (Scenario: No Index): Query 270 for '胡伟' done. Time: 264.64 ms. Rows found: 39. Approx table rows: 490000
  Querier (Scenario: No Index): Query 280 for '王旭' done. Time: 286.54 ms. Rows found: 306. Approx table rows: 490000
  Querier (Scenario: No Index): Max runtime reached. Stopping.
  Querier (Scenario: No Index): Finished querying.
  Querier (Scenario: No Index): MySQL connection closed.
  Querier (Scenario: No Index) has finished.

  Querier script finished. Choose a scenario to run by uncommenting.
  Ensure the Passenger table is populated and indexes are set correctly for each scenario.
○ bananapig@BananadeMacBook-Pro Lab8_22281188 % █
```

4. **监控**: 观察脚本运行，等待插入达到目标数量。查询脚本会根据配置运行一段时间或次数。

场景2: 有索引压力测试

1. **准备数据库 (有索引)**:
   - 连接到 `AirlineDB`。
   - 清空 `Passenger` 表：

     ```sql
     USE AirlineDB;
     TRUNCATE TABLE Passenger;
     ```

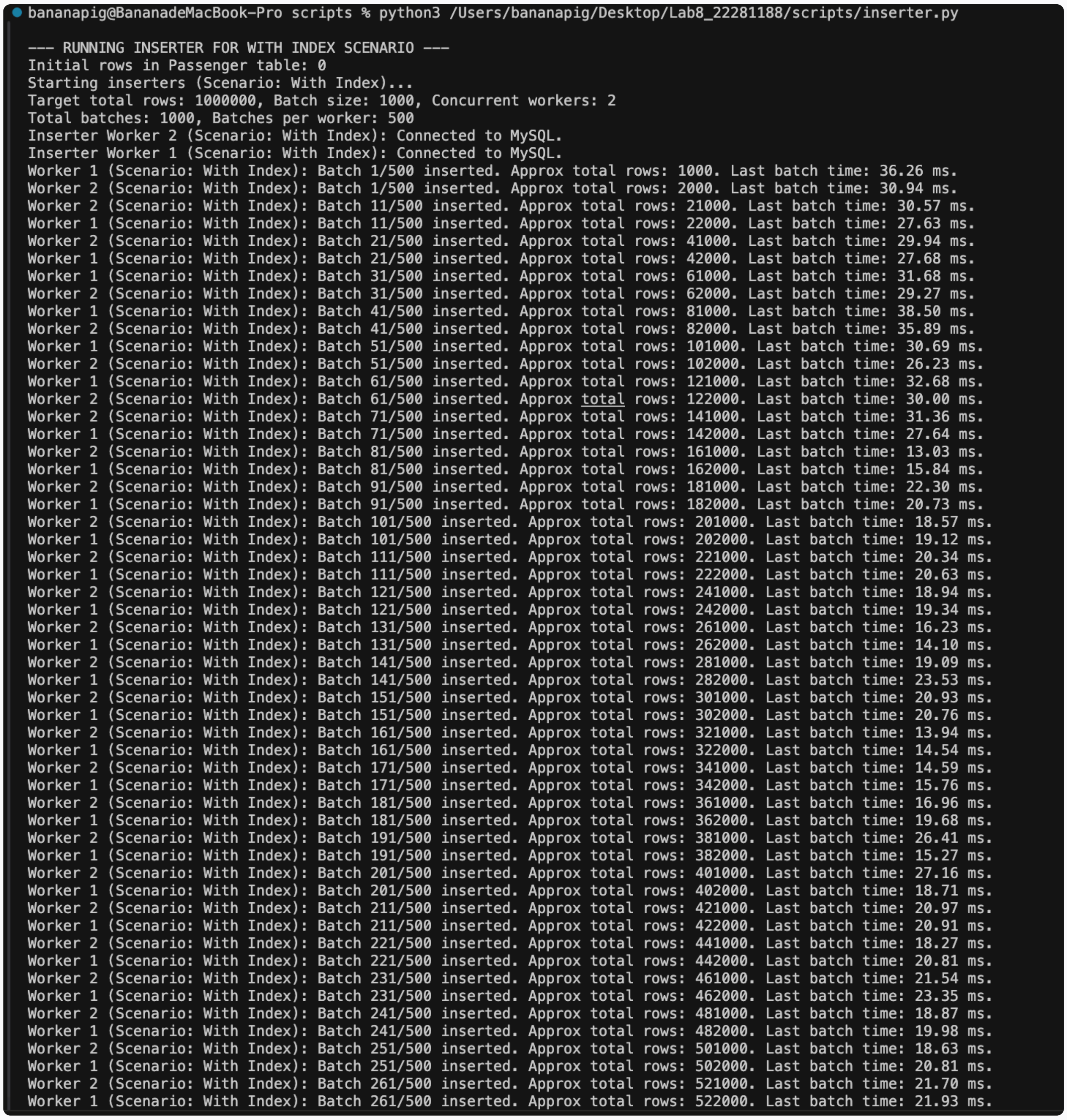
   - 为 `Passenger.name` 列创建索引，执行 `DBLab8_Add_Index_A.sql` 脚本：

     ```
     # 命令行示例
     mysql -u your_mysql_user -p AirlineDB <
     /path/to/your/project/DBLab8_Add_Index_A.sql
     ```

    - 验证索引已创建：`SHOW INDEX FROM Passenger WHERE Key_name = 'idx_passenger_name';`
2. **运行插入脚本**:
    - 打开一个终端，导航到 `scripts/` 目录。
    - 修改 `inserter.py` 文件，确保 `if __name__ == '__main__':` 部分调用的是有索引场景：

    ```
    # inserter.py
    if __name__ == '__main__':
        # print("--- RUNNING INSERTER FOR NO INDEX SCENARIO ---")
        # run_inserters(use_index_scenario=False)
        print("\n--- RUNNING INSERTER FOR WITH INDEX SCENARIO ---
    ")
        run_inserters(use_index_scenario=True)
    ```

    - 执行插入脚本：`python inserter.py`
    - 耗时数据将记录在 `results/with_index_insert_log.csv`。
    -

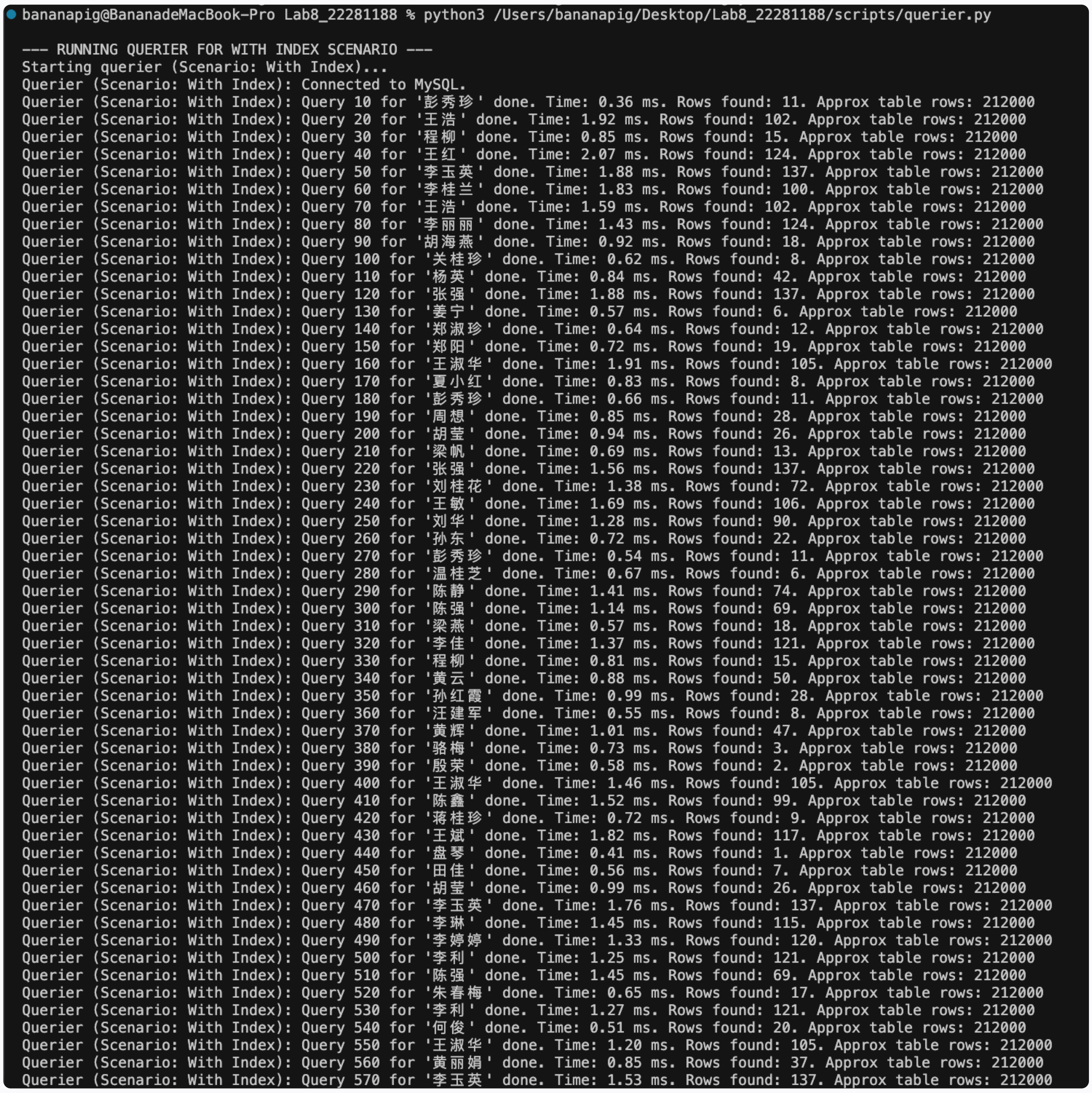


```
● bananapig@BananadeMacBook-Pro scripts % python3 /Users/bananapig/Desktop/Lab8_22281188/scripts/inserter.py

--- RUNNING INSERTER FOR WITH INDEX SCENARIO ---
Initial rows in Passenger table: 0
Starting inserters (Scenario: With Index)...
Target total rows: 1000000, Batch size: 1000, Concurrent workers: 2
Total batches: 1000, Batches per worker: 500
Inserter Worker 2 (Scenario: With Index): Connected to MySQL.
Inserter Worker 1 (Scenario: With Index): Connected to MySQL.
Worker 1 (Scenario: With Index): Batch 1/500 inserted. Approx total rows: 1000. Last batch time: 36.26 ms.
Worker 2 (Scenario: With Index): Batch 1/500 inserted. Approx total rows: 2000. Last batch time: 30.94 ms.
Worker 2 (Scenario: With Index): Batch 11/500 inserted. Approx total rows: 21000. Last batch time: 30.57 ms.
Worker 1 (Scenario: With Index): Batch 11/500 inserted. Approx total rows: 22000. Last batch time: 27.63 ms.
Worker 2 (Scenario: With Index): Batch 21/500 inserted. Approx total rows: 41000. Last batch time: 29.94 ms.
Worker 1 (Scenario: With Index): Batch 21/500 inserted. Approx total rows: 42000. Last batch time: 27.68 ms.
Worker 1 (Scenario: With Index): Batch 31/500 inserted. Approx total rows: 61000. Last batch time: 31.68 ms.
Worker 2 (Scenario: With Index): Batch 31/500 inserted. Approx total rows: 62000. Last batch time: 29.27 ms.
Worker 1 (Scenario: With Index): Batch 41/500 inserted. Approx total rows: 81000. Last batch time: 38.50 ms.
Worker 2 (Scenario: With Index): Batch 41/500 inserted. Approx total rows: 82000. Last batch time: 35.89 ms.
Worker 1 (Scenario: With Index): Batch 51/500 inserted. Approx total rows: 101000. Last batch time: 30.69 ms.
Worker 2 (Scenario: With Index): Batch 51/500 inserted. Approx total rows: 102000. Last batch time: 26.23 ms.
Worker 1 (Scenario: With Index): Batch 61/500 inserted. Approx total rows: 121000. Last batch time: 32.68 ms.
Worker 2 (Scenario: With Index): Batch 61/500 inserted. Approx total rows: 122000. Last batch time: 30.00 ms.
Worker 2 (Scenario: With Index): Batch 71/500 inserted. Approx total rows: 141000. Last batch time: 31.36 ms.
Worker 1 (Scenario: With Index): Batch 71/500 inserted. Approx total rows: 142000. Last batch time: 27.64 ms.
Worker 2 (Scenario: With Index): Batch 81/500 inserted. Approx total rows: 161000. Last batch time: 13.03 ms.
Worker 1 (Scenario: With Index): Batch 81/500 inserted. Approx total rows: 162000. Last batch time: 15.84 ms.
Worker 2 (Scenario: With Index): Batch 91/500 inserted. Approx total rows: 181000. Last batch time: 22.30 ms.
Worker 1 (Scenario: With Index): Batch 91/500 inserted. Approx total rows: 182000. Last batch time: 20.73 ms.
Worker 2 (Scenario: With Index): Batch 101/500 inserted. Approx total rows: 201000. Last batch time: 18.57 ms.
Worker 1 (Scenario: With Index): Batch 101/500 inserted. Approx total rows: 202000. Last batch time: 19.12 ms.
Worker 2 (Scenario: With Index): Batch 111/500 inserted. Approx total rows: 221000. Last batch time: 20.34 ms.
Worker 1 (Scenario: With Index): Batch 111/500 inserted. Approx total rows: 222000. Last batch time: 20.63 ms.
Worker 2 (Scenario: With Index): Batch 121/500 inserted. Approx total rows: 241000. Last batch time: 18.94 ms.
Worker 1 (Scenario: With Index): Batch 121/500 inserted. Approx total rows: 242000. Last batch time: 19.34 ms.
Worker 2 (Scenario: With Index): Batch 131/500 inserted. Approx total rows: 261000. Last batch time: 16.23 ms.
Worker 1 (Scenario: With Index): Batch 131/500 inserted. Approx total rows: 262000. Last batch time: 14.10 ms.
Worker 2 (Scenario: With Index): Batch 141/500 inserted. Approx total rows: 281000. Last batch time: 19.09 ms.
Worker 1 (Scenario: With Index): Batch 141/500 inserted. Approx total rows: 282000. Last batch time: 23.53 ms.
Worker 2 (Scenario: With Index): Batch 151/500 inserted. Approx total rows: 301000. Last batch time: 20.93 ms.
Worker 1 (Scenario: With Index): Batch 151/500 inserted. Approx total rows: 302000. Last batch time: 20.76 ms.
Worker 2 (Scenario: With Index): Batch 161/500 inserted. Approx total rows: 321000. Last batch time: 13.94 ms.
Worker 1 (Scenario: With Index): Batch 161/500 inserted. Approx total rows: 322000. Last batch time: 14.54 ms.
Worker 2 (Scenario: With Index): Batch 171/500 inserted. Approx total rows: 341000. Last batch time: 14.59 ms.
Worker 1 (Scenario: With Index): Batch 171/500 inserted. Approx total rows: 342000. Last batch time: 15.76 ms.
Worker 2 (Scenario: With Index): Batch 181/500 inserted. Approx total rows: 361000. Last batch time: 16.96 ms.
Worker 1 (Scenario: With Index): Batch 181/500 inserted. Approx total rows: 362000. Last batch time: 19.68 ms.
Worker 2 (Scenario: With Index): Batch 191/500 inserted. Approx total rows: 381000. Last batch time: 26.41 ms.
Worker 1 (Scenario: With Index): Batch 191/500 inserted. Approx total rows: 382000. Last batch time: 15.27 ms.
Worker 2 (Scenario: With Index): Batch 201/500 inserted. Approx total rows: 401000. Last batch time: 27.16 ms.
Worker 1 (Scenario: With Index): Batch 201/500 inserted. Approx total rows: 402000. Last batch time: 18.71 ms.
Worker 2 (Scenario: With Index): Batch 211/500 inserted. Approx total rows: 421000. Last batch time: 20.97 ms.
Worker 1 (Scenario: With Index): Batch 211/500 inserted. Approx total rows: 422000. Last batch time: 20.91 ms.
Worker 2 (Scenario: With Index): Batch 221/500 inserted. Approx total rows: 441000. Last batch time: 18.27 ms.
Worker 1 (Scenario: With Index): Batch 221/500 inserted. Approx total rows: 442000. Last batch time: 20.81 ms.
Worker 2 (Scenario: With Index): Batch 231/500 inserted. Approx total rows: 461000. Last batch time: 21.54 ms.
Worker 1 (Scenario: With Index): Batch 231/500 inserted. Approx total rows: 462000. Last batch time: 23.35 ms.
Worker 2 (Scenario: With Index): Batch 241/500 inserted. Approx total rows: 481000. Last batch time: 18.87 ms.
Worker 1 (Scenario: With Index): Batch 241/500 inserted. Approx total rows: 482000. Last batch time: 19.98 ms.
Worker 2 (Scenario: With Index): Batch 251/500 inserted. Approx total rows: 501000. Last batch time: 18.63 ms.
Worker 1 (Scenario: With Index): Batch 251/500 inserted. Approx total rows: 502000. Last batch time: 20.81 ms.
Worker 2 (Scenario: With Index): Batch 261/500 inserted. Approx total rows: 521000. Last batch time: 21.70 ms.
Worker 1 (Scenario: With Index): Batch 261/500 inserted. Approx total rows: 522000. Last batch time: 21.93 ms.
```

3. **并发运行查询脚本**:
   - 在插入脚本运行时，打开**另一个**终端，导航到 `scripts/` 目录。
   - 修改 `querier.py` 文件，确保 `if __name__ == '__main__':` 部分调用的是有索引场景：

```
# querier.py
if __name__ == '__main__':
    # print("--- RUNNING QUERIER FOR NO INDEX SCENARIO ---")
    # run_querier(use_index_scenario=False)
    print("\n--- RUNNING QUERIER FOR WITH INDEX SCENARIO ---
")
    run_querier(use_index_scenario=True)
```

- 执行查询脚本：`python querier.py`
- 耗时数据将记录在 `results/with_index_query_log.csv`。
-

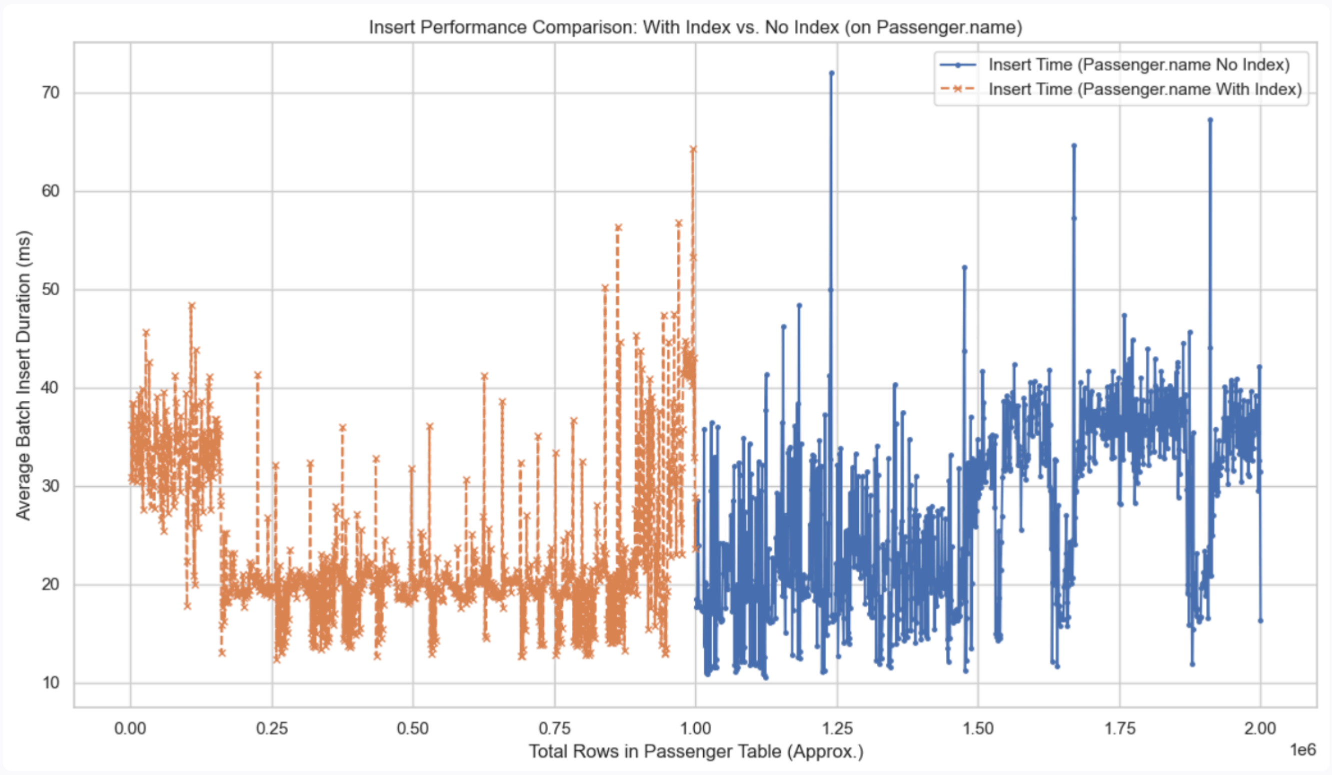


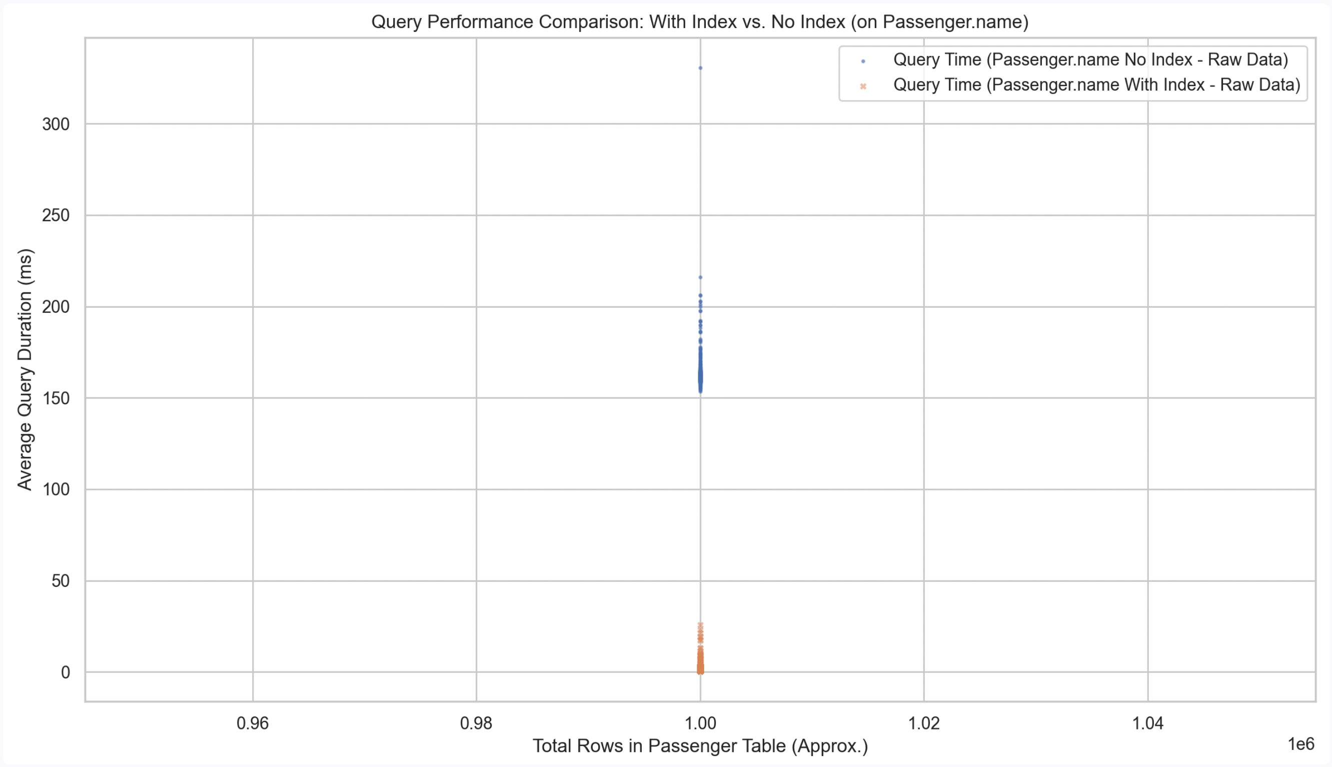
```
● bananapig@BananadeMacBook-Pro Lab8_22281188 % python3 /Users/bananapig/Desktop/Lab8_22281188/scripts/querier.py

  --- RUNNING QUERIER FOR WITH INDEX SCENARIO ---
  Starting querier (Scenario: With Index)...
  Querier (Scenario: With Index): Connected to MySQL.
  Querier (Scenario: With Index): Query 10 for '彭秀珍' done. Time: 0.36 ms. Rows found: 11. Approx table rows: 212000
  Querier (Scenario: With Index): Query 20 for '王浩' done. Time: 1.92 ms. Rows found: 102. Approx table rows: 212000
  Querier (Scenario: With Index): Query 30 for '程柳' done. Time: 0.85 ms. Rows found: 15. Approx table rows: 212000
  Querier (Scenario: With Index): Query 40 for '王红' done. Time: 2.07 ms. Rows found: 124. Approx table rows: 212000
  Querier (Scenario: With Index): Query 50 for '李玉英' done. Time: 1.88 ms. Rows found: 137. Approx table rows: 212000
  Querier (Scenario: With Index): Query 60 for '李桂兰' done. Time: 1.83 ms. Rows found: 100. Approx table rows: 212000
  Querier (Scenario: With Index): Query 70 for '王浩' done. Time: 1.59 ms. Rows found: 102. Approx table rows: 212000
  Querier (Scenario: With Index): Query 80 for '李丽丽' done. Time: 1.43 ms. Rows found: 124. Approx table rows: 212000
  Querier (Scenario: With Index): Query 90 for '胡海燕' done. Time: 0.92 ms. Rows found: 18. Approx table rows: 212000
  Querier (Scenario: With Index): Query 100 for '关桂珍' done. Time: 0.62 ms. Rows found: 8. Approx table rows: 212000
  Querier (Scenario: With Index): Query 110 for '杨英' done. Time: 0.84 ms. Rows found: 42. Approx table rows: 212000
  Querier (Scenario: With Index): Query 120 for '张强' done. Time: 1.88 ms. Rows found: 137. Approx table rows: 212000
  Querier (Scenario: With Index): Query 130 for '姜宁' done. Time: 0.57 ms. Rows found: 6. Approx table rows: 212000
  Querier (Scenario: With Index): Query 140 for '郑淑珍' done. Time: 0.64 ms. Rows found: 12. Approx table rows: 212000
  Querier (Scenario: With Index): Query 150 for '郑阳' done. Time: 0.72 ms. Rows found: 19. Approx table rows: 212000
  Querier (Scenario: With Index): Query 160 for '王淑华' done. Time: 1.91 ms. Rows found: 105. Approx table rows: 212000
  Querier (Scenario: With Index): Query 170 for '夏小红' done. Time: 0.83 ms. Rows found: 8. Approx table rows: 212000
  Querier (Scenario: With Index): Query 180 for '彭秀珍' done. Time: 0.66 ms. Rows found: 11. Approx table rows: 212000
  Querier (Scenario: With Index): Query 190 for '周想' done. Time: 0.85 ms. Rows found: 28. Approx table rows: 212000
  Querier (Scenario: With Index): Query 200 for '胡莹' done. Time: 0.94 ms. Rows found: 26. Approx table rows: 212000
  Querier (Scenario: With Index): Query 210 for '梁帆' done. Time: 0.69 ms. Rows found: 13. Approx table rows: 212000
  Querier (Scenario: With Index): Query 220 for '张强' done. Time: 1.56 ms. Rows found: 137. Approx table rows: 212000
  Querier (Scenario: With Index): Query 230 for '刘桂花' done. Time: 1.38 ms. Rows found: 72. Approx table rows: 212000
  Querier (Scenario: With Index): Query 240 for '王敏' done. Time: 1.69 ms. Rows found: 106. Approx table rows: 212000
  Querier (Scenario: With Index): Query 250 for '刘华' done. Time: 1.28 ms. Rows found: 90. Approx table rows: 212000
  Querier (Scenario: With Index): Query 260 for '孙东' done. Time: 0.72 ms. Rows found: 22. Approx table rows: 212000
  Querier (Scenario: With Index): Query 270 for '彭秀珍' done. Time: 0.54 ms. Rows found: 11. Approx table rows: 212000
  Querier (Scenario: With Index): Query 280 for '温桂芝' done. Time: 0.67 ms. Rows found: 6. Approx table rows: 212000
  Querier (Scenario: With Index): Query 290 for '陈静' done. Time: 1.41 ms. Rows found: 74. Approx table rows: 212000
  Querier (Scenario: With Index): Query 300 for '陈强' done. Time: 1.14 ms. Rows found: 69. Approx table rows: 212000
  Querier (Scenario: With Index): Query 310 for '梁燕' done. Time: 0.57 ms. Rows found: 18. Approx table rows: 212000
  Querier (Scenario: With Index): Query 320 for '李佳' done. Time: 1.37 ms. Rows found: 121. Approx table rows: 212000
  Querier (Scenario: With Index): Query 330 for '程柳' done. Time: 0.81 ms. Rows found: 15. Approx table rows: 212000
  Querier (Scenario: With Index): Query 340 for '黄云' done. Time: 0.88 ms. Rows found: 50. Approx table rows: 212000
  Querier (Scenario: With Index): Query 350 for '孙红霞' done. Time: 0.99 ms. Rows found: 28. Approx table rows: 212000
  Querier (Scenario: With Index): Query 360 for '汪建军' done. Time: 0.55 ms. Rows found: 8. Approx table rows: 212000
  Querier (Scenario: With Index): Query 370 for '黄辉' done. Time: 1.01 ms. Rows found: 47. Approx table rows: 212000
  Querier (Scenario: With Index): Query 380 for '骆梅' done. Time: 0.73 ms. Rows found: 3. Approx table rows: 212000
  Querier (Scenario: With Index): Query 390 for '殷荣' done. Time: 0.58 ms. Rows found: 2. Approx table rows: 212000
  Querier (Scenario: With Index): Query 400 for '王淑华' done. Time: 1.46 ms. Rows found: 105. Approx table rows: 212000
  Querier (Scenario: With Index): Query 410 for '陈鑫' done. Time: 1.52 ms. Rows found: 99. Approx table rows: 212000
  Querier (Scenario: With Index): Query 420 for '蒋桂珍' done. Time: 0.72 ms. Rows found: 9. Approx table rows: 212000
  Querier (Scenario: With Index): Query 430 for '王斌' done. Time: 1.82 ms. Rows found: 117. Approx table rows: 212000
  Querier (Scenario: With Index): Query 440 for '盘琴' done. Time: 0.41 ms. Rows found: 1. Approx table rows: 212000
  Querier (Scenario: With Index): Query 450 for '田佳' done. Time: 0.56 ms. Rows found: 7. Approx table rows: 212000
  Querier (Scenario: With Index): Query 460 for '胡莹' done. Time: 0.99 ms. Rows found: 26. Approx table rows: 212000
  Querier (Scenario: With Index): Query 470 for '李玉英' done. Time: 1.76 ms. Rows found: 137. Approx table rows: 212000
  Querier (Scenario: With Index): Query 480 for '李琳' done. Time: 1.45 ms. Rows found: 115. Approx table rows: 212000
  Querier (Scenario: With Index): Query 490 for '李婷婷' done. Time: 1.33 ms. Rows found: 120. Approx table rows: 212000
  Querier (Scenario: With Index): Query 500 for '李利' done. Time: 1.25 ms. Rows found: 121. Approx table rows: 212000
  Querier (Scenario: With Index): Query 510 for '陈强' done. Time: 1.45 ms. Rows found: 69. Approx table rows: 212000
  Querier (Scenario: With Index): Query 520 for '朱春梅' done. Time: 0.65 ms. Rows found: 17. Approx table rows: 212000
  Querier (Scenario: With Index): Query 530 for '李利' done. Time: 1.27 ms. Rows found: 121. Approx table rows: 212000
  Querier (Scenario: With Index): Query 540 for '何俊' done. Time: 0.51 ms. Rows found: 20. Approx table rows: 212000
  Querier (Scenario: With Index): Query 550 for '王淑华' done. Time: 1.20 ms. Rows found: 105. Approx table rows: 212000
  Querier (Scenario: With Index): Query 560 for '黄丽娟' done. Time: 0.85 ms. Rows found: 37. Approx table rows: 212000
  Querier (Scenario: With Index): Query 570 for '李玉英' done. Time: 1.53 ms. Rows found: 137. Approx table rows: 212000
```

4. **监控**: 同无索引场景。

数据分析

- 测试完成后，`results/` 目录下会生成4个CSV日志文件。
- 使用实验报告（`Lab8_Report_22281188.md`）中提供的Python绘图代码片段（或您选择的其他工具如Excel）来分析这些数据并生成性能对比图表。将图表和分析结果填入实验报告。【删除】
-

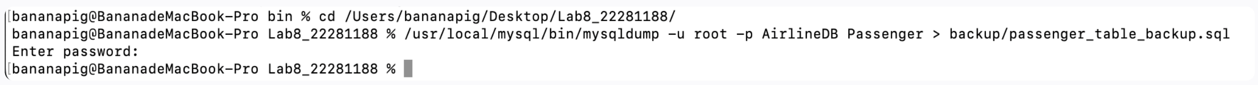


- 

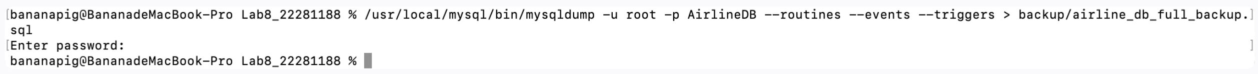
第二部分：备份与日志初步实验

1. **数据备份与恢复:**
   - 使用 `mysqldump` 工具进行单表和整库备份，将备份文件存放在 `backup/` 目录。

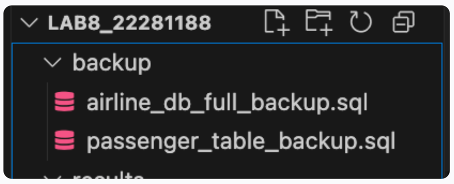
```
# 备份 Passenger 表
# 我这里需要使用mysqlump的绝对路径去做，所以在我的电脑上
（MacOS：/usr/local/mysql/bin/mysqldump）
# 并且注意：一定要在我的Lab8_22281188的文件目录下去完成
mysqldump -u root -p AirlineDB Passenger >
backup/passenger_table_backup.sql
/usr/local/mysql/bin/mysqldump -u root -p AirlineDB Passenger
> backup/passenger_table_backup.sql

# 备份整个 AirlineDB 数据库
/usr/local/mysql/bin/mysqldump -u root -p AirlineDB --
routines --events --triggers >
backup/airline_db_full_backup.sql
```

- 

```
bananapig@BananadeMacBook-Pro bin % cd /Users/bananapig/Desktop/Lab8_22281188/
bananapig@BananadeMacBook-Pro Lab8_22281188 % /usr/local/mysql/bin/mysqldump -u root -p AirlineDB Passenger > backup/passenger_table_backup.sql
Enter password:
bananapig@BananadeMacBook-Pro Lab8_22281188 % 
```

- 

```
bananapig@BananadeMacBook-Pro Lab8_22281188 % /usr/local/mysql/bin/mysqldump -u root -p AirlineDB --routines --events --triggers > backup/airline_db_full_backup.
sql
Enter password:
bananapig@BananadeMacBook-Pro Lab8_22281188 % 
```



- 尝试在另一个（测试）数据库或另一台机器上使用这些备份文件恢复数据，并验证恢复的完整性。

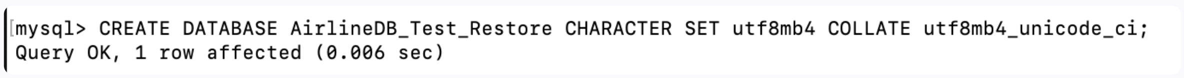
```
# 恢复示例
mysql -u root -p TargetDB < backup/passenger_table_backup.sql
```

  - 首先先进入当前数据库

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

  - 执行以下SQL命令来创建新数据库。将其命名为 `AirlineDB_Test_Restore`：

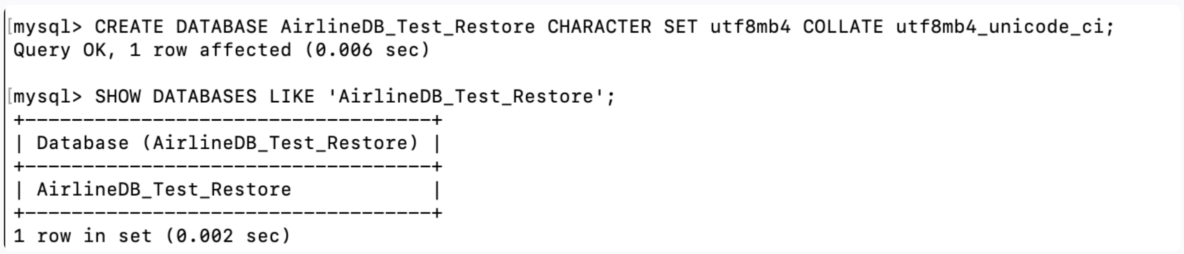
```sql
CREATE DATABASE AirlineDB_Test_Restore CHARACTER SET
utf8mb4 COLLATE utf8mb4_unicode_ci;
```

- 

```
mysql> CREATE DATABASE AirlineDB_Test_Restore CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
Query OK, 1 row affected (0.006 sec)
```

- 检查是否执行创建成功

```sql
SHOW DATABASES LIKE 'AirlineDB_Test_Restore';
```

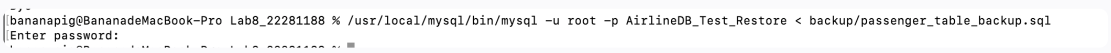
- 

```
mysql> CREATE DATABASE AirlineDB_Test_Restore CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
Query OK, 1 row affected (0.006 sec)

mysql> SHOW DATABASES LIKE 'AirlineDB_Test_Restore';
+-----------------------------------+
| Database (AirlineDB_Test_Restore) |
+-----------------------------------+
| AirlineDB_Test_Restore            |
+-----------------------------------+
1 row in set (0.002 sec)
```

```sql
EXIT;
```

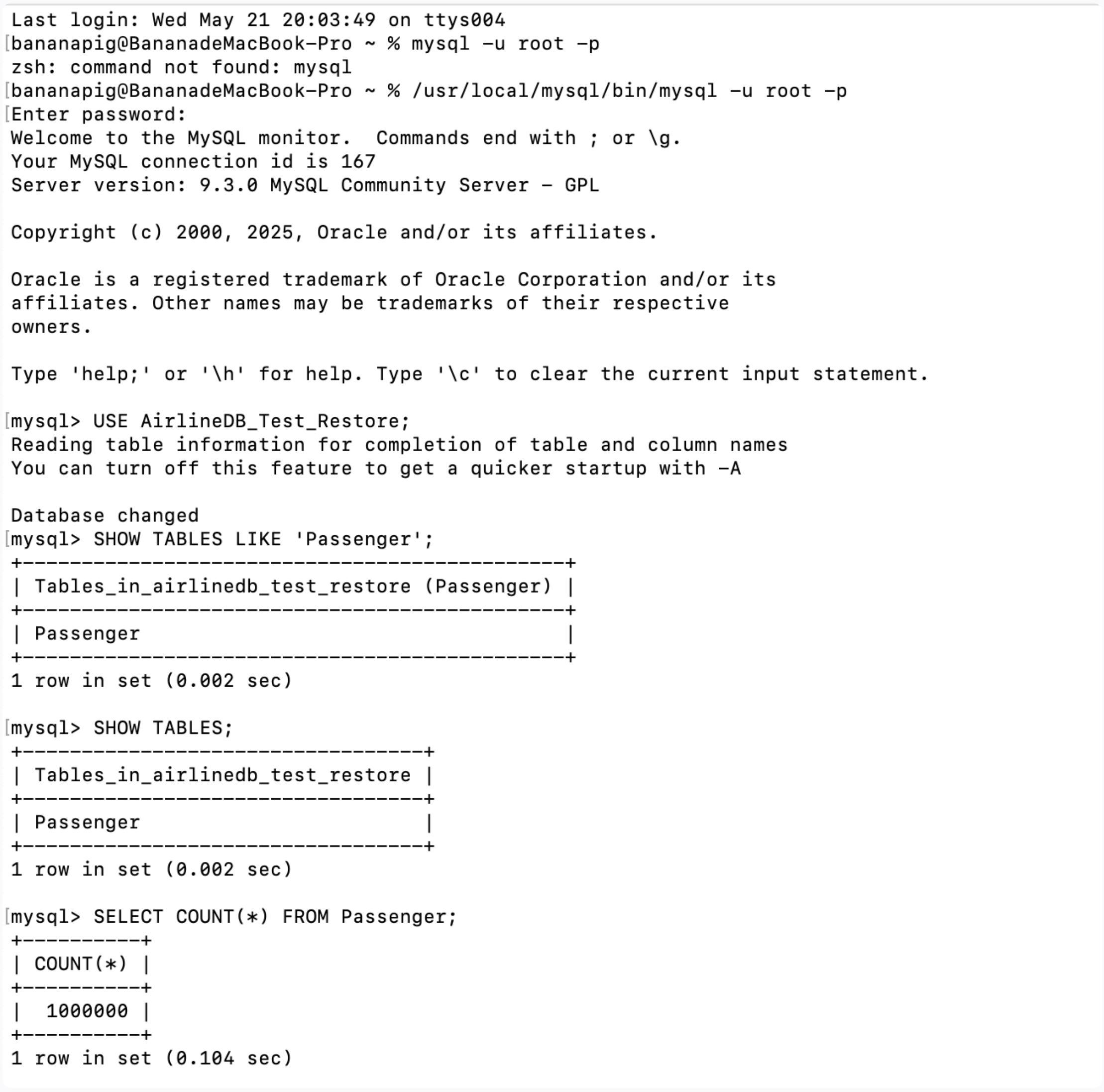
- 而后恢复恢复备份数据到新数据库
  - 情况 A：恢复单个表的备份（`passenger_table_backup.sql`）到 `AirlineDB_Test_Restore`
  - 确保在 `Lab8_22281188/` 目录下。

    ```
    /usr/local/mysql/bin/mysql -u root -p
    AirlineDB_Test_Restore <
    backup/passenger_table_backup.sql
    ```

  - 

    ```
    bananapig@BananadeMacBook-Pro Lab8_22281188 % /usr/local/mysql/bin/mysql -u root -p AirlineDB_Test_Restore < backup/passenger_table_backup.sql
    Enter password:
    ```

  - 检查结果
  -

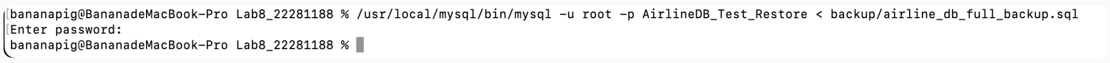


```
Last login: Wed May 21 20:03:49 on ttys004
bananapig@BananadeMacBook-Pro ~ % mysql -u root -p
zsh: command not found: mysql
bananapig@BananadeMacBook-Pro ~ % /usr/local/mysql/bin/mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 167
Server version: 9.3.0 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> USE AirlineDB_Test_Restore;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES LIKE 'Passenger';
+-----------------------------------------------+
| Tables_in_airlinedb_test_restore (Passenger) |
+-----------------------------------------------+
| Passenger                                     |
+-----------------------------------------------+
1 row in set (0.002 sec)

mysql> SHOW TABLES;
+-----------------------------------+
| Tables_in_airlinedb_test_restore |
+-----------------------------------+
| Passenger                         |
+-----------------------------------+
1 row in set (0.002 sec)

mysql> SELECT COUNT(*) FROM Passenger;
+----------+
| COUNT(*) |
+----------+
|  1000000 |
+----------+
1 row in set (0.104 sec)
```

- 恢复整个数据库的备份（`airline_db_full_backup.sql`）到 `AirlineDB_Test_Restore`

```
mysql -u root -p AirlineDB_Test_Restore < backup/airline_db_full_backup.sql
```

-



```
bananapig@BananadeMacBook-Pro Lab8_22281188 % /usr/local/mysql/bin/mysql -u root -p AirlineDB_Test_Restore < backup/airline_db_full_backup.sql
Enter password:
bananapig@BananadeMacBook-Pro Lab8_22281188 %
```

2. **数据库日志分析 (MySQL Binary Log):**
   - 首先，我在mac上需要先配置my.inf，参考https://github.com/songdeveloper/my.cnf/blob/master/my.cnf进行配置，而后更新完在重启mysql，经过下面语句的查询表示配置成功（中间非常曲折。。。。
   - **验证Binary Log是否已启用：**
     - 登录到MySQL客户端：

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

- 在MySQL提示符下执行以下SQL命令：

  ```
  SHOW VARIABLES LIKE 'log_bin';
  ```

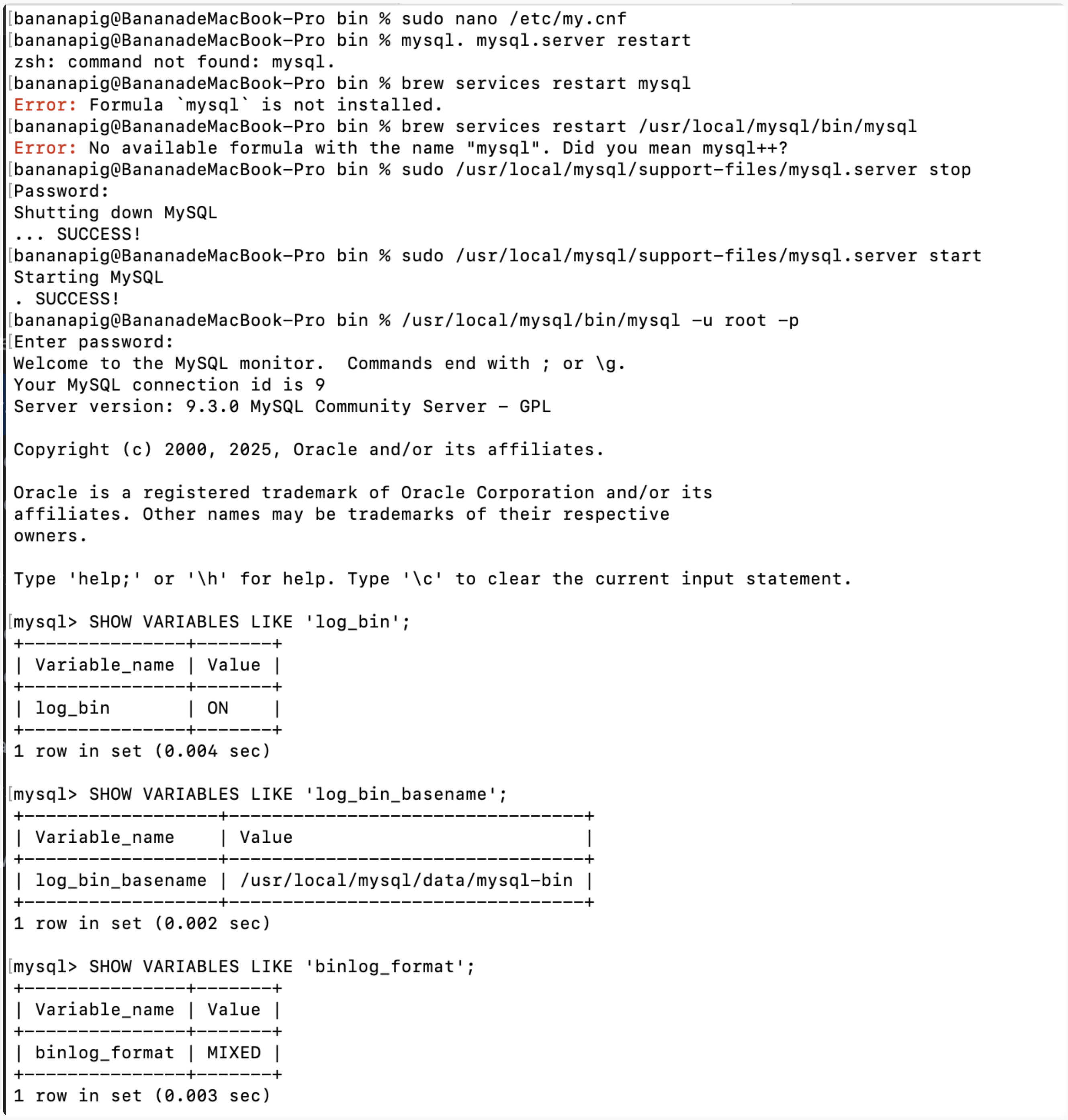
  如果输出结果中 `Value` 列为 `ON`，则表示binlog已成功启用。

  ```
  SHOW VARIABLES LIKE 'log_bin_basename';
  ```

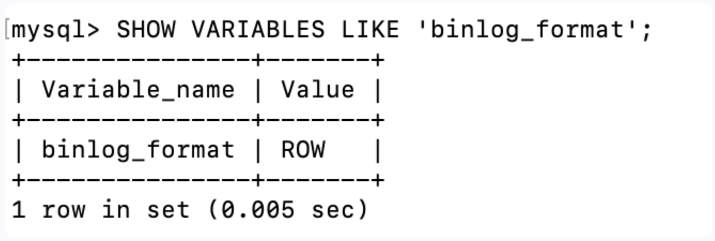
  这将显示binlog文件的基础路径和名称前缀。

  ```
  SHOW VARIABLES LIKE 'binlog_format';
  ```

  确认 `Value` 是 `ROW`



```
bananapig@BananadeMacBook-Pro bin % sudo nano /etc/my.cnf
bananapig@BananadeMacBook-Pro bin % mysql. mysql.server restart
zsh: command not found: mysql.
bananapig@BananadeMacBook-Pro bin % brew services restart mysql
Error: Formula `mysql` is not installed.
bananapig@BananadeMacBook-Pro bin % brew services restart /usr/local/mysql/bin/mysql
Error: No available formula with the name "mysql". Did you mean mysql++?
bananapig@BananadeMacBook-Pro bin % sudo /usr/local/mysql/support-files/mysql.server stop
Password:
Shutting down MySQL
... SUCCESS!
bananapig@BananadeMacBook-Pro bin % sudo /usr/local/mysql/support-files/mysql.server start
Starting MySQL
. SUCCESS!
bananapig@BananadeMacBook-Pro bin % /usr/local/mysql/bin/mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.3.0 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> SHOW VARIABLES LIKE 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
1 row in set (0.004 sec)

mysql> SHOW VARIABLES LIKE 'log_bin_basename';
+------------------+----------------------------------+
| Variable_name    | Value                            |
+------------------+----------------------------------+
| log_bin_basename | /usr/local/mysql/data/mysql-bin |
+------------------+----------------------------------+
1 row in set (0.002 sec)

mysql> SHOW VARIABLES LIKE 'binlog_format';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | MIXED |
+---------------+-------+
1 row in set (0.003 sec)
```



```
mysql> SHOW VARIABLES LIKE 'binlog_format';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | ROW   |
+---------------+-------+
1 row in set (0.005 sec)
```

对 `Passenger` 表执行数据修改操作

在binlog启用后，您对数据库所做的任何数据更改都会被记录下来。

1. 登录到MySQL客户端。
2. 切换到实验数据库：

```
USE AirlineDB;
```

3. 执行一些数据修改操作。例如：
   - 插入数据 (INSERT)

     ```
     INSERT INTO Passenger (name, id_card_number,
     phone_number, email, frequent_flyer_number)
     VALUES ('Binlog User One', '11122219900101333X',
     '13100000001', 'user1.binlog@example.com', 'BLG001');
     ```

   - 更新数据 (UPDATE)

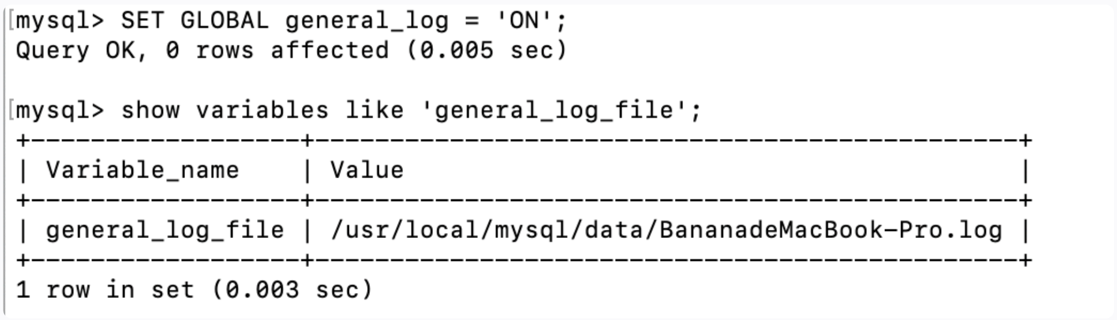
     ```
     UPDATE Passenger
     SET phone_number = '13100000099', email =
     'user1.binlog.updated@example.com'
     WHERE name = 'Binlog User One';
     ```

   - 删除数据 (DELETE)

     ```
     DELETE FROM Passenger WHERE name = 'Binlog User One';
     ```

4. 查看日志文件输出
   - 在mysql输入

     ```
     SET GLOBAL general_log = 'ON';
     show variables like 'general_log_file';
     ```



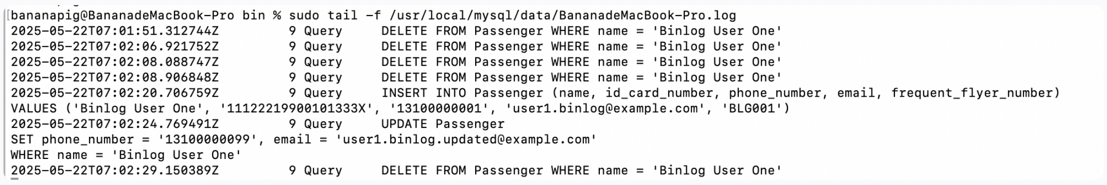
```
mysql> SET GLOBAL general_log = 'ON';
Query OK, 0 rows affected (0.005 sec)

mysql> show variables like 'general_log_file';
+------------------+-----------------------------------------------+
| Variable_name    | Value                                         |
+------------------+-----------------------------------------------+
| general_log_file | /usr/local/mysql/data/BananadeMacBook-Pro.log |
+------------------+-----------------------------------------------+
1 row in set (0.003 sec)
```

   - 在终端输入

```
tail -f /usr/local/var/mysql/SunPing.log
```

- 而后便可以看到日志：

```
bananapig@BananadeMacBook-Pro bin % sudo tail -f
/usr/local/mysql/data/BananadeMacBook-Pro.log
2025-05-22T07:01:51.312744Z      9 Query DELETE FROM
Passenger WHERE name = 'Binlog User One'
2025-05-22T07:02:06.921752Z      9 Query DELETE FROM
Passenger WHERE name = 'Binlog User One'
2025-05-22T07:02:08.088747Z      9 Query DELETE FROM
Passenger WHERE name = 'Binlog User One'
2025-05-22T07:02:08.906848Z      9 Query DELETE FROM
Passenger WHERE name = 'Binlog User One'
2025-05-22T07:02:20.706759Z      9 Query INSERT INTO
Passenger (name, id_card_number, phone_number, email,
frequent_flyer_number)
VALUES ('Binlog User One', '11122219900101333X',
'13100000001', 'user1.binlog@example.com', 'BLG001')
2025-05-22T07:02:24.769491Z      9 Query UPDATE
Passenger
SET phone_number = '13100000099', email =
'user1.binlog.updated@example.com'
WHERE name = 'Binlog User One'
2025-05-22T07:02:29.150389Z      9 Query DELETE FROM
Passenger WHERE name = 'Binlog User One'
```

- 

```
bananapig@BananadeMacBook-Pro bin % sudo tail -f /usr/local/mysql/data/BananadeMacBook-Pro.log
2025-05-22T07:01:51.312744Z         9 Query     DELETE FROM Passenger WHERE name = 'Binlog User One'
2025-05-22T07:02:06.921752Z         9 Query     DELETE FROM Passenger WHERE name = 'Binlog User One'
2025-05-22T07:02:08.088747Z         9 Query     DELETE FROM Passenger WHERE name = 'Binlog User One'
2025-05-22T07:02:08.906848Z         9 Query     DELETE FROM Passenger WHERE name = 'Binlog User One'
2025-05-22T07:02:20.706759Z         9 Query     INSERT INTO Passenger (name, id_card_number, phone_number, email, frequent_flyer_number)
VALUES ('Binlog User One', '11122219900101333X', '13100000001', 'user1.binlog@example.com', 'BLG001')
2025-05-22T07:02:24.769491Z         9 Query     UPDATE Passenger
SET phone_number = '13100000099', email = 'user1.binlog.updated@example.com'
WHERE name = 'Binlog User One'
2025-05-22T07:02:29.150389Z         9 Query     DELETE FROM Passenger WHERE name = 'Binlog User One'
```

四、心得体会

1. **索引的重要性：** 实验直观地展示了索引对查询性能的巨大提升。在无索引的情况下，随着数据量的增加，查询时间显著增长；而添加索引后，查询速度得到了质的飞跃。同时，我也理解到索引并非没有代价，它会占用额外的存储空间，并在数据插入、更新、删除时带来一定的性能开销，因为数据库需要维护索引结构。因此，在实际应用中，需要权衡查询需求和写操作的频率，合理创建索引。
2. **压力测试的意义：** 通过模拟并发插入和查询，我体会到了压力测试对于评估系统瓶颈和稳定性的重要性。虽然本次实验的压力测试脚本相对简单，但它揭示了在高并发下数据库可能出现的性能变化。在更复杂的系统中，压力测试是发现潜在问题、优化性能的关键环节。
3. **数据备份与恢复的必要性：** 实践 `mysqldump` 进行备份和恢复的过程，让我认识到数据备份是数据安全的生命线。无论是硬件故障、软件错误还是人为误操作，可靠的备份和恢复机制都能最大限度地减少数据丢失的风险。定期备份并验证备份的可用性是数据库管理员的基本职责。
4. **日志的价值：** 初步接触MySQL的日志系统（无论是General Query Log还是Binlog），让我了解到日志不仅可以用于问题排查和性能分析，更是数据恢复（特别是基于时间点的恢复Point-in-Time Recovery, PITR，通常结合Binlog实现）和数据审计的关键。配置和管理日志虽然需要一定的学习和实践（我在配置Binlog时就遇到了一些波折，需要查阅资料并仔细调整配置文件）