

1. The query (having Like statement) for searching in the text-type column with the clause *UNION* is optimized using the *OR* clause.

We infer from the following figure that query with *OR* clause takes 0.001455 units which is less than that of query with union clause (0.00241975 units). This is because the first query scans each record in the table twice while the second scans only once.

The screenshot shows the SQL IDE interface with the following query in the editor:

```
1 (SELECT * FROM STUDENT WHERE NAME LIKE "A%") UNION (SELECT * FROM STUDENT WHERE LAST_NAME LIKE "B%");
2 • SELECT * FROM STUDENT WHERE NAME LIKE "A%" OR LAST_NAME LIKE "B%";
3
4 • SHOW PROFILES;
5 • SET PROFILING = 1;
```

The Result Grid below the editor displays the execution results:

Query_ID	Duration	Query
1	0.00033475	SHOW WARNINGS
2	0.00241975	(SELECT * FROM STUDENT WHERE NAME LIKE "A%") UNION (SELECT * FROM STUDENT WHERE LAST_NAME LIKE "B%")
3	0.00145500	SELECT * FROM STUDENT WHERE NAME LIKE "A%" OR LAST_NAME LIKE "B%" LIMIT 0, 1000

The screenshot shows the SQL IDE interface with the following query in the editor:

```
8
9
10 • EXPLAIN (SELECT * FROM STUDENT WHERE NAME LIKE "A%") UNION (SELECT * FROM STUDENT WHERE LAST_NAME LIKE "B%");
11
12
```

The Result Grid below the editor displays the execution results:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	STUDENT	HALL	ALL	HALL	HALL	HALL	HALL	700	11.11	Using where
2	UNION	STUDENT	HALL	ALL	HALL	HALL	HALL	HALL	700	11.11	Using where
	UNION RESULT	<union1,2>	HALL	ALL	HALL	HALL	HALL	HALL			Using temporary

SQL File 4* Administration - Data Import/Res...

Limit to 1000 rows

12
13
14 • **EXPLAIN SELECT * FROM STUDENT WHERE NAME LIKE "A%" OR LAST_NAME LIKE "B%";**
15
16

Result Grid Filter Rows: Export: Wrap Cell Contents: TA

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	STUDENT	N/A	ALL	N/A	N/A	N/A	N/A	700	20.99	Using where

Result Grid
Form Editor
Field Types
Query Stats

2. *select * from staff where remarks like '%time%';*
*select * from staff where match(remarks) against ('time');*

Unlike the first query, the second one uses a full text index against the column and thus takes much less time than the first one.

MySQL 8.0 Command Line Client

```
mysql> select * from staff where match(remarks) against ('time');
```

staff_id	name	service	salary	remarks
1	David	cleaner	5925	a. You always come in on time, follow your schedule and adhere to your designated lunch break time.?
3	Craig	cook	5148	c. You schedule your time off in advance and can be relied upon for completing your tasks on time.?
103	Robbie	cook	11257	c. You schedule your time off in advance and can be relied upon for completing your tasks on time.?
203	Daryn	cook	7143	c. You schedule your time off in advance and can be relied upon for completing your tasks on time.?
301	Saul	cleaner	12154	a. You always come in on time, follow your schedule and adhere to your designated lunch break time.?
303	Sonny	cook	12211	c. You schedule your time off in advance and can be relied upon for completing your tasks on time.?
401	Mohsin	cleaner	9853	a. You always come in on time, follow your schedule and adhere to your designated lunch break time.?
403	Nadeem	cook	9827	c. You schedule your time off in advance and can be relied upon for completing your tasks on time.?
4	Scott	cook	7763	d. He replies to emails and calls in time.?
6	Michael	mopper	12347	b. You repeatedly exceed your lunch break time that interferes with your work schedule.?
51	Brian	cleaner	14186	b. You always meet your deadlines and make the best use of your time.?
54	Neil	cook	8270	a. You constantly push deadlines and fail to deliver on time.?
55	Stewart	cook	10740	b. You are usually unable to calculate the time required to complete a task and therefore end up missing deadlines.?
57	Anthony	bathroom	10040	d. You have started delivering low-quality work just to finish your work on time.??
66	Aaron	mopper	14329	c. She does not take the time to develop and sustain positive and beneficial relationships.???
91	Gregor	cleaner	10118	c. She is very punctual and understands the value of others? time. She is never late to a meeting.???
104	Angus	cook	13616	d. He replies to emails and calls in time.?
106	Gareth	mopper	8042	b. You repeatedly exceed your lunch break time that interferes with your work schedule.?
151	Glenn	cleaner	6768	b. You always meet your deadlines and make the best use of your time.?
154	Phillip	cook	8512	a. You constantly push deadlines and fail to deliver on time.?
155	Ronald	cook	11282	b. You are usually unable to calculate the time required to complete a task and therefore end up missing deadlines.?
157	Struan	bathroom	9739	d. You have started delivering low-quality work just to finish your work on time.??
166	Ruaridh	mopper	6048	c. She does not take the time to develop and sustain positive and beneficial relationships.???
191	Johnathan	cleaner	6110	c. She is very punctual and understands the value of others? time. She is never late to a meeting.???
204	Gregory	cook	10136	d. He replies to emails and calls in time.?
206	Myles	mopper	9675	b. You repeatedly exceed your lunch break time that interferes with your work schedule.?
251	Joss	cleaner	6669	b. You always meet your deadlines and make the best use of your time.?
254	Rikki	cook	5864	a. You constantly push deadlines and fail to deliver on time.?
255	Osman	cook	12151	b. You are usually unable to calculate the time required to complete a task and therefore end up missing deadlines.?
257	Gal	bathroom	11562	d. You have started delivering low-quality work just to finish your work on time.??
266	Nathaniel	mopper	8227	c. She does not take the time to develop and sustain positive and beneficial relationships.???
291	Jim	cleaner	6324	c. She is very punctual and understands the value of others? time. She is never late to a meeting.???
304	Stevie	cook	5818	d. He replies to emails and calls in time.?
306	Tommy	mopper	9711	b. You repeatedly exceed your lunch break time that interferes with your work schedule.?
351	Abdul	cleaner	9090	b. You always meet your deadlines and make the best use of your time.?
354	Alfred	cook	9967	a. You constantly push deadlines and fail to deliver on time.?
355	Amar	cook	10112	b. You are usually unable to calculate the time required to complete a task and therefore end up missing deadlines.?
357	Antonio	bathroom	12068	d. You have started delivering low-quality work just to finish your work on time.??
366	Conner	mopper	5338	c. She does not take the time to develop and sustain positive and beneficial relationships.???
391	Kristian	cleaner	14433	c. She is very punctual and understands the value of others? time. She is never late to a meeting.???
404	Noale	cook	12496	d. He replies to emails and calls in time.?
406	Nigel	mopper	8740	b. You repeatedly exceed your lunch break time that interferes with your work schedule.?
451	Cavan	cleaner	7810	b. You always meet your deadlines and make the best use of your time.?
454	Chun	cook	10810	a. You constantly push deadlines and fail to deliver on time.?

20:56 17-04-2022

```
mysql> show profiles;
```

Query_ID	Duration	Query
1	0.00013275	select * from staff where remarks like '%time%'
2	0.00020275	show database
3	0.06217000	show databases
4	0.00020850	SELECT DATABASE()
5	0.00181775	select * from staff where remarks like '%time%'
6	0.00115200	select * from staff where match(remarks) against ('time')
7	0.00160625	select * from staff where remarks like '%time%'
8	0.00104200	select * from staff where match(remarks) against ('time')

8 rows in set, 1 warning (0.00 sec)

```
mysql> explain select * from staff where match(remarks) against ('time');
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	staff	NULL	fulltext	remarks_ft	remarks_ft	0	const	1	100.00	Using where; Ft_hints: sorted

1 row in set, 1 warning (0.00 sec)

```
mysql> explain select * from staff where remarks like '%time%';
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	staff	NULL	ALL	NULL	NULL	NULL	NULL	499	11.11	Using where

1 row in set, 1 warning (0.00 sec)

- Modified the roll_no datatype from 'INT' to 'TINYINT'. This decreased the execution time. This is because smaller data types are usually faster, because they use less space on the disk, in memory, and in the CPU cache. They also generally require fewer CPU cycles to process. Same reason can be applied to mediumint.

```
mysql> SHOW PROFILES;
```

Query_ID	Duration	Query
10	0.04021625	show tables
11	0.00095600	explain select * from staff where match(remarks) against ('time')
12	0.00043350	explain select * from staff where remarks like '%time%'
13	0.00158500	describe student_hostel_payment
14	0.03785000	SELECT * FROM information_schema.columns WHERE table_schema = 'finalhostel'
15	0.47299925	ALTER TABLE STUDENT_HOSTEL_PAYMENT MODIFY TRANSACTION_ID MEDIUMINT(50)
16	0.00132300	ALTER TABLE STUDENT_HOSTEL_PAYMENT MODIFY TRANSACTION_ID MEDIUMINT
17	0.37439150	ALTER TABLE TRANSACTION MODIFY TRANSACTION_ID MEDIUMINT
18	0.26193000	ALTER TABLE TRANSACTION MODIFY TRANSACTION_ID MEDIUMINT()
19	0.00013850	SET PROFILING
20	0.06792775	SET PROFILING=1
21	0.28855750	SELECT * FROM STUDENT LIMIT 5
22	0.02267700	SELECT ROLL_NO FROM STUDENT_HOSTEL_PAYMENT
23	0.03129800	SELECT DATABASE()
24	0.02922075	SELECT ROLL_NO FROM STUDENT_HOSTEL_PAYMENT

15 rows in set, 1 warning (0.00 sec)


```
mysql> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 24 | 0.00043600 | select roll_no from student_hostel_payment |
+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> explain select roll_no from student_hostel_payment;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | student_hostel_payment | NULL | index | NULL | roll_no_idx | 1 | NULL | 32 | 100.00 | Using index |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 31 | 0.03050150 | alter table student_hostel_payment modify roll_no MEDIUMINT(50) |
| 32 | 0.00183800 | select roll_no from student_hostel_payment |
+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

```
mysql> explain select roll_no from student_hostel_payment;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | student_hostel_payment | NULL | index | NULL | roll_no_idx | 3 | NULL | 32 | 100.00 | Using index |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

- We altered the table by adding two new columns `dob_varchar` (with `dob` as `varchar` data type) and `dob_int` (with `dob` as `int` data type) to the `students` table. The values have to be taken from the `dob` (stored as `date` data type) column. We infer from the following that `int` comparisons are faster, for the simple fact that `ints` take up much less space than `varchar` and `date` data types.

The screenshot shows a MySQL IDE window with a query editor and a result grid. The query editor contains the following SQL statements:

```
34 • select * from student where dob = "1999-09-14";
35 • select * from student where dob_varchar = "1999-09-14";
36 • select * from student where dob_int = unix_timestamp("1999-09-14");
37
38 • show profiles;
```

The result grid displays the execution plan for the query `select * from student where dob = "1999-09-14" LIMIT 0, 1000`. The grid has columns for `Query_ID`, `Duration`, and `Query`. The query is shown as a single row in the grid.

Query_ID	Duration	Query
29	0.00142775	select * from student where dob = "1988-02-05" LIMIT 0, 1000
30	0.00126925	select * from student where dob_int = unix_timestamp("1988-02-05") LIMIT 0, 1000
31	0.00129700	select * from student where dob = "1993-04-11" LIMIT 0, 1000
32	0.01291800	select * from student where dob_varchar = "1993-04-11" LIMIT 0, 1000
33	0.00153025	select * from student where dob = "1988-02-05" LIMIT 0, 1000
34	0.00130675	select * from student where dob_varchar = "1988-02-05" LIMIT 0, 1000
35	0.00148475	select * from student where dob_int = unix_timestamp("1988-02-05") LIMIT 0, 1000
36	0.00303875	select * from student LIMIT 0, 1000
37	0.02275100	select * from student where dob = "2002-03-17" LIMIT 0, 1000
38	0.00127925	select * from student where dob_varchar = "2002-03-17" LIMIT 0, 1000
39	0.00137075	select * from student where dob_int = unix_timestamp("2002-03-17") LIMIT 0, 1000
40	0.00470575	select * from student LIMIT 0, 1000
41	0.00127175	select * from student where dob = "1999-09-14" LIMIT 0, 1000
42	0.00192450	select * from student where dob_varchar = "1999-09-14" LIMIT 0, 1000
43	0.00100550	select * from student where dob_int = unix_timestamp("1999-09-14") LIMIT 0, 1000

5. Searching with `isnull` works faster than without explicitly writing `not null`.

```
mysql> SELECT COUNT(SALARY) FROM STAFF;
+-----+
| COUNT(SALARY) |
+-----+
|          443 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT COUNT(SALARY) FROM STAFF WHERE SALARY IS NOT NULL;
+-----+
| COUNT(SALARY) |
+-----+
|          443 |
+-----+
1 row in set (0.00 sec)

mysql> SHOW PROFILES;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 17 | 0.37439150 | ALTER TABLE TRANSACTION MODIFY TRANSACTION_ID MEDIUMINT |
| 18 | 0.26193000 | ALTER TABLE TRANSACTION MODIFY TRANSACTION_ID MEDIUMINT() |
| 19 | 0.00013850 | SET PROFILING |
| 20 | 0.06792775 | SET PROFILING=1 |
| 21 | 0.28855750 | SELECT * FROM STUDENT LIMIT 5 |
| 22 | 0.02267700 | SELECT ROLL_NO FROM STUDENT_HOSTEL_PAYMENT |
| 23 | 0.03129800 | SELECT DATABASE() |
| 24 | 0.02922075 | SELECT ROLL_NO FROM STUDENT_HOSTEL_PAYMENT |
| 25 | 0.19772825 | SELECT COUNT(*) FROM STAFF |
| 26 | 0.00070025 | SELECT COUNT(SALARY) FROM STAFF |
| 27 | 0.01663100 | SELECT COUNT(SALARY) FROM STAFF WHERE SALARY NOT NULL |
| 28 | 0.00062875 | SELECT COUNT(SALARY) FROM STAFF WHERE SALARY IS NOT NULL |
| 29 | 0.07104075 | UPDATE STAFF SET SALARY = NULL WHERE SALARY < 6000 |
| 30 | 0.00080000 | SELECT COUNT(SALARY) FROM STAFF |
| 31 | 0.00063450 | SELECT COUNT(SALARY) FROM STAFF WHERE SALARY IS NOT NULL |
+-----+-----+-----+
15 rows in set, 1 warning (0.00 sec)
```

6. `mysql` caches the select query as well as the result set, allowing identical selects to run faster as data is fetched from the memory cache. We are not able to perform caching because the `mysql` version 8.0 does not support it.
7. We performed *JOIN* on `warden` and `furniture` tables and compared it with a subquery. We infer that retrieval time of the query using joins is faster than that of a subquery thus helps better in the optimization. This is because the subquery runs the internal queries first and then from the resulting table again filters out the actual results while *JOIN*

produces the result in one go.

```
mysql> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 7 | 0.00076600 | select warden_id from warden where block_name in (select block_name from furniture where block_name='A') |
| 8 | 0.00194200 | select warden.block_name from warden natural join furniture where furniture.block_name = 'A' |
+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

```
mysql> explain select warden_id from warden where block_name in (select block_name from furniture where block_name='A');
+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+
| 1 | SIMPLE | warden | NULL | ref | block_name_idx | block_name_idx | 83 | const | 49 | 100.00 | Using where; |
| 1 | SIMPLE | <subquery2> | NULL | eq_ref | <auto_distinct_key> | <auto_distinct_key> | 83 | final.warden.block_name | 1 | 100.00 | Using where |
| 2 | MATERIALIZED | furniture | NULL | ref | block_name | block_name | 83 | const | 49 | 100.00 | Using index |
+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)
```

```
mysql> explain select warden.block_name from warden natural join furniture where furniture.block_name = 'A';
+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+
| 1 | SIMPLE | warden | NULL | ref | block_name_idx | block_name_idx | 83 | const | 49 | 100.00 | Using index |
| 1 | SIMPLE | furniture | NULL | ref | block_name | block_name | 83 | const | 49 | 100.00 | Using index |
+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

However, more joins in a query means the database server has to do more work, which means that it is a more time consuming process to retrieve data.

```
mysql> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 9 | 0.00286850 | select warden.id from warden natural join furniture natural join hostel where hostel.block_name='A' |
| 10 | 0.01692025 | select warden.block_name from warden natural join furniture natural join hostel where hostel.block_name='A' |
| 11 | 0.00085725 | explain select warden.block_name from warden natural join furniture natural join hostel where hostel.block_name='A' |
+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)
```

```
mysql> explain select warden.block_name from warden natural join furniture natural join hostel where hostel.block_name='A';
+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+
| 1 | SIMPLE | hostel | NULL | const | PRIMARY | PRIMARY | 82 | const | 1 | 100.00 | Using index |
| 1 | SIMPLE | warden | NULL | ref | block_name_idx | block_name_idx | 83 | const | 49 | 100.00 | Using index |
| 1 | SIMPLE | furniture | NULL | ref | block_name | block_name | 83 | const | 49 | 100.00 | Using index |
+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)
```

8. The execution time is decreased as we used natural join instead of subqueries. This is because Subquery runs the internal queries first and then from the result set again filters out the actual results. Wherein join runs the and produces the result in one go.

```
mysql> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 19 | 0.42159175 | select name from student where roll_no in (select roll_no from postgrad where block_name in (select block_name from furniture where room_no in (select room_no from hostel_rooms where sharing_type = 'Double')))) |
| 20 | 0.01484350 | select student.name from student natural join postgrad natural join furniture natural join hostel_rooms where sharing_type='Double' |
+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

```
mysql> explain select name from student where roll_no in (select roll_no from postgrad where block_name in (select block_name from furniture where room_no in (select room_no from hostel_rooms where sharing_type = 'Double')));
+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | row |
+-----+-----+-----+
| 1 | SIMPLE | postgrad | NULL | index | PRIMARY,block_name_idx | block_name_idx | 83 | NULL | 35 | |
| 100.00 | Using where; Using index; Start temporary | | | | | | | | | |
| 1 | SIMPLE | student | NULL | eq_ref | PRIMARY | PRIMARY | 4 | final.postgrad.roll_no | |
| 100.00 | NULL | | | | | | | | |
| 1 | SIMPLE | hostel_rooms | NULL | ALL | PRIMARY | NULL | NULL | NULL | 49 |
| 10.00 | Using where; Using join buffer (hash join) | | | | | | | | |
| 1 | SIMPLE | furniture | NULL | ref | block_name | block_name | 88 | final.postgrad.block_name,final.hostel_rooms.room_no | |
| 100.00 | Using index; End temporary | | | | | | | | |
+-----+-----+-----+
4 rows in set, 1 warning (0.00 sec)
```

```
mysql> explain select student.name from student natural join postgrad natural join furniture natural join hostel_rooms where sharing_type='Double';
+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows |
+-----+-----+-----+
| 1 | SIMPLE | furniture | NULL | index | block_name | block_name | 88 | NULL | 388 |
| 100.00 | Using where; Using index | | | | | | | | |
| 1 | SIMPLE | hostel_rooms | NULL | eq_ref | PRIMARY,block_name | PRIMARY | 86 | final.furniture.room_no,final.furniture.block_name | 1 |
| 10.00 | Using where | | | | | | | | |
| 1 | SIMPLE | postgrad | NULL | ref | PRIMARY,block_name_idx | block_name_idx | 83 | final.furniture.block_name | 35 |
| 100.00 | Using index | | | | | | | | |
| 1 | SIMPLE | student | NULL | eq_ref | PRIMARY | PRIMARY | 4 | final.postgrad.roll_no | 1 |
| 100.00 | NULL | | | | | | | | |
+-----+-----+-----+
4 rows in set, 1 warning (0.00 sec)
```

On the web app server of the database,”SEARCH” option has been added on the “Student” table page to be able to search the names of students starting from a certain alphabet and the last names starting from a certain alphabet. This helps in accessing required information easily.

Group G1 Contribution (out of 50%)

Shubham Ghosh 21120017- 12.5%
Pranav Ninawe 21120015 - 12.5%
Rovin Singh 21120016 - 12.5%
Darshan Patil 21120006 - 12.5%
(Equal participation of all group members)

Group G2 Contribution (out of 50%)

Iram Nawab 21250011 - 12.5%
Manasvi Kothari 21120011 - 12.5%
Hardik Mahur 19110086 - 12.5%
Shoaib Alam 21250024 - 12.5%
(Equal participation of all group members)