实验十：mysql 的查询处理与查询优化

一、实验目的

- 理解并掌握 mysql 使用 EXPLAIN 命令查看执行计划的方法，包括输出结果中各字段的含义

二、实验要求

- 自行完成，无需提交实验报告

三、实验内容与步骤

1.EXPLAIN 语法

EXPLAIN [FORMAT=traditional | tree | json] SELECT stmt | INSERT stmt | UPDATE stmt | DELETE stmt;

（1）EXPLAIN、DESCRIBE 和 DESC（为 DESCRIBE 缩写）三者效果等同，彼此之间可互换，下同

```
mysql> explain s1;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int          | NO   | PRI | NULL    | auto_increment |
| key1        | varchar(100) | YES  | MUL | NULL    |                |
| key2        | int          | YES  | UNI | NULL    |                |
| key3        | varchar(100) | YES  | MUL | NULL    |                |
| key_part1   | varchar(100) | YES  | MUL | NULL    |                |
| key_part2   | varchar(100) | YES  |     | NULL    |                |
| key_part3   | varchar(100) | YES  |     | NULL    |                |
| common_field | varchar(100) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
8 rows in set (0.05 sec)

mysql> desc s1;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int          | NO   | PRI | NULL    | auto_increment |
| key1        | varchar(100) | YES  | MUL | NULL    |                |
| key2        | int          | YES  | UNI | NULL    |                |
| key3        | varchar(100) | YES  | MUL | NULL    |                |
| key_part1   | varchar(100) | YES  | MUL | NULL    |                |
| key_part2   | varchar(100) | YES  |     | NULL    |                |
| key_part3   | varchar(100) | YES  |     | NULL    |                |
| common_field | varchar(100) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

（2）例子：EXPLAIN SELECT * FROM INVENTORIES;

```
mysql> EXPLAIN SELECT * FROM INVENTORIES;
+----+-------------+-------------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table       | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | INVENTORIES | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 1112 |   100.00 | NULL  |
+----+-------------+-------------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- 输出格式默认为表格（tabular），也即使 traditional
- 上述表格有 12 列，每列取值的含义可通过后面例子（也是练习）的说明来理解

（3）指定输出格式：format=traditional | tree | json

- EXPLAIN SELECT stmt; ---默认输出格式为 traditonal，即表格形式，等同于 EXPLAIN FORMAT=traditional
- EXPLAIN FORMAT =TREE SELECT stmt; --指定输出格式为树形

```
mysql> EXPLAIN FORMAT=TREE SELECT * FROM INVENTORIES;
+----------------------------------------------------+
| EXPLAIN                                            |
+----------------------------------------------------+
| -> Table scan on INVENTORIES  (cost=112 rows=1112)
 |
+----------------------------------------------------+
1 row in set (0.01 sec)
```

- EXPLAIN FORMAT =JSON SELECT stmt; --指定输出格式为 json

- json 有两种形式，查询当前 json 的值：SELECT @@explain_json_format_version;

```
mysql> SELECT @@explain_json_format_version;
+--------------------------------+
| @@explain_json_format_version |
+--------------------------------+
|                              1 |
+--------------------------------+
1 row in set (0.00 sec)
```

  - 这是 json 默认值 1，它显示的正是上面的格式。
  - json 只有两个值可选：1 或 2

- 使用 SET 命令设置值为 2 的 json 值，它对应另外一种 json 格式：SET @@explain_json_format_version=2;

```
mysql> SET @@explain_json_format_version=2;
Query OK, 0 rows affected (0.01 sec)
```

发布语句：EXPLAIN FORMAT =JSON SELECT * FROM INVENTORIES;

显示当前 EXPLAIN 的输出格式命令：SELECT @@EXPLAIN_FORMAT;

- 可以设置当前 EXPLAIN 的输出格式：SET @@EXPLAIN_FORMAT= [TREE | JSON]　--TREE 和 JSON 任选一

```
mysql> SET @@EXPLAIN_FORMAT=TREE;
Query OK, 0 rows affected (0.00 sec)
```

再执行 EXPLAIN SELECT stmt;时将以该设置显示，无需加上 format=tree

```
mysql> EXPLAIN SELECT * FROM INVENTORIES;
+----------------------------------------------------+
| EXPLAIN                                            |
+----------------------------------------------------+
| -> Table scan on INVENTORIES  (cost=112 rows=1112)
 |
+----------------------------------------------------+
1 row in set (0.00 sec)
```

效果等于同于：

```
mysql> EXPLAIN FORMAT=TREE SELECT * FROM INVENTORIES;
+----------------------------------------------------+
| EXPLAIN                                            |
+----------------------------------------------------+
| -> Table scan on INVENTORIES  (cost=112 rows=1112)
 |
+----------------------------------------------------+
1 row in set (0.00 sec)
```

2.EXPLAIN ANALYZE 语法

EXPLAIN ANALYZE [format=tree | json] SELECT stmt | INSERT stmt | UPDATE stmt | DELETE stmt;　---输出格式为 tree

EXPLAIN ANALYZE 与 EXPLAIN 的区别

功能差异

1. EXPLAIN：
   - 提供查询计划的估计信息，包括使用的索引、连接方式、行数估计等。
   - 不实际执行查询，因此不会对数据库数据产生影响。
   - 适合在开发阶段快速验证查询计划，避免执行开销 1 2 。
2. EXPLAIN ANALYZE：
   - 展示查询计划的同时，实际执行查询并显示每一步的执行时间和实际扫描的行数等详细性能数据。
   - 可以帮助开发者精准定位问题，如索引失效或统计信息偏差。
   - 适合在生产环境中进行调优，但需要注意实际执行开销 1 2 。

使用场景

- EXPLAIN：适用于开发阶段，快速验证查询计划，避免实际执行开销。
- EXPLAIN ANALYZE：适用于生产环境调优，通过实际执行数据精准定位问题，优化查询性能。

语法和示例

- EXPLAIN： `EXPLAIN SELECT * FROM table_name WHERE condition;`
- EXPLAIN ANALYZE： `EXPLAIN ANALYZE SELECT * FROM table_name WHERE condition;`

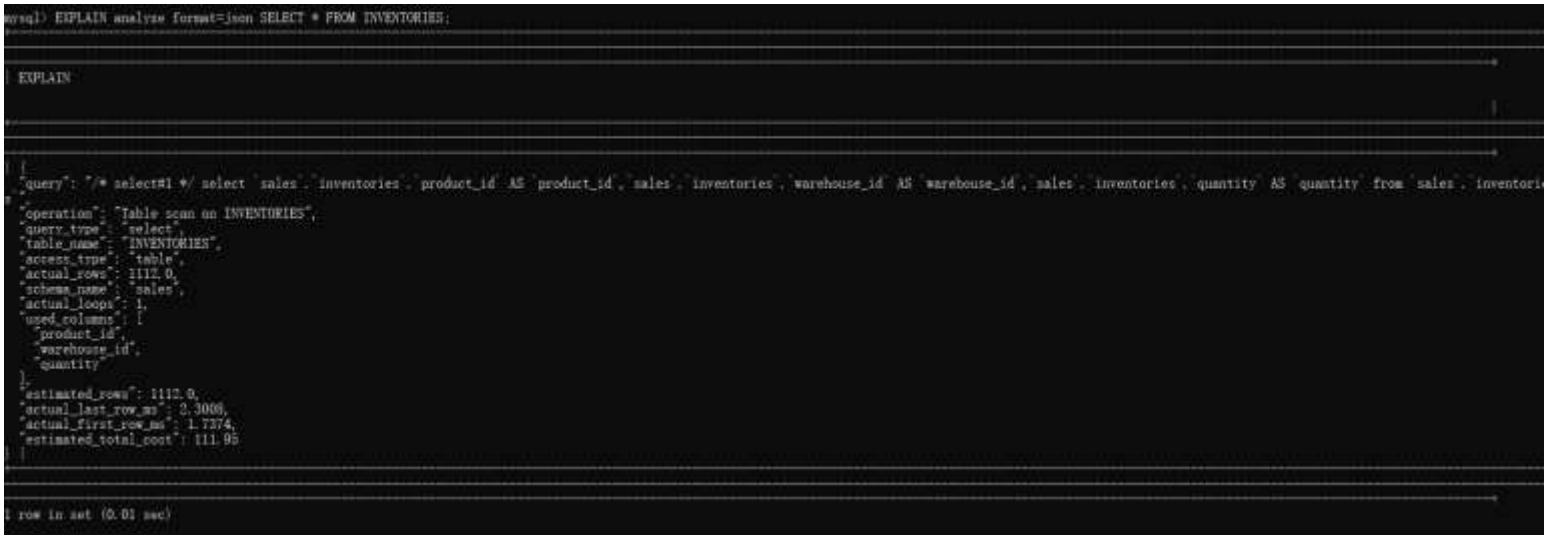
通过这些区别，开发者可以根据不同的需求选择合适的工具来优化SQL查询。

```
mysql> EXPLAIN FORMAT=TREE SELECT * FROM INVENTORIES;
+-------------------------------------------------------+
| EXPLAIN                                               |
+-------------------------------------------------------+
| -> Table scan on INVENTORIES  (cost=112 rows=1112)
 |
+-------------------------------------------------------+
1 row in set (0.00 sec)

mysql> EXPLAIN analyze SELECT * FROM INVENTORIES;
+-----------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                   |
+-----------------------------------------------------------------------------------------------------------+
| -> Table scan on INVENTORIES  (cost=112 rows=1112) (actual time=9.69..13 rows=1112 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------+
1 row in set (0.12 sec)
```

对照前述分析上述命令中 explain 和 expalin analyze 的不同

- EXPLAIN ANALYZE 的输出格式为 tree 或 json，默认为 tree，但不支持表格（与 explain 不同）



```
mysql> EXPLAIN analyze format=traditional SELECT * FROM INVENTORIES;
ERROR 1235 (42000): This version of MySQL doesn't yet support 'EXPLAIN ANALYZE with TRADITIONAL format'
```

- json 格式的设置同 explain 中的说明

3.按以下教程完成文中的所有操作，理解并掌握输出结果中各字段的含义

- 分析查询语句 EXPLAIN 详解

  https://blog.csdn.net/m0_51295655/article/details/123025861

四、实验练习

根据上面的知识解读以下实例。

1. explain analyze select * from inventories inv, warehouses wh where inv.warehouse_id=wh.warehouse_id;

```
mysql> explain analyze select * from inventories inv, warehouses wh where inv.warehouse_id=wh.warehouse_id;
+--------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                |
+--------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=133 rows=1112) (actual time=0.319..7.38 rows=1112 loops=1)
    -> Table scan on wh  (cost=1.15 rows=9) (actual time=0.0587..0.0858 rows=9 loops=1)
    -> Index lookup on inv using fk_inventories_warehouses (warehouse_id=wh.warehouse_id)  (cost=3.62 rows=124) (actual time=0.604..0.787 rows=124 loops=9)
 |
+--------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

2. explain select * from inventories inv, warehouses wh where inv.warehouse_id=wh.warehouse_id;

```
mysql> explain select * from inventories inv, warehouses wh where inv.warehouse_id=wh.warehouse_id;
+----+-------------+-------+------------+------+----------------------------+----------------------------+---------+----------------------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys              | key                        | key_len | ref                  | rows | filtered | Extra |
+----+-------------+-------+------------+------+----------------------------+----------------------------+---------+----------------------+------+----------+-------+
|  1 | SIMPLE      | wh    | NULL       | ALL  | PRIMARY                    | NULL                       | NULL    | NULL                 |    9 |   100.00 | NULL  |
|  1 | SIMPLE      | inv   | NULL       | ref  | fk_inventories_warehouses  | fk_inventories_warehouses  | 2       | sales.wh.warehouse_id|  123 |   100.00 | NULL  |
+----+-------------+-------+------------+------+----------------------------+----------------------------+---------+----------------------+------+----------+-------+
2 rows in set, 1 warning (0.01 sec)
```

3.explain select warehouse_id, count(*) from inventories group by warehouse_id;

```
mysql> explain select warehouse_id, count(*) from inventories group by warehouse_id;
+------------------------------------------------------------------------------------------
| EXPLAIN
+------------------------------------------------------------------------------------------
| -> Group aggregate: count(0)  (cost=223 rows=9)
    -> Covering index scan on inventories using fk_inventories_warehouses  (cost=112 rows=1112)
 |
+------------------------------------------------------------------------------------------
1 row in set (0.01 sec)
```

五、参考资料

1.MySQL 8.4 Reference Manual

https://dev.mysql.com/doc/refman/8.4/en/optimization.html

2. 读懂 MySQL Explain 结果，上亿数据的查询耗时从几分钟降到 63 毫秒

https://www.modb.pro/db/148364

3. MySQL8 查询优化新工具 Explain Analyze

https://www.cnblogs.com/yeyuzhuanjia/p/15660518.html

4.全面解析 MySQL Explain 如何优化 SQL 查询性能

https://www.jb51.net/database/285817cqh.htm

5. mysql 优化之慢查询分析+explain 命令分析+优化技巧总结

https://www.jb51.net/article/275740.htm

6. 手把手教你彻底理解 MySQL 的 explain 关键字

https://cloud.tencent.com/developer/article/1833092

7. 你会看 MySQL 的执行计划（EXPLAIN）吗?

https://blog.csdn.net/liushuijinger/article/details/122876352