Project Stage 3: Database Implementation and Indexing

Team name- SRKC

Database Connection

```
[2022/10/21 23:21:45] → saket@MBA → ~
$ mysql -u root -proottoor
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, 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 databases; use SRKC;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| SRKC               |
| sys                |
+--------------------+
5 rows in set (0.01 sec)

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;
+------------------+
| Tables_in_srkc   |
+------------------+
| AvailableSlots   |
| EquipmentRentals |
| Equipments       |
| EventBookings    |
| Events           |
| Facilities       |
| Roles            |
| SlotBookings     |
| Slots            |
| Sports           |
| Users            |
+------------------+
11 rows in set (0.00 sec)

mysql>
```

Row count in tables

```
[2022/10/21 23:27:41] → saket@MBA → ~
$ mysql -u root -proottoor
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, 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 SRKC;
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;
+------------------+
| Tables_in_srkc   |
+------------------+
| AvailableSlots   |
| EquipmentRentals |
| Equipments       |
| EventBookings    |
| Events           |
| Facilities       |
| Roles            |
| SlotBookings     |
| Slots            |
| Sports           |
| Users            |
+------------------+
11 rows in set (0.00 sec)

mysql> select count(*) from EquipmentRentals;
+----------+
| count(*) |
+----------+
|     2317 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from SlotBookings;
+----------+
| count(*) |
+----------+
|    22000 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from AvailableSlots;
+----------+
| count(*) |
+----------+
|     4752 |
+----------+
1 row in set (0.00 sec)

mysql>
```

Advanced Queries:

Query to find out how much time does a User spend playing a Basketball from 22nd August 2022 to 22nd December 2022 based on the slot booked (considering slot in the future)

⇒ Screenshot of query result with top 15 rows

```
mysql> SELECT sp.SPORT_NAME, s.NET_ID, COUNT(SLOT_ID) as TOTAL_HOURS_SPENT
    -> FROM slotbookings s JOIN Facilities f USING(FACILITY_ID) JOIN sports sp USING(SPORT_ID)
    -> WHERE s.BOOKING_DATE BETWEEN '2022-08-22' AND '2022-12-22' AND sp.sport_id=2
    -> GROUP BY sp.SPORT_NAME, s.NET_ID limit 15;
+------------+-----------------+-------------------+
| SPORT_NAME | NET_ID          | TOTAL_HOURS_SPENT |
+------------+-----------------+-------------------+
| Basketball | _greenwings     |                13 |
| Basketball | _johnny215      |                16 |
| Basketball | 123infographics |                12 |
| Basketball | adriankao       |                 7 |
| Basketball | afrikaaman      |                16 |
| Basketball | ajlisteningpost |                10 |
| Basketball | alayna_powers   |                10 |
| Basketball | anonymtipster   |                15 |
| Basketball | antlerbunny     |                22 |
| Basketball | antwiivirus     |                21 |
| Basketball | b0_s8_hx        |                15 |
| Basketball | bbingl          |                14 |
| Basketball | bcakoparati208  |                12 |
| Basketball | blancamiosi     |                18 |
| Basketball | blemmie         |                10 |
+------------+-----------------+-------------------+
15 rows in set (0.01 sec)
```

Indexing Design 1

⇒ Index created on *net_id* in the **SlotBookings** table

Indexing Design 2

⇒ Index created on *booking_date* in the **SlotBookings** table

Indexing Design 3

⇒ Index created on *slot_id* and *booking_date* in the **SlotBookings** table

Reason

⇒ Our first query uses *net_id* present in the **SlotBookings** table to GROUP BY such that we get how much time a User spends playing BasketBall.

⇒ By default, MySQL does not index Foreign Key column. So, to check whether this improves the performance for query 1, we create an index on *net_id* in **SlotBookings** which is a Foreign Key reference to **Users (***net_id***)** column.

⇒ Index design 2 which creates index on *booking_date* to see if this will help us retrieve data faster.

⇒ For Index design 3, we use a combination of *slot_id* and *booking_date*. We test that if query performance improves when we search for the slots in between two dates.

Explain Analyze of query 1 before indexing

```
Database changed
mysql> explain analyze (SELECT sp.SPORT_NAME, s.NET_ID, COUNT(SLOT_ID) as TOTAL_HOURS_SPENT
    -> FROM slotbookings s JOIN Facilities f USING(FACILITY_ID) JOIN sports sp USING(SPORT_ID)
    -> WHERE s.BOOKING_DATE BETWEEN '2022-08-22' AND '2022-12-22' AND sp.sport_id=2
    -> GROUP BY sp.SPORT_NAME, s.NET_ID) ;
+--------------------------------------------------------------------------------
| EXPLAIN
                                                                                  |
+--------------------------------------------------------------------------------
| -> Table scan on <temporary>  (actual time=7.964..8.009 rows=200 loops=1)
    -> Aggregate using temporary table  (actual time=7.960..7.960 rows=200 loops=1)
        -> Nested loop inner join  (cost=334.68 rows=351) (actual time=0.128..4.927 rows=2672 loops=1)
            -> Covering index lookup on f using sport_id (sport_id=2)  (cost=1.85 rows=16) (actual time=0.029..0.056 rows=16 loops=1)
            -> Filter: (s.booking_date between '2022-08-22' and '2022-12-22')  (cost=1.22 rows=22) (actual time=0.060..0.283 rows=167 loops=16)
                -> Covering index lookup on s using facility_id (facility_id=f.facility_id)  (cost=1.22 rows=197) (actual time=0.058..0.162 rows=180 loops=16)
 |
+--------------------------------------------------------------------------------
1 row in set (0.01 sec)
```

After creating index net_id

```
mysql> create index sp1 on SlotBookings(net_id);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze (SELECT sp.SPORT_NAME, s.NET_ID, COUNT(SLOT_ID) as TOTAL_HOURS_SPENT
    -> FROM slotbookings s JOIN Facilities f USING(FACILITY_ID) JOIN sports sp USING(SPORT_ID)
    -> WHERE s.BOOKING_DATE = '2022-08-22' AND sp.sport_id=2
    -> GROUP BY sp.SPORT_NAME, s.NET_ID);
+--------------------------------------------------------------------------------
| EXPLAIN
                                                                                  |
+--------------------------------------------------------------------------------
| -> Table scan on <temporary>  (actual time=0.591..0.592 rows=1 loops=1)
    -> Aggregate using temporary table  (actual time=0.587..0.587 rows=1 loops=1)
        -> Nested loop inner join  (cost=43.28 rows=29) (actual time=0.263..0.532 rows=1 loops=1)
            -> Covering index lookup on f using sport_id (sport_id=2)  (cost=1.85 rows=16) (actual time=0.021..0.028 rows=16 loops=1)
            -> Filter: (s.booking_date = TIMESTAMP'2022-08-22 00:00:00')  (cost=0.81 rows=2) (actual time=0.030..0.031 rows=0 loops=16)
                -> Covering index lookup on s using facility_id (facility_id=f.facility_id)  (cost=0.81 rows=18) (actual time=0.018..0.028 rows=19 loops=16)
 |
+--------------------------------------------------------------------------------
1 row in set (0.01 sec)
```

After creating index booking_date

```
mysql> create index sp1 on SlotBookings(booking_date);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze (SELECT sp.SPORT_NAME, s.NET_ID, COUNT(SLOT_ID) as TOTAL_HOURS_SPENT
    -> FROM slotbookings s JOIN Facilities f USING(FACILITY_ID) JOIN sports sp USING(SPORT_ID)
    -> WHERE s.BOOKING_DATE = '2022-08-22' AND sp.sport_id=2
    -> GROUP BY sp.SPORT_NAME, s.NET_ID);
+------------------------------------------------------------------------------------------------
| EXPLAIN

+------------------------------------------------------------------------------------------------
| -> Group aggregate: count(s.slot_id)  (cost=13.08 rows=1) (actual time=0.129..0.129 rows=1 loops=1)
    -> Nested loop inner join  (cost=12.94 rows=1) (actual time=0.098..0.119 rows=1 loops=1)
        -> Covering index lookup on s using sp1 (booking_date=TIMESTAMP'2022-08-22 00:00:00')  (cost=1.94 rows=10) (actual time=0.019..0.026 rows=10 loops=1)
        -> Filter: (f.sport_id = 2)  (cost=1.00 rows=0.1) (actual time=0.009..0.009 rows=0 loops=10)
            -> Single-row index lookup on f using PRIMARY (facility_id=s.facility_id)  (cost=1.00 rows=1) (actual time=0.008..0.008 rows=1 loops=10)
 |
+------------------------------------------------------------------------------------------------
1 row in set (0.00 sec)
```

After creating index slot_id and booking_date

```
mysql> create index sp1 on SlotBookings(slot_id, booking_date);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze (SELECT sp.SPORT_NAME, s.NET_ID, COUNT(SLOT_ID) as TOTAL_HOURS_SPENT
    -> FROM slotbookings s JOIN Facilities f USING(FACILITY_ID) JOIN sports sp USING(SPORT_ID)
    -> WHERE s.BOOKING_DATE = '2022-08-22' AND sp.sport_id=2
    -> GROUP BY sp.SPORT_NAME, s.NET_ID);
+------------------------------------------------------------------------------------------------
| EXPLAIN
                                                                            |
+------------------------------------------------------------------------------------------------
| -> Table scan on <temporary>  (actual time=0.410..0.411 rows=1 loops=1)
    -> Aggregate using temporary table  (actual time=0.407..0.407 rows=1 loops=1)
        -> Nested loop inner join  (cost=43.28 rows=29) (actual time=0.184..0.370 rows=1 loops=1)
            -> Covering index lookup on f using sport_id (sport_id=2)  (cost=1.85 rows=16) (actual time=0.017..0.022 rows=16 loops=1)
            -> Filter: (s.booking_date = TIMESTAMP'2022-08-22 00:00:00')  (cost=0.81 rows=2) (actual time=0.021..0.021 rows=0 loops=16)
                -> Covering index lookup on s using facility_id (facility_id=f.facility_id)  (cost=0.81 rows=18) (actual time=0.012..0.019 rows=19 loops=16)
 |
+------------------------------------------------------------------------------------------------
1 row in set (0.00 sec)
```

Observation

- ⇒ After creating an index on *net_id* we can see that the time taken to lookup improves from 0.058 sec to 0.018 sec. This may be because the index is helping speed up the GROUP BY aggregation.
- ⇒ We need more impressive improvements when index is created on *booking_date*. The time for executing the WHERE clause drops from 0.06 sec to 0.019 sec. It is seen in the above screenshot that the lookup is done on the index *sp1*. Thus, this index helps the query to quickly grab the dates specified in the WHERE clause.

⇒ Index on *slot_id* and *booking_date* also helps to improve the performance as the time drops from 0.058 sec to 0.012 sec as this index helps to speed up the SELECT query with the WHERE clause.

Query to identify top 15 events that bring the most revenue

⇒ Screenshot of query result with top 15 rows

```
mysql>
mysql> SELECT e.event_name, SUM(eb.ticket_count * e.ticket_cost) as total_val
    -> FROM eventbookings eb JOIN events e USING(EVENT_ID)
    -> GROUP BY e.EVENT_ID
    -> ORDER BY total_val desc limit 15;
+-----------------------------------+-----------+
| event_name                        | total_val |
+-----------------------------------+-----------+
| Volleyball Competition Fall 2022  |      9660 |
| Volleyball Competition Fall 2022  |      8835 |
| Volleyball Competition Fall 2022  |      8700 |
| Volleyball Competition Fall 2022  |      8520 |
| Volleyball Competition Fall 2022  |      7785 |
| Volleyball Competition Fall 2022  |      6900 |
| Welcome Parade 2022               |      4780 |
| Swimming Fall 2022 Contest        |      4488 |
| Swimming Fall 2022 Contest        |      3536 |
| Table Tennis Fall 2022 Competition |      3275 |
| Table Tennis Fall 2022 Competition |      3185 |
| Table Tennis Fall 2022 Competition |      2820 |
| Table Tennis Fall 2022 Competition |      2730 |
| Table Tennis Fall 2022 Competition |      2500 |
| Table Tennis Fall 2022 Competition |      2360 |
+-----------------------------------+-----------+
15 rows in set (0.00 sec)
```

Indexing Design 1

⇒ Index created on *net_id* in the **EventBookings** table

Indexing Design 2

⇒ Index created on *event_id* in the **EventBookings** table

Indexing Design 3

⇒ Index created on *event_id* and *booking_date* in the **EventBookings** table

Reason

⇒ This query doesn't use *net_id* present in the **EventBookings** table. We wanted to check whether creating this can have any impact on the speed of execution.

⇒ Index design 2 which creates index on *event_id*, which is a Foreign key referencing Events(event_*id*) and as previously discussed MySQL does not create an index on Foreign key by default, so to see some performance gains we create this index. It can help speed up the join between **EventBookings** and **Events** tables

⇒ For Index design 3, we use a combination of *event_id* and *booking_date* and this index could help improve performance as *event_id* is used as the JOIN attribute and used in GROUP BY clause.

Explain Analyze of query 3 before indexing

```
mysql>
mysql> explain analyze (SELECT e.event_name, SUM(eb.ticket_count * e.ticket_cost) as total_val
    -> FROM eventbookings eb JOIN events e USING(EVENT_ID)
    -> GROUP BY e.EVENT_ID
    -> ORDER BY total_val desc);
+-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| EXPLAIN


                                        |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| -> Sort: total_val DESC  (actual time=5.713..5.716 rows=19 loops=1)
    -> Stream results  (cost=401.90 rows=1500) (actual time=1.407..5.576 rows=19 loops=1)
        -> Group aggregate: sum((eb.ticket_count * e.ticket_cost))  (cost=401.90 rows=1500) (actual time=1.396..5.548 rows=19 loops=1)
            -> Nested loop inner join  (cost=251.90 rows=1500) (actual time=0.676..5.210 rows=1500 loops=1)
                -> Index scan on e using PRIMARY  (cost=2.15 rows=19) (actual time=0.213..0.259 rows=19 loops=1)
                -> Index lookup on eb using event_id (event_id=e.event_id)  (cost=5.67 rows=79) (actual time=0.213..0.248 rows=79 loops=19)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
1 row in set (0.01 sec)
```

After creating index net_id

```
mysql> create index index_eb1 on eventbookings(net_id);
Query OK, 0 rows affected, 1 warning (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> explain analyze (SELECT e.event_name, SUM(eb.ticket_count * e.ticket_cost) as total_val
    -> FROM eventbookings eb JOIN events e USING(EVENT_ID)
    -> GROUP BY e.EVENT_ID
    -> ORDER BY total_val desc);
+-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| EXPLAIN


                                        |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| -> Sort: total_val DESC  (actual time=4.708..4.709 rows=19 loops=1)
    -> Stream results  (cost=401.90 rows=1500) (actual time=0.763..4.666 rows=19 loops=1)
        -> Group aggregate: sum((eb.ticket_count * e.ticket_cost))  (cost=401.90 rows=1500) (actual time=0.750..4.637 rows=19 loops=1)
            -> Nested loop inner join  (cost=251.90 rows=1500) (actual time=0.518..4.319 rows=1500 loops=1)
                -> Index scan on e using PRIMARY  (cost=2.15 rows=19) (actual time=0.080..0.099 rows=19 loops=1)
                -> Index lookup on eb using event_id (event_id=e.event_id)  (cost=5.67 rows=79) (actual time=0.177..0.211 rows=79 loops=19)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
1 row in set (0.01 sec)
```

After creating index event_id

```
mysql> create index eb1 on EventBookings(event_id);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze (SELECT e.event_name, SUM(eb.ticket_count * e.ticket_cost) as total_val
    -> FROM eventbookings eb JOIN events e USING(EVENT_ID)
    -> GROUP BY e.EVENT_ID ORDER BY total_val desc);
+---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------+
| EXPLAIN

                                          |
+---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------+
| -> Sort: total_val DESC  (actual time=3.994..3.996 rows=19 loops=1)
    -> Stream results  (cost=159.90 rows=500) (actual time=2.831..3.965 rows=19 loops=1)
        -> Group aggregate: sum((eb.ticket_count * e.ticket_cost))  (cost=159.90 rows=500) (actual time=2.821..3.945 rows=19 loops=1)
            -> Nested loop inner join  (cost=109.90 rows=500) (actual time=2.737..3.867 rows=500 loops=1)
                -> Index scan on e using PRIMARY  (cost=2.90 rows=19) (actual time=2.599..2.606 rows=19 loops=1)
                -> Index lookup on eb using eb1 (event_id=e.event_id)  (cost=3.14 rows=26) (actual time=0.056..0.064 rows=26 loops=19)
 |
+---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------+
1 row in set (0.01 sec)
```

After creating index event_id and booking_date

```
mysql> create index eb2 on EventBookings(event_id, booking_date);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze (SELECT e.event_name, SUM(eb.ticket_count * e.ticket_cost) as total_val
    -> FROM eventbookings eb JOIN events e USING(EVENT_ID)
    -> GROUP BY e.EVENT_ID ORDER BY total_val desc);
+---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------+
| EXPLAIN

                                          |
+---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------+
| -> Sort: total_val DESC  (actual time=1.508..1.511 rows=19 loops=1)
    -> Stream results  (cost=159.15 rows=500) (actual time=0.319..1.475 rows=19 loops=1)
        -> Group aggregate: sum((eb.ticket_count * e.ticket_cost))  (cost=159.15 rows=500) (actual time=0.311..1.456 rows=19 loops=1)
            -> Nested loop inner join  (cost=109.15 rows=500) (actual time=0.217..1.358 rows=500 loops=1)
                -> Index scan on e using PRIMARY  (cost=2.15 rows=19) (actual time=0.072..0.080 rows=19 loops=1)
                -> Index lookup on eb using eb1 (event_id=e.event_id)  (cost=3.14 rows=26) (actual time=0.055..0.064 rows=26 loops=19)
 |
+---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------+
1 row in set (0.00 sec)
```

Observation

⇒ The first index *net_id* is not really used in the query so the performance gains from it are not that much, which is as expected.

⇒ *event_id* column is used as the JOIN attribute and in the GROUP BY aggregation. As expected, it does improve the speed of query execution significantly. The speed for joining the tables goes down from 0.213 sec to 0.056 sec.

⇒ Lastly, the combined index does not come into play as seen in the above screenshot. The look up still relies on index *eb1* for retrieving the data faster.