《数据库系统及应用实践》课程实验报告

实验 5：存储、索引和基准测试

姓　　名：李鹏达　　学　　号：10225101460　　完成日期：2024 年 5 月 16 日

1　实验目标

1. 学习和理解 MySQL 数据库管理系统中表空间、模式和索引的基本概念和操作；
2. 理解 MySQL 数据库管理系统中数据存储的基本结构；
3. 了解索引对查询性能的影响，能够根据应用场景设计合适的索引；
4. 使用 TPC-H 基准测试对 MySQL 数据库管理系统进行评测；

2　实验过程记录

在 Docker 容器中启动 MySQL 数据库，并连接到数据库 `dbcourse`。

2.1　表空间管理

执行以下 SQL 语句，创建一个名为 `myspace` 的表空间；

```
1 create tablespace myspace add datafile 'myspace.ibd';
```

运行结果如下：

```
mysql> create tablespace myspace add datafile 'myspace.ibd';
Query OK, 0 rows affected (0.03 sec)
```

图 1: 创建表空间

执行以下 SQL 语句，在 `myspace` 表空间中创建数据表 `t1`；

```
1 create table t1(c1 int primary key, c2 varchar(10)) tablespace myspace;
```

运行结果如下：

```
mysql> create table t1(c1 int primary key, c2 varchar(10)) tablespace myspace;
Query OK, 0 rows affected (0.06 sec)
```

图 2: 在表空间中创建数据表

执行以下 SQL 语句，在 System Tablespace, File-Per-Table Tablespace 和 General Tablespace 之间迁移数据表；

```
alter table advisor tablespace myspace;
alter table classroom tablespace innodb_system;
alter table t1 tablespace innodb_file_per_table;
```

运行结果如下：

```
mysql> alter table advisor tablespace myspace;
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table classroom tablespace innodb_system;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table t1 tablespace innodb_file_per_table;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

图 3: 迁移数据表

执行以下 SQL 语句，查询系统中的表空间信息；

```
select space, name, space_type, file_size from information_schema.innodb_tablespaces
    ;
```

运行结果如下：

```
mysql> select space, name, space_type, file_size from information_schema.innodb_tablespaces;
+------------+-------------------------+------------+-----------+
| space      | name                    | space_type | file_size |
+------------+-------------------------+------------+-----------+
| 4294967294 | mysql                   | General    |  32505856 |
| 4294967293 | innodb_temporary        | System     |  12582912 |
| 4294967279 | innodb_undo_001         | Undo       |  16777216 |
| 4294967278 | innodb_undo_002         | Undo       |  16777216 |
|          1 | sys/sys_config          | Single     |    114688 |
|         24 | dbtest/classroom        | Single     |    114688 |
|         25 | dbtest/department       | Single     |    114688 |
|         26 | dbtest/course           | Single     |    131072 |
|         27 | dbtest/instructor       | Single     |    131072 |
|         28 | dbtest/section          | Single     |    131072 |
|         29 | dbtest/teaches          | Single     |    131072 |
|         30 | dbtest/student          | Single     |    131072 |
|         31 | dbtest/takes            | Single     |    131072 |
|         32 | dbtest/advisor          | Single     |    131072 |
|         33 | dbtest/time_slot        | Single     |    114688 |
|         34 | dbtest/prereq           | Single     |    131072 |
|         58 | dbcourse/department     | Single     |    114688 |
|         59 | dbcourse/course         | Single     |    131072 |
|         60 | dbcourse/instructor     | Single     |    131072 |
|         61 | dbcourse/section        | Single     |    131072 |
|         62 | dbcourse/teaches        | Single     |    131072 |
|         63 | dbcourse/student        | Single     |    327680 |
|         64 | dbcourse/takes          | Single     |  12582912 |
|         66 | dbcourse/time_slot      | Single     |    114688 |
|         67 | dbcourse/prereq         | Single     |    131072 |
```

图 4: 查询表空间信息

执行以下 SQL 语句，查询系统中数据表在表空间的分布情况；

```
select ts.name as tablespaces, t.name as tables from information_schema.
    innodb_tablespaces ts, information_schema.innodb_tables t where ts.space = t.
    space;
```

运行结果如下：

```
mysql> select ts.name as tablespaces, t.name as tables from information_schema.innodb_tablespaces ts, informatio
n_schema.innodb_tables t where ts.space = t.space;
+--------------------+-----------------------------------------------------------+
| tablespaces        | tables                                                    |
+--------------------+-----------------------------------------------------------+
| mysql              | mysql/db                                                  |
| mysql              | mysql/user                                                |
| mysql              | mysql/default_roles                                       |
| mysql              | mysql/role_edges                                          |
| mysql              | mysql/global_grants                                       |
| mysql              | mysql/password_history                                    |
| mysql              | mysql/func                                                |
| mysql              | mysql/plugin                                              |
| mysql              | mysql/help_topic                                          |
| mysql              | mysql/help_category                                       |
| mysql              | mysql/help_relation                                       |
| mysql              | mysql/servers                                             |
| mysql              | mysql/tables_priv                                         |
| mysql              | mysql/columns_priv                                        |
| mysql              | mysql/help_keyword                                        |
| mysql              | mysql/time_zone_name                                      |
| mysql              | mysql/time_zone                                           |
| mysql              | mysql/time_zone_transition                                |
| mysql              | mysql/time_zone_transition_type                           |
| mysql              | mysql/time_zone_leap_second                               |
| mysql              | mysql/procs_priv                                          |
| mysql              | mysql/component                                           |
| mysql              | mysql/slave_relay_log_info                                |
| mysql              | mysql/slave_master_info                                   |
| mysql              | mysql/slave_worker_info                                   |
| mysql              | mysql/gtid_executed                                       |
| mysql              | mysql/replication_asynchronous_connection_failover        |
| mysql              | mysql/replication_asynchronous_connection_failover_managed |
| mysql              | mysql/replication_group_member_actions                    |
| mysql              | mysql/replication_group_configuration_version             |
| mysql              | mysql/server_cost                                         |
```

图 5: 查询数据表在表空间的分布情况

执行以下 SQL 语句，将表空间 `myspace` 改名为 `tablespace_460`；

```
alter tablespace myspace rename to tablespace_460;
```

运行结果如下：

```
mysql> alter tablespace myspace rename to tablespace_460;
Query OK, 0 rows affected (0.02 sec)
```

图 6: 将表空间改名

执行以下 SQL 语句，删除表空间 `tablespace_460`；

```
alter table advisor tablespace innodb_system;
drop tablespace tablespace_460;
```

运行结果如下：

```
mysql> alter table advisor tablespace innodb_system;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> drop tablespace tablespace_460;
Query OK, 0 rows affected (0.02 sec)
```

图 7: 删除表空间

2.2 模式

执行下列 SQL 语句，练习在查询中使用模式；

```
select * from movies;
select * from movie.movies;
use movie;
select * from movies;
use dbcourse;
```

运行结果如下：

```
mysql> select * from movies;
ERROR 1146 (42S02): Table 'dbcourse.movies' doesn't exist
mysql> select * from movie.movies;
+-----+--------------------+
| MID | title              |
+-----+--------------------+
|   1 | The Great Dictator |
|   2 | Modern Times       |
|   3 | City Lights        |
+-----+--------------------+
3 rows in set (0.01 sec)

mysql> use movie;
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> select * from movies;
+-----+--------------------+
| MID | title              |
+-----+--------------------+
|   1 | The Great Dictator |
|   2 | Modern Times       |
|   3 | City Lights        |
+-----+--------------------+
3 rows in set (0.00 sec)

mysql> use dbcourse;
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
```

图 8: 使用模式

2.3 索引管理

下载实验所需的数据文件 (*.tbl)，并将其复制到容器中的 /dbcourse 文件夹中；

```
unzip tbl.zip
for file in *.tbl; do
  sudo docker cp "$file" "dbcourse:/dbcourse/$file"
done
```

执行以下 SQL 语句，创建数据表 tpch；

```
CREATE DATABASE tpch CHARACTER SET utf8mb4;
USE tpch;
CREATE TABLE nation(n_nationKEY INTEGER PRIMARY KEY, n_name CHAR(25) NOT NULL,
    n_regionkey INTEGER NOT NULL, n_comment VARCHAR(152));
CREATE TABLE region(r_regionkey INTEGER PRIMARY KEY, r_name CHAR(25) NOT NULL,
    r_comment VARCHAR(152));
CREATE TABLE part(p_partkey INTEGER PRIMARY KEY, p_name VARCHAR(55) NOT NULL, p_mfgr
     CHAR(25) NOT NULL, p_brand CHAR(10) NOT NULL, p_type VARCHAR(25) NOT NULL,
    p_size INTEGER NOT NULL, p_container CHAR(10) NOT NULL, p_retailprice DECIMAL
    (15,2) NOT NULL, p_comment VARCHAR(23) NOT NULL);
CREATE TABLE supplier(s_suppkey INTEGER PRIMARY KEY, s_name CHAR(25) NOT NULL,
    s_address VARCHAR(40) NOT NULL, s_nationkey INTEGER NOT NULL, s_phone CHAR(15)
    NOT NULL, s_acctbal DECIMAL(15,2) NOT NULL, s_comment VARCHAR(101) NOT NULL);
CREATE TABLE partsupp(ps_partkey INTEGER NOT NULL, ps_suppkey INTEGER NOT NULL,
    ps_availqty INTEGER NOT NULL, ps_supplycost DECIMAL(15,2) NOT NULL, ps_comment
    VARCHAR(199) NOT NULL, PRIMARY KEY (ps_partkey, ps_suppkey) );
CREATE TABLE customer(c_custkey INTEGER PRIMARY KEY, c_name VARCHAR(25) NOT NULL,
    c_address VARCHAR(40) NOT NULL, c_nationkey INTEGER NOT NULL, c_phone CHAR(15)
    NOT NULL, c_acctbal DECIMAL(15,2) NOT NULL, c_mktsegment CHAR(10) NOT NULL,
    c_comment VARCHAR(117) NOT NULL);
CREATE TABLE orders(o_orderkey INTEGER PRIMARY KEY, o_custkey INTEGER NOT NULL,
    o_orderstatus CHAR(1) NOT NULL, o_totalprice DECIMAL(15,2) NOT NULL, o_orderdate
     date NOT NULL, o_orderpriority CHAR(15) NOT NULL, o_clerk CHAR(15) NOT NULL,
    o_shippriority INTEGER NOT NULL, o_comment VARCHAR(79) NOT NULL);
CREATE TABLE lineitem(l_orderkey INTEGER NOT NULL, l_partkey INTEGER NOT NULL,
    l_suppkey INTEGER NOT NULL, l_linenumber INTEGER NOT NULL, l_quantity DECIMAL
    (15,2) NOT NULL, l_extendedprice DECIMAL(15,2) NOT NULL, l_discount DECIMAL
    (15,2) NOT NULL, l_tax DECIMAL(15,2) NOT NULL, l_returnflag CHAR(1) NOT NULL,
    l_linestatus CHAR(1) NOT NULL, l_shipdate date NOT NULL, l_commitdate date NOT
    NULL, l_receiptdate date NOT NULL, l_shipinstruct CHAR(25) NOT NULL, l_shipmode
    CHAR(10) NOT NULL, l_comment VARCHAR(44) NOT NULL, PRIMARY KEY(l_orderkey,
    l_linenumber));
```

部分运行结果如下：

```
mysql> CREATE TABLE partsupp(ps_partkey INTEGER NOT NULL, ps_suppkey INTEGER NOT NULL, ps_availqty INTEGER NOT N
ULL, ps_supplycost DECIMAL(15,2) NOT NULL, ps_comment VARCHAR(199) NOT NULL, PRIMARY KEY (ps_partkey, ps_suppkey
) );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE customer(c_custkey INTEGER PRIMARY KEY, c_name VARCHAR(25) NOT NULL, c_address VARCHAR(40) N
OT NULL, c_nationkey INTEGER NOT NULL, c_phone CHAR(15) NOT NULL, c_acctbal DECIMAL(15,2) NOT NULL, c_mktsegment
 CHAR(10) NOT NULL, c_comment VARCHAR(117) NOT NULL);
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE orders(o_orderkey INTEGER PRIMARY KEY, o_custkey INTEGER NOT NULL, o_orderstatus CHAR(1) NOT
 NULL, o_totalprice DECIMAL(15,2) NOT NULL, o_orderdate date NOT NULL, o_orderpriority CHAR(15) NOT NULL, o_cler
k CHAR(15) NOT NULL, o_shippriority INTEGER NOT NULL, o_comment VARCHAR(79) NOT NULL);
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE lineitem(l_orderkey INTEGER NOT NULL, l_partkey INTEGER NOT NULL, l_suppkey INTEGER NOT NULL
, l_linenumber INTEGER NOT NULL, l_quantity DECIMAL(15,2) NOT NULL, l_extendedprice DECIMAL(15,2) NOT NULL, l_di
scount DECIMAL(15,2) NOT NULL, l_tax DECIMAL(15,2) NOT NULL, l_returnflag CHAR(1) NOT NULL, l_linestatus CHAR(1)
 NOT NULL, l_shipdate date NOT NULL, l_commitdate date NOT NULL, l_receiptdate date NOT NULL, l_shipinstruct CHA
R(25) NOT NULL, l_shipmode CHAR(10) NOT NULL, l_comment VARCHAR(44) NOT NULL, PRIMARY KEY(l_orderkey,l_linenumbe
r));
Query OK, 0 rows affected (0.03 sec)
```

图 9: 创建数据表

执行下列命令/语句，修改系统配置，允许从 MySQL 客户端加载数据；

```
1 show global variables like 'local_infile';
2 set global local_infile = on;
3 show global variables like 'local_infile';
4 quit
5 mysql --local-infile=1 -uroot -p -Dtpch
```

```
mysql> show global variables like 'local_infile';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| local_infile  | OFF   |
+---------------+-------+
1 row in set (0.06 sec)

mysql> set global local_infile = on;
Query OK, 0 rows affected (0.01 sec)

mysql> show global variables like 'local_infile';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| local_infile  | ON    |
+---------------+-------+
1 row in set (0.00 sec)
```

图 10: 修改系统配置

执行下列命令，从客户端本地数据文件中导入数据；

```
load data local infile '/dbcourse/nation.tbl' into table nation fields terminated by '|' lines terminated by '\n';
load data local infile '/dbcourse/region.tbl' into table region fields terminated by '|' lines terminated by '\n';
load data local infile '/dbcourse/part.tbl' into table part fields terminated by '|' lines terminated by '\n';
load data local infile '/dbcourse/supplier.tbl' into table supplier fields terminated by '|' lines terminated by '\n';
load data local infile '/dbcourse/partsupp.tbl' into table partsupp fields terminated by '|' lines terminated by '\n';
load data local infile '/dbcourse/customer.tbl' into table customer fields terminated by '|' lines terminated by '\n';
load data local infile '/dbcourse/orders.tbl' into table orders fields terminated by '|' lines terminated by '\n';
load data local infile '/dbcourse/lineitem.tbl' into table lineitem fields terminated by '|' lines terminated by '\n';
```

部分运行结果如下：

```
mysql> load data local infile '/dbcourse/customer.tbl' into table customer fields terminated by '|' lines termin
ated by '\n';
Query OK, 30000 rows affected (0.25 sec)
Records: 30000  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data local infile '/dbcourse/orders.tbl' into table orders fields terminated by '|' lines terminated
 by '\n';
Query OK, 300000 rows affected (1.49 sec)
Records: 300000  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data local infile '/dbcourse/lineitem.tbl' into table lineitem fields terminated by '|' lines termin
ated by '\n';
Query OK, 1199969 rows affected (8.59 sec)
Records: 1199969  Deleted: 0  Skipped: 0  Warnings: 0
```

图 11: 导入数据

修改会话变量 `profiling`，允许统计当前会话中每条语句的资源消耗情况；

```
show variables like 'profiling';
set profiling = on;
show variables like 'profiling';
```

如图所示：

```
mysql> show variables like 'profiling';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| profiling     | OFF   |
+---------------+-------+
1 row in set (0.00 sec)

mysql> set profiling = on;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> show variables like 'profiling';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| profiling     | ON    |
+---------------+-------+
1 row in set (0.00 sec)
```

图 12: 修改会话变量

执行下列语句，对比两个查询的资源消耗情况，记录并分析相关数据；

```sql
select * from lineitem;
show profile;
select l_orderkey, l_partkey, l_suppkey, l_linenumber, l_quantity,l_extendedprice,
    l_discount, l_tax, l_returnflag, l_linestatus, l_shipdate, l_commitdate,
    l_receiptdate, l_shipinstruct, l_shipmode, l_comment from lineitem;
show profile;
show profiles;
```

```
1199969 rows in set (1.38 sec)

mysql> show profile;
+--------------------------------+----------+
| Status                         | Duration |
+--------------------------------+----------+
| starting                       | 0.000192 |
| Executing hook on transaction  | 0.000025 |
| starting                       | 0.000010 |
| checking permissions           | 0.000007 |
| Opening tables                 | 0.000059 |
| init                           | 0.000008 |
| System lock                    | 0.000025 |
| optimizing                     | 0.000004 |
| statistics                     | 0.000059 |
| preparing                      | 0.000023 |
| executing                      | 1.381169 |
| end                            | 0.000016 |
| query end                      | 0.000005 |
| waiting for handler commit     | 0.000033 |
| closing tables                 | 0.000012 |
| freeing items                  | 0.000023 |
| cleaning up                    | 0.000037 |
+--------------------------------+----------+
17 rows in set, 1 warning (0.02 sec)
```

(a) 查询 1

```
g frays. carefully even          |
1199969 rows in set (1.33 sec)

mysql> show profile;
+--------------------------------+----------+
| Status                         | Duration |
+--------------------------------+----------+
| starting                       | 0.000161 |
| Executing hook on transaction  | 0.000004 |
| starting                       | 0.000021 |
| checking permissions           | 0.000011 |
| Opening tables                 | 0.000090 |
| init                           | 0.000006 |
| System lock                    | 0.000010 |
| optimizing                     | 0.000004 |
| statistics                     | 0.000038 |
| preparing                      | 0.000018 |
| executing                      | 1.325929 |
| end                            | 0.000022 |
| query end                      | 0.000006 |
| waiting for handler commit     | 0.000013 |
| closing tables                 | 0.000013 |
| freeing items                  | 0.000032 |
| cleaning up                    | 0.000574 |
+--------------------------------+----------+
17 rows in set, 1 warning (0.00 sec)
```

(b) 查询 2

图 13: 查询资源消耗情况

```
mysql> show profiles;
+----------+------------+---------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------+
| Query_ID | Duration   | Query
                                                                                                |
+----------+------------+---------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------+
|        1 | 0.00115175 | show variables like 'profiling'
                                                                                                |
|        2 | 0.00110775 | show variables like 'profiling'
                                                                                                |
|        3 | 1.38170525 | select * from lineitem
                                                                                                |
|        4 | 1.32694875 | select l_orderkey,l_partkey,l_suppkey, l_linenumber, l_quantity,l_extendedprice, l_discount, l_tax, l_returnf
lag, l_linestatus, l_shipdate, l_commitdate, l_receiptdate, l_shipinstruct, l_shipmode, l_comment from lineitem |
+----------+------------+---------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------+
4 rows in set, 1 warning (0.00 sec)
```

图 14: 查询资源消耗情况比较

可以看到，查询 2 的资源消耗情况与查询 1 几乎相同，这是因为查询 2 和查询 1 均查询了整个数据表中的所有列，因此资源消耗情况相差不大。

执行下列语句，观察主键查询的性能，记录并分析相关数据；

```sql
select * from lineitem where l_orderkey = 354 and l_linenumber = 7;
select * from lineitem where l_orderkey = 354;
select * from lineitem where l_linenumber = 7;
show profiles;
```

结果如下：

```
mysql> show profiles;
+----------+------------+-------------------------------------------------------------------+
| Query_ID | Duration   | Query                                                             |
+----------+------------+-------------------------------------------------------------------+
|       20 | 0.01337175 | select * from lineitem where l_orderkey = 354 and l_linenumber = 7 |
|       21 | 0.00021800 | select * from lineitem where l_orderkey = 354                     |
|       22 | 0.78383100 | select * from lineitem where l_linenumber = 7                     |
+----------+------------+-------------------------------------------------------------------+
3 rows in set, 1 warning (0.01 sec)
```

图 15: 主键查询性能

根据分析，查询 1 和查询 2 的执行时间非常短，表明它们能够有效利用复合主键索引。而查询 3 的执行时间较长，因为它使用的是复合索引的第二项，不能利用主键索引，可能需要全表扫描。

执行下列语句，观察索引对查询性能的影响，记录并分析相关数据；

```sql
select * from lineitem where l_partkey = 860;
create index lineitem_partkey on lineitem(l_partkey);
select * from lineitem where l_partkey = 860;
show profiles;
drop index lineitem_partkey on lineitem;
```

结果如下：

```
mysql> show profiles;
+----------+-------------+-----------------------------------------------------+
| Query_ID | Duration    | Query                                               |
+----------+-------------+-----------------------------------------------------+
|       23 | 1.10674600 | select * from lineitem where l_partkey = 860         |
|       24 | 2.80146250 | create index lineitem_partkey on lineitem(l_partkey) |
|       25 | 0.01401525 | select * from lineitem where l_partkey = 860         |
+----------+-------------+-----------------------------------------------------+
3 rows in set, 1 warning (0.00 sec)
```

图 16: 索引对查询性能的影响

根据分析，创建索引后，查询的执行时间明显减少，表明索引对查询性能有显著的提升。

执行下列语句，观察唯一索引、前缀索引对查询性能的影响，记录并分析相关数据；

```
select * from customer where c_address = 'KvpyuHCplrB84WgAiGV6sYpZq7Tj';
select * from customer where c_address like 'KvpyuH%';
create index customer_address on customer(c_address);
select * from customer where c_address = 'KvpyuHCplrB84WgAiGV6sYpZq7Tj';
select * from customer where c_address like 'KvpyuH%';
drop index customer_address on customer;
create unique index customer_address on customer(c_address);
select * from customer where c_address = 'KvpyuHCplrB84WgAiGV6sYpZq7Tj';
select * from customer where c_address like 'KvpyuH%';
drop index customer_address on customer;
create index customer_address on customer(c_address(4));
select * from customer where c_address = 'KvpyuHCplrB84WgAiGV6sYpZq7Tj';
select * from customer where c_address like 'KvpyuH%';
show profiles;
drop index customer_address on customer;
```

结果如下：

```
mysql> show profiles;
+----------+-------------+-----------------------------------------------------------------------+
| Query_ID | Duration    | Query                                                                 |
+----------+-------------+-----------------------------------------------------------------------+
|       27 | 0.03713300 | select * from customer where c_address = 'KvpyuHCplrB84WgAiGV6sYpZq7Tj' |
|       28 | 0.02250750 | select * from customer where c_address like 'KvpyuH%'                 |
|       29 | 0.17657125 | create index customer_address on customer(c_address)                  |
|       30 | 0.00071875 | select * from customer where c_address = 'KvpyuHCplrB84WgAiGV6sYpZq7Tj' |
|       31 | 0.01313100 | select * from customer where c_address like 'KvpyuH%'                 |
|       32 | 0.01276500 | drop index customer_address on customer                               |
|       33 | 0.15576000 | create unique index customer_address on customer(c_address)           |
|       34 | 0.00053675 | select * from customer where c_address = 'KvpyuHCplrB84WgAiGV6sYpZq7Tj' |
|       35 | 0.00111400 | select * from customer where c_address like 'KvpyuH%'                 |
|       36 | 0.01390800 | drop index customer_address on customer                               |
|       37 | 0.09298900 | create index customer_address on customer(c_address(4))               |
|       38 | 0.00055850 | select * from customer where c_address = 'KvpyuHCplrB84WgAiGV6sYpZq7Tj' |
|       39 | 0.00056475 | select * from customer where c_address like 'KvpyuH%'                 |
+----------+-------------+-----------------------------------------------------------------------+
13 rows in set, 1 warning (0.00 sec)
```

图 17: 唯一索引、前缀索引对查询性能的影响

从查询结果和执行时间分析来看，创建索引后，查询性能显著提升，尤其是在精确匹配查询中表现突出。而对于 LIKE 查询，前缀索引的性能更优，因为在这个场景下，LIKE 查询匹配的区域恰好是前缀索引的索引区域。

执行下列语句，观察索引对于更新性能的影响，记录并分析相关数据；

```
1  insert into customer values(30001,'Customer#000030001','314hjafd2354',8,'
       27-147-574-1234',888.88,'AUTOMOBILE','This is a test!');
2  update customer set c_address = 'kqe34gqerl' where c_custkey = 30001;
3  delete from customer where c_custkey = 30001;
4  show profiles;
5  create index customer_2 on customer(c_name);
6  create index customer_3 on customer(c_address);
7  create index customer_4 on customer(c_nationkey);
8  create index customer_5 on customer(c_phone);
9  create index customer_6 on customer(c_acctbal);
10 create index customer_7 on customer(c_mktsegment);
11 create index customer_8 on customer(c_comment);
12 create index customer_23 on customer(c_name,c_address);
13 create index customer_24 on customer(c_name,c_nationkey);
14 create index customer_25 on customer(c_name,c_phone);
15 create index customer_26 on customer(c_name,c_acctbal);
16 create index customer_27 on customer(c_name,c_mktsegment);
17 create index customer_28 on customer(c_name,c_comment);
18 insert into customer values(30001,'Customer#000030001','314hjafd2354',8,'
       27-147-574-1234',888.88,'AUTOMOBILE','This is a test!');
19 update customer set c_address = 'kqe34gqerl' where c_custkey = 30001;
20 delete from customer where c_custkey = 30001;
21 show profiles;
22 drop index customer_2 on customer;
```

```sql
drop index customer_3 on customer;
drop index customer_4 on customer;
drop index customer_5 on customer;
drop index customer_6 on customer;
drop index customer_7 on customer;
drop index customer_8 on customer;
drop index customer_23 on customer;
drop index customer_24 on customer;
drop index customer_25 on customer;
drop index customer_26 on customer;
drop index customer_27 on customer;
drop index customer_28 on customer;
```

结果如下：

```
mysql> show profiles;
+----------+------------+-----------------------------------------------------------------------------------------------------------+
| Query_ID | Duration   | Query                                                                                                     |
+----------+------------+-----------------------------------------------------------------------------------------------------------+
|      104 | 0.00481400 | insert into customer values(30001,'Customer#000030001','314hjafd2354',8,'27-147-574-1234',888.88,'AUTOMOBILE','This is a test!') |
|      105 | 0.00347075 | update customer set c_address = 'kqe34gqerl' where c_custkey = 30001                                      |
|      106 | 0.00229025 | delete from customer where c_custkey = 30001                                                              |
|      107 | 0.13165200 | create index customer_2 on customer(c_name)                                                               |
|      108 | 0.12281475 | create index customer_3 on customer(c_address)                                                            |
|      109 | 0.07368475 | create index customer_4 on customer(c_nationkey)                                                          |
|      110 | 0.11840175 | create index customer_5 on customer(c_phone)                                                              |
|      111 | 0.07915225 | create index customer_6 on customer(c_acctbal)                                                            |
|      112 | 0.10151975 | create index customer_7 on customer(c_mktsegment)                                                         |
|      113 | 0.25970600 | create index customer_8 on customer(c_comment)                                                            |
|      114 | 0.19555075 | create index customer_23 on customer(c_name,c_address)                                                    |
|      115 | 0.13767925 | create index customer_24 on customer(c_name,c_nationkey)                                                  |
|      116 | 0.16017825 | create index customer_25 on customer(c_name,c_phone)                                                      |
|      117 | 0.14033450 | create index customer_26 on customer(c_name,c_acctbal)                                                    |
|      118 | 0.14793050 | create index customer_27 on customer(c_name,c_mktsegment)                                                 |
|      119 | 0.29532750 | create index customer_28 on customer(c_name,c_comment)                                                    |
|      120 | 0.00384775 | insert into customer values(30001,'Customer#000030001','314hjafd2354',8,'27-147-574-1234',888.88,'AUTOMOBILE','This is a test!') |
|      121 | 0.00341350 | update customer set c_address = 'kqe34gqerl' where c_custkey = 30001                                      |
|      122 | 0.00278675 | delete from customer where c_custkey = 30001                                                              |
+----------+------------+-----------------------------------------------------------------------------------------------------------+
19 rows in set, 1 warning (0.00 sec)
```

图 18: 索引对于更新性能的影响

可以看到，索引对于更新性能的影响并不明显，因为更新操作主要是对数据表的修改，而索引主要影响查询操作的性能。

2.4 TPC-H 基准测试

2.4.1 测试环境

- OS: Ubuntu 24.04 LTS on Windows 10 x86_64
- Kernel: 5.15.146.1-microsoft-standard-WSL2
- CPU: 11th Gen Intel i7-11800H (16) @ 2.304GHz
- Memory: 7832MiB
- MySQL: 8.2.0
- PostgreSQL: 16.3 (Debian 16.3-1.pgdg120+1)

2.4.2 数据集

- 数据规模：220MB
- 索引：默认

2.4.3 测试方法

先进行 2 次预热后，进行 5 次测试，取平均值作为最终结果。这样可以尽可能避免冷启动与数据抖动对测试结果的影响。

2.4.4 测试过程

依次执行 TPC-H 的 22 个查询，记录每个查询的执行时间，并计算总用时。在测试过程中，使用 `nmon` 工具查看 CPU 和内存的使用情况并记录。

2.4.5 测试结果

查询用时如下表所示：

表 1: 查询用时

测试查询	**MySQL** 执行时间（s）	**PostgreSQL** 执行时间（s）
Q1	2.2788	0.5588
Q2	0.0966	0.0964
Q3	0.4922	0.2320
Q4	0.1644	0.0500
Q5	0.2599	0.0562
Q6	0.3825	0.0722
Q7	0.8277	0.0666
Q8	0.5842	0.0688
Q9	3.5387	0.1872
Q10	0.3859	0.1118

接下页

表 1: （续）

测试查询	**MySQL** 执行时间（s）	**PostgreSQL** 执行时间（s）
Q11	0.3572	0.0452
Q12	0.5793	0.0946
Q13	0.3725	0.1356
Q14	0.4602	0.0700
Q15	0.4225	0.0698
Q16	0.0681	0.0834
Q17	313.2492	75.3384
Q18	0.5279	0.4054
Q19	0.8516	0.0972
Q20	475.0988	118.2270
Q21	1.0844	0.1034
Q22	0.1727	0.0488
Total	802.2555	196.2188

CPU 和内存使用情况如下表所示：

表 2: 资源使用情况

资源	**MySQL**	**PostgreSQL**
CPU	117%	7.17%
内存	96.3%	3.8%

根据测试结果中的查询用时，以查询为横坐标，执行时间的对数为纵坐标，绘制 MySQL 和 PostgreSQL 的查询用时对比图如下：

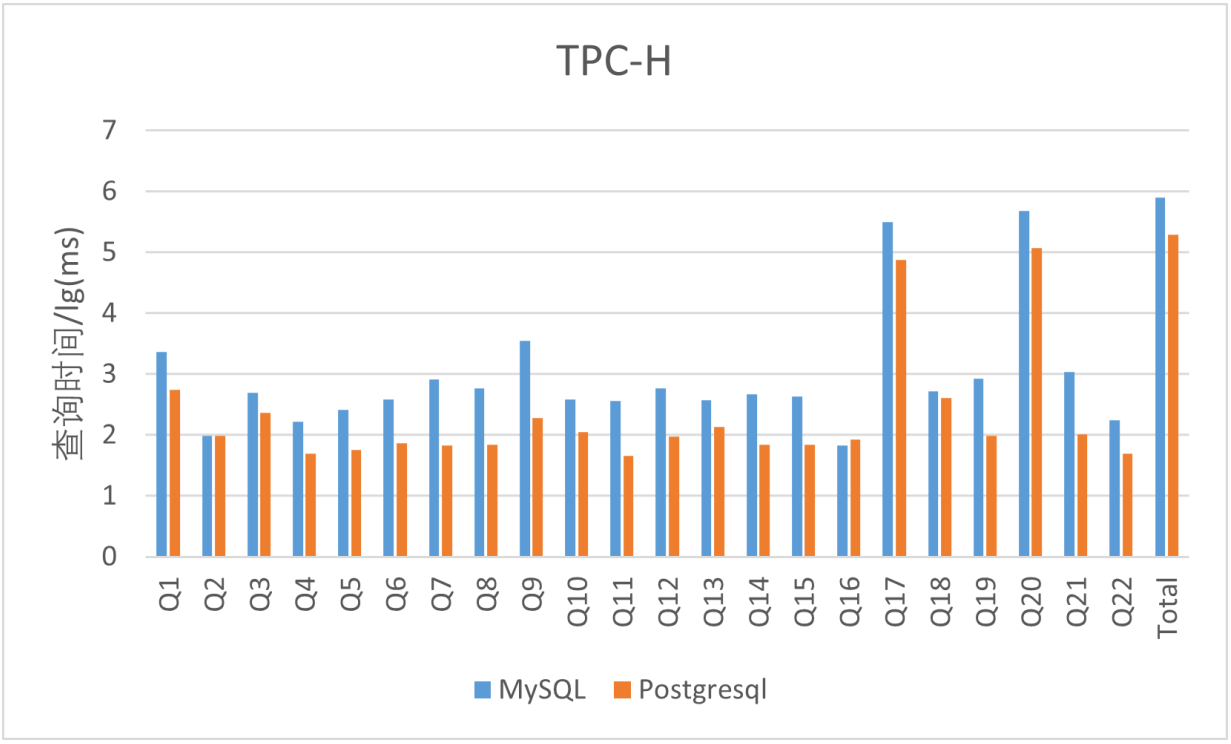

图 19: 查询用时对比

2.4.6 测试结论

根据测试结果，可以得出以下结论：

1. 在 TPC-H 基准测试的查询中，PostgreSQL 的执行时间明显优于 MySQL，表明 PostgreSQL 在执行 TPC-H 基准测试时的性能更好；
2. PostgreSQL 的 CPU 和内存使用率明显低于 MySQL，表明 PostgreSQL 在执行 TPC-H 基准测试时的资源消耗更低。

2.4.7 测试分析

PostgreSQL 在执行 TPC-H 基准测试时的性能更好，这可能是由于 TPC-H 基准测试中的查询都比较复杂，PostgreSQL 的查询优化器能够更好地优化查询计划，提高查询性能。

3 存在的问题及解决方案

1. 在使用 `drop tablespace` 删除表空间时，提示 `ERROR 3120 (HY000): Tablespace` `` `tablespace_460` `` `is not empty`，无法删除表空间；

   这是因为表空间中还存在数据表，需要先将数据表迁移到其他表空间，再删除表空间。

2. 在分析查询资源消耗情况时，以前的查询结果会一直留在 `profiles` 表中；

   可以使用以下 SQL 语句清空 `profiles` 表中的数据：

```
SET @@profiling = 0;
SET @@profiling_history_size = 0;
SET @@profiling_history_size = 100;
SET @@profiling = 1;
```

3. 在执行 TPC-H 基准测试时，使用了其生成的 1GB 数据集，但在这种数据量下，部分查询需要超过 1 小时才能完成。因此，为了节省时间，使用了大夏学堂提供的 220MB 的数据集进行测试。

4 实验小结

通过本次实验，我学习了 MySQL 数据库管理系统中表空间、模式和索引的基本概念和操作，了解了 MySQL 数据库管理系统中数据存储的基本结构，掌握了索引对查询性能的影响，能够根据应用场景设计合适的索引，同时也学习了使用 TPC-H 基准测试对数据库管理系统进行评测。通过实验，我对数据库系统的存储、索引和基准测试有了更深入的理解，为今后的数据库系统设计和应用打下了基础。