

Name: Dhruv Bheda
Batch: B1

SapID: 60004200102

ADBMS Experiment-3

Dhruv Bheda
60004200102
B1/B1

ADBMS EXP-3

Aim : To simulate query optimisation by performing SQL query on database.

Theory : Query optimization is of great importance for performance of a relational database, especially for execution of complexed SQL statements. There are 2 ways :-
i) Heuristic Based ii) Cost Based

Heuristic Based :-

A query tree is a data structure that corresponds to a relational algebra expression. The same query could be correspond to many different relational expressions & hence many different query trees. The task of heuristic optimization of query trees is to find a final query tree that is efficient to execute. The main heuristic is to apply first the operations that reduce the size of intermediate results.

Cost Based :-

Estimate and compare the costs of executing a query using different execution strategies and chose the strategy with lowest cost estimate. The cost of any query includes access cost to secondary storage, storage cost, memory usage cost, no. of memory buffer at time of execution, communication etc.

Conclusion :-

This, we performed table level & index level optimisation & compared it to results of no optimisation

1) SELECT QUERY

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 • USE sakila;
2 • select * from actor where upper(last_name) like "NGEN%";
3
```

The left sidebar shows the database schema, including the **actor** table. The right sidebar shows the **Visual Explain** tab, which displays the execution plan:

```
graph TD
    query_block[query_block #1] -- 25 25 | 200 rows --> full_scan[Full Table Scan]
    full_scan --> actor[actor]
```

The **actor** table structure is shown in the left sidebar:

Column	Type
actor_id	smallint UNSIGNED AUTO INCREMENTAL PRIMARY KEY
first_name	VARCHAR(45)
last_name	VARCHAR(45)
last_update	TIMESTAMP

The right sidebar contains a text box: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

BEFORE OPTIMIZING

The screenshot shows the MySQL Workbench interface with the same query as above. The **Query Statistics** tab is selected, displaying the following information:

Timing (as measured at client side):
Execution time: 0:00:0.000000000

Timing (as measured by the server):
Execution time: 0:00:0.000406000
Table lock wait time: 0:00:0.000003000

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 4
Rows examined: 200

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event ID: 92
Thread ID: 60

The right sidebar contains the same text box as in the previous screenshot: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

AFTER OPTIMIZING

MySQL Workbench interface showing the execution of a query. The query is:

```
1 • USE sakila;
2 • select * from actor where upper(last_name) like "NGEN%";
3 • optimize table actor;
```

The query statistics are displayed below the query:

Timing (as measured at client side):
Execution time: 0:00:0.06200000

Timing (as measured by the server):
Execution time: 0:00:0.05738530
Table lock wait time: 0:00:0.00003900

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 0
Rows examined: 0

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
At least one index was used

Other Info:
Event Id: 106
Thread Id: 60

The interface also shows the schema structure on the left and the table details for 'actor' on the bottom left.

2) NESTED

MySQL Workbench interface showing the execution of a nested query. The query is:

```
1 • USE sakila;
2 • select * from actor where upper(last_name) like "NGEN%";
3 • SELECT concat(first_name, " ", last_name) AS name, email
4 • from customer where customer_id IN
5 • (select customer_id from rental where inventory_id in
6 • (select inventory_id from inventory where film_id in
7 • (select film_id from film_category join category using (category_id) where category.name = "Action"))));
```

The query execution plan is visualized below the query:

Visual Explain: Display Info Read + Eval cost Overview View Source

The execution plan shows a series of nested loops and key lookups:

- Full Table Scan on category (1.8499999999999999)
- Non-Unique Key Lookup on film_category (10.41)
- Non-Unique Key Lookup on inventory (72.88)
- Non-Unique Key Lookup on rental (584.97)
- Unique Key Lookup on customer PRIMARY (477.73999999999995)

The plan also shows the cost of each step and the number of rows processed. The final result is a query block with 1483 rows.

BEFORE OPTIMIZING

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
1 • USE sakila;
2 • select * from actor where upper(last_name) like "NGENK";
3 • SELECT concat(first_name, " ",last_name) AS name, email
4 • from customer where customer_id IN
5 • (select customer_id from rental where inventory_id in
6 • (select inventory_id from inventory where film_id in
7 • (select film_id from film_category join category using (category_id) where category.name = "Action"))));
```

The Query Statistics panel displays the following information:

- Timing (as measured at client side):** Execution time: 0:00:0.00000000
- Timing (as measured by the server):** Execution time: 0:00:0.00514680, Table lock wait time: 0:00:0.00000000
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 510, Rows examined: 2616
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 1
- Joins per Type:** Full table scans (Select_scan): 1, Joins using table scans (Select_full_join): 0, Joins using range search (Select_full_range_join): 0, Joins with range checks (Select_range_check): 0, Joins using range (Select_range): 0
- Sorting:** Sorted rows (Sort_rows): 0, Sort merge passes (Sort_merge_passes): 0, Sorts with ranges (Sort_range): 0, Sorts with table scans (Sort_scan): 0
- Index Usage:** No Index used
- Other Info:** Event Id: 130, Thread Id: 60

The left sidebar shows the database schema with the 'actor' table selected. The table structure is as follows:

Column	Type
actor_id	smallint UNSIGNED, AI, PK
first_name	varchar(45)
last_name	varchar(45)
last_update	timestamp

AFTER OPTIMIZING

The screenshot shows the MySQL Workbench interface with the same SQL query as before, but with an additional line at the bottom:

```
9 • optimize table customer , rental , inventory , category ;
```

The Query Statistics panel displays the following information:

- Timing (as measured at client side):** Execution time: 0:00:0.62500000
- Timing (as measured by the server):** Execution time: 0:00:0.61268200, Table lock wait time: 0:00:0.00011400
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 0, Rows examined: 0
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 0
- Joins per Type:** Full table scans (Select_scan): 0, Joins using table scans (Select_full_join): 0, Joins using range search (Select_full_range_join): 0, Joins with range checks (Select_range_check): 0, Joins using range (Select_range): 0
- Sorting:** Sorted rows (Sort_rows): 0, Sort merge passes (Sort_merge_passes): 0, Sorts with ranges (Sort_range): 0, Sorts with table scans (Sort_scan): 0
- Index Usage:** At least one index was used
- Other Info:** Event Id: 135, Thread Id: 60

The left sidebar shows the database schema with the 'actor' table selected. The table structure is as follows:

Column	Type
actor_id	smallint UNSIGNED, AI, PK
first_name	varchar(45)
last_name	varchar(45)
last_update	timestamp

3) LEFT JOIN

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
6 (select inventory_id from inventory where film_id in
7 (select film_id from film_category join category using (category_id) where category.name = "Action"));
8
9 select stf.first_name, stf.last_name, ad.address, ad.district, ad.postal_code, ad.city_id
10 from staff stf
11 left join address ad
12 on stf.address_id = ad.address_id;
```

The execution plan is displayed below the query, showing a nested loop join. The first step is a 'Full Table Scan' on the 'staff' table, which takes 3.2 seconds and returns 2 rows. The second step is a 'Unique Key Lookup' on the 'address' table, which takes 0.7 seconds and returns 1 row. The total query cost is 3.90.

Table: actor

Columns:	
actor_id	smallint UNSIGNED
first_name	VARCHAR(45)
last_name	VARCHAR(45)
last_update	TIMESTAMP

BEFORE OPTIMIZING

The screenshot shows the MySQL Workbench interface with the same query as above. The 'Query Statistics' tab is selected, displaying the following information:

Timing (as measured at client side):
Execution time: 0:00:0.000000000

Timing (as measured by the server):
Execution time: 0:00:0.00041500
Table lock wait time: 0:00:0.00000400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 2
Rows examined: 4

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event ID: 170
Thread ID: 60

AFTER OPTIMIZING

The screenshot shows the MySQL Workbench interface with a query window titled 'SQL File 4*'. The query is as follows:

```
9 • select stf.first_name, stf.last_name, ad.address, ad.district, ad.postal_code, ad.city_id
10 from staff stf
11 left join address ad
12 on stf.address_id = ad.address_id;
13
14 • optimize table staff, address
```

The 'Query Statistics' panel displays the following information:

- Timing (as measured at client side):**
Execution time: 0:00:0.18700000
- Timing (as measured by the server):**
Execution time: 0:00:0.17137930
Table lock wait time: 0:00:0.00011100
- Errors:**
Had Errors: NO
Warnings: 0
- Rows Processed:**
Rows affected: 0
Rows sent to client: 0
Rows examined: 0
- Temporary Tables:**
Temporary disk tables created: 0
Temporary tables created: 0
- Joins per Type:**
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0
- Sorting:**
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0
- Index Usage:**
At least one index was used
- Other Info:**
Event Id: 191
Thread Id: 60

The 'Table: actor' information is also visible:

Columns:

- actor_id: smallint UNSIGNED, AI PK
- first_name: varchar(45)
- last_name: varchar(45)
- last_update: timestamp

4) RIGHT JOIN

The screenshot shows the MySQL Workbench interface with a query window titled 'SQL File 4*'. The query is as follows:

```
15 from film film
16 right join film_actor film_act
17 on film.film_id = film_act.film_id
18 group by film.title
19 order by number_of_actors desc;
```

The 'Visual Explain' panel displays the execution plan:

- Full Index Scan** (film_act, idx_fk_film_id) with 556.38 rows and 8.48K rows.
- Unique Key Lookup** (film, PRIMARY) with 556.56 rows and 1 row.
- GROUP** operation (tmp table) with 6154.75 rows and 48K rows.
- ORDER** operation (filesort) with 6154.92 rows.

The 'Table: actor' information is also visible:

Columns:

- actor_id: smallint UNSIGNED, AI PK
- first_name: varchar(45)
- last_name: varchar(45)
- last_update: timestamp

BEFORE OPTIMIZING

The screenshot shows the MySQL Workbench interface with a query executed. The query is:

```
15 from film film
16 right join film_actor fim_act
17 on film.film_id = fim_act.film_id
18 group by film.title
19 order by number_of_actors desc;
20
21
```

The Query Statistics panel shows the following details:

- Timing (as measured at client side):** Execution time: 0:00:0.00000000
- Timing (as measured by the server):** Execution time: 0:00:0.00533390, Table lock wait time: 0:00:0.00000300
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 997, Rows examined: 11921
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 1
- Joins per Type:** Full table scans (Select_scan): 1, Joins using table scans (Select_full_join): 0, Joins using range search (Select_full_range_join): 0, Joins with range checks (Select_range_check): 0, Joins using range (Select_range): 0
- Sorting:** Sorted rows (Sort_rows): 997, Sort merge passes (Sort_merge_passes): 0, Sorts with ranges (Sort_range): 0, Sorts with table scans (Sort_scan): 1
- Index Usage:** At least one Index was used
- Other Info:** Event Id: 325, Thread Id: 60

The left sidebar shows the database schema with the 'actor' table selected. The table structure is:

Column	Type
actor_id	smallint(5) UNSIGNED
first_name	VARCHAR(45)
last_name	VARCHAR(45)
last_update	TIMESTAMP

AFTER OPTIMIZING

The screenshot shows the same MySQL Workbench interface, but the query has been optimized. The query is:

```
15 from film film
16 right join film_actor fim_act
17 on film.film_id = fim_act.film_id
18 group by film.title
19 order by number_of_actors desc;
20
21 • optimize table film , film_actor
```

The Query Statistics panel shows the following details:

- Timing (as measured at client side):** Execution time: 0:00:0.23400000
- Timing (as measured by the server):** Execution time: 0:00:0.21942460, Table lock wait time: 0:00:0.00005600
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 0, Rows examined: 0
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 0
- Joins per Type:** Full table scans (Select_scan): 0, Joins using table scans (Select_full_join): 0, Joins using range search (Select_full_range_join): 0, Joins with range checks (Select_range_check): 0, Joins using range (Select_range): 0
- Sorting:** Sorted rows (Sort_rows): 0, Sort merge passes (Sort_merge_passes): 0, Sorts with ranges (Sort_range): 0, Sorts with table scans (Sort_scan): 0
- Index Usage:** At least one Index was used
- Other Info:** Event Id: 330, Thread Id: 60

The left sidebar shows the database schema with the 'actor' table selected. The table structure is:

Column	Type
actor_id	smallint(5) UNSIGNED
first_name	VARCHAR(45)
last_name	VARCHAR(45)
last_update	TIMESTAMP

5) INNER JOIN

The screenshot shows the MySQL Workbench interface with a query window containing the following SQL code:

```
15 from film film
16 inner join film_actor fim_act
17 on film.film_id = fim_act.film_id
18 group by film.title
19 order by number_of_actors desc;
```

The execution plan is displayed below the query, showing a nested loop join. The first step is a 'Full Index Scan' on the 'film' table, which takes 110.18 ms and processes 1000 rows. This is followed by a 'Non-Unique Key Lookup' on the 'film_actor' table, which takes 1303.0 ms and processes 5 rows. The result of the join is then grouped and ordered, taking 1473.18 ms and processing 1473 rows. The final output is sorted by the number of actors per film.

Table: actor

Columns:

- actor_id: smallest UN
- first_name: varchar(45)
- last_name: varchar(45)
- last_update: timestamp

BEFORE OPTIMIZING

The screenshot shows the same MySQL Workbench interface with the same query. The 'Query Statistics' window is open, displaying the following information:

Timing (as measured at client side):

- Execution time: 0:00:0.00000000

Timing (as measured by the server):

- Execution time: 0:00:0.00371900
- Table lock wait time: 0:00:0.00000300

Errors:

- Had Errors: NO
- Warnings: 0

Rows Processed:

- Rows affected: 0
- Rows sent to client: 997
- Rows examined: 7459

Temporary Tables:

- Temporary disk tables created: 0
- Temporary tables created: 0

Joins per Type:

- Full table scans (Select_scan): 1
- Joins using table scans (Select_full_join): 0
- Joins using range search (Select_full_range_join): 0
- Joins with range checks (Select_range_check): 0
- Joins using range (Select_range): 0

Sorting:

- Sorted rows (Sort_rows): 997
- Sort merge passes (Sort_merge_passes): 0
- Sorts with ranges (Sort_range): 0
- Sorts with table scans (Sort_scan): 1

Index Usage:

- At least one Index was used

Other Info:

- Event ID: 208
- Thread ID: 60

AFTER OPTIMIZING

The screenshot displays the MySQL Workbench environment. The top toolbar includes icons for file operations, editing, and running queries. The left sidebar shows the 'SCHEMAS' tree with 'sakila' selected, and the 'Tables' list under 'actor'. The main window shows a SQL query in 'SQL File 4*' that joins 'film' and 'film_actor' tables, grouped by 'film.title' and ordered by 'number_of_actors desc'. Below the query editor, the 'Query Statistics' tab is active, showing execution metrics. The 'Table: actor' is highlighted in the left sidebar, and its columns are listed below. The bottom status bar shows 'actor 30' and 'Result 31'.

SQL Query:

```

15  from film film
16  inner_join film_actor fim_act
17  on film.film_id = fim_act.film_id
18  group by film.title
19  order by number_of_actors desc;
20
21  * optimize table film , film_actor
--

```

Query Statistics:

- Timing (as measured at client side):**
 - Execution time: 0:00:0.18800000
- Timing (as measured by the server):**
 - Execution time: 0:00:0.17838100
 - Table lock wait time: 0:00:0.00006000
- Errors:**
 - Had Errors: NO
 - Warnings: 0
- Rows Processed:**
 - Rows affected: 0
 - Rows sent to client: 0
 - Rows examined: 0
- Temporary Tables:**
 - Temporary dis tables created: 0
 - Temporary tables created: 0
- Joins per Type:**
 - Full table scans (Select_scan): 0
 - Joins using table scans (Select_full_join): 0
 - Joins using range search (Select_full_range_join): 0
 - Joins with range checks (Select_range_check): 0
 - Joins using range (Select_range): 0
- Sorting:**
 - Sorted rows (Sort_rows): 0
 - Sort merge passes (Sort_merge_passes): 0
 - Sorts with ranges (Sort_range): 0
 - Sorts with table scans (Sort_scan): 0
- Index Usage:**
 - At least one Index was used
- Other Info:**
 - Event ID: 293
 - Thread ID: 60

Table: actor

Columns:

Column	Size	Null	Index
actor_id	smallint	UN	AI PK
first_name	varchar(45)		
last_name	varchar(45)		
last_update	timestamp		

6) CROSS JOIN

[illegible]

BEFORE OPTIMIZING

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
15 from film film
16 cross join film_actor fim_act
17 on film.film_id = fim_act.film_id
18 group by film.title
19 order by number_of_actors desc;
20
21
```

The Query Statistics panel displays the following information:

- Timing (as measured at client side):** Execution time: 0:00:0.01300000
- Timing (as measured by the server):** Execution time: 0:00:0.00581580, Table lock wait time: 0:00:0.00000300
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 997, Rows examined: 7459
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 0
- Joins per Type:** Full table scans (Select_scan): 1, Joins using table scans (Select_full_join): 0, Joins using range search (Select_full_range_join): 0, Joins with range checks (Select_range_check): 0, Joins using range (Select_range): 0
- Sorting:** Sorted rows (Sort_rows): 997, Sort merge passes (Sort_merge_passes): 0, Sorts with ranges (Sort_range): 0, Sorts with table scans (Sort_scan): 1
- Index Usage:** At least one Index was used
- Other Info:** Event Id: 222, Thread Id: 60

The left sidebar shows the database schema, and the bottom status bar indicates the current row is 23.

AFTER OPTIMIZING

The screenshot shows the MySQL Workbench interface after optimization. The SQL editor contains the following query:

```
15 from film film
16 cross join film_actor fim_act
17 on film.film_id = fim_act.film_id
18 group by film.title
19 order by number_of_actors desc;
20
21 optimize table film , film_actor
22
```

The Query Statistics panel displays the following information:

- Timing (as measured at client side):** Execution time: 0:00:0.18800000
- Timing (as measured by the server):** Execution time: 0:00:0.19220070, Table lock wait time: 0:00:0.00007700
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 0, Rows examined: 0
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 0
- Joins per Type:** Full table scans (Select_scan): 0, Joins using table scans (Select_full_join): 0, Joins using range search (Select_full_range_join): 0, Joins with range checks (Select_range_check): 0, Joins using range (Select_range): 0
- Sorting:** Sorted rows (Sort_rows): 0, Sort merge passes (Sort_merge_passes): 0, Sorts with ranges (Sort_range): 0, Sorts with table scans (Sort_scan): 0
- Index Usage:** At least one Index was used
- Other Info:** Event Id: 245, Thread Id: 60

The left sidebar shows the database schema, and the bottom status bar indicates the current row is 29.

USING INDEXING

1)SIMPLE QUERY

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 • USE sakila;  
2 • select * from film where length = 120 ;
```

The execution plan for this query is displayed in the 'Visual Explain' tab. It shows a single step: 'Non-Unique Key Lookup' on the 'film' table, with a cost of 9.22 and 9 rows. The 'Query cost' is also 9.22.

On the right side, there is a sidebar with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

BEFORE INDEXING

The screenshot shows the MySQL Workbench interface with the same query as before. The 'Query Statistics' tab is selected, displaying the following information:

Timing (as measured at client side):
Execution time: 0:00:0.000000000

Timing (as measured by the server):
Execution time: 0:00:0.00041420
Table lock wait time: 0:00:0.00000300

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 9
Rows examined: 9

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
At least one Index was used

Other Info:
Event ID: 104
Thread ID: 131

On the right side, there is a sidebar with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

AFTER INDEXING

MySQL Workbench interface showing the execution of a query. The query is:

```
1 • USE sakila;
2 • create index length on film(length);
3 • select * from film where length = 120 ;
```

The Query Statistics section displays the following information:

- Timing (as measured at client side):** Execution time: 0:00:0.000000000
- Timing (as measured by the server):** Execution time: 0:00:0.00036020, Table lock wait time: 0:00:0.00000200
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 9, Rows examined: 9
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 0
- Joins per Type:** Full table scans (Select_scan): 0, Joins using table scans (Select_full_join): 0, Joins using range search (Select_full_range_join): 0, Joins with range checks (Select_range_check): 0, Joins using range (Select_range): 0
- Sorting:** Sorted rows (Sort_rows): 0, Sort merge passes (Sort_merge_passes): 0, Sorts with ranges (Sort_range): 0, Sorts with table scans (Sort_scan): 0
- Index Usage:** At least one index was used
- Other Info:** Event Id: 140, Thread Id: 131

The right sidebar shows a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

2) NESTED

MySQL Workbench interface showing the execution of a nested query. The query is:

```
1 • USE sakila;
2 • select * from actor where upper(last_name) like "NGEN";
3 • SELECT concat(first_name, " ", last_name) AS name, email
4 • from customer where customer_id IN
5 • (select customer_id from rental where inventory_id in
6 • (select inventory_id from inventory where film_id in
7 • (select film_id from film_category join category using (category_id) where category.name = "Action"))));
```

The Visual Explain section displays the execution plan, showing a series of nested loops and key lookups. The plan starts with a "Full Table Scan" on the "category" table, followed by a "Non-Unique Key Lookup" on the "film_category" table, and then a series of "Non-Unique Key Lookup" operations on the "inventory", "rental", and "customer" tables. The final step is a "Unique Key Lookup" on the "customer" table. The plan also shows a "DISTINCT" operation and a "tmp table" used for the final result.

The right sidebar shows a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

BEFORE INDEXING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

SCHEMAS

- b'sakila
- newschema
- sakila
 - Tables
 - actor
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - address
 - category
 - city
 - country
 - customer
 - film
 - film_actor
 - film_category
 - film_text
 - inventory
 - language
 - payment

Administration Schemas

Information

Table: actor

Columns:

- actor_id smallint(4) PK
- first_name varchar(45)
- last_name varchar(45)
- last_update timestamp

SQL File 4*

```
1 • USE sakila;
2 • select * from actor where upper(last_name) like "NGENK";
3 • SELECT concat(first_name, " ",last_name) AS name, email
4   from customer where customer_id IN
5   (select customer_id from rental where inventory_id in
6    (select inventory_id from inventory where film_id in
7     (select film_id from film_category join category using (category_id) where category.name = "Action"))));
```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.000000000

Timing (as measured by the server):
Execution time: 0:00:0.00514680
Table lock wait time: 0:00:0.000000000

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 510
Rows examined: 2616

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 1

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 130
Thread Id: 60

actor 6 Result 7 Result 8 x

Apply Revert Context Help Snippets

Output

AFTER INDEXING

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

SCHEMAS

- b'sakila
- newschema
- sakila
 - Tables
 - actor
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - address
 - category
 - city
 - country
 - customer
 - film
 - film_actor
 - film_category
 - film_text
 - inventory
 - language
 - payment

Administration Schemas

Information

Table: category

Columns:

- category_id tinyint(4) PK
- name varchar(255)
- last_update timestamp

SQL File 4*

```
1 • create index first_name on customer(first_name);
2 • select concat(first_name, " ",last_name) AS name, email
3   from customer where customer_id IN
4   (select customer_id from rental where inventory_id in
5    (select inventory_id from inventory where film_id in
6     (select film_id from film_category join category using (category_id) where category.name = "Action"))));
```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.016000000

Timing (as measured by the server):
Execution time: 0:00:0.00457000
Table lock wait time: 0:00:0.000000000

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 510
Rows examined: 2616

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 1

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 130
Thread Id: 134

Result 18 Result 19 x

Apply Context Help Snippets

Output

3) LEFT JOIN

The screenshot shows MySQL Workbench with a query window containing the following SQL:

```
1 select film.title, count(*) number_of_actors
2 from film film
3 left join film_actor film_act
4 on film.film_id = film_act.film_id
5 group by film.rental_duration
6 order by number_of_actors desc;
```

The Visual Explain tab displays the execution plan:

- Full Table Scan** (film) with 110.18 MB and 1000 rows.
- Non-unique Key Lookup** (film_act) with 1363.0 MB and 0 rows.
- GROUP** operation with 1473.18 MB and 48K rows.
- ORDER** operation (filesort) with 1473.18 MB.

The right sidebar shows a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

BEFORE INDEXING

The screenshot shows the same query in MySQL Workbench, but with the Query Statistics tab selected:

Timing (as measured at client side):
Execution time: 0:00:0.01600000

Timing (as measured by the server):
Execution time: 0:00:0.00683940
Table lock wait time: 0:00:0.00000300

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 5
Rows examined: 6467

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 1

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 5
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 1

Index Usage:
No index used

Other Info:
Event ID: 382
Thread ID: 134

The right sidebar shows the same message as the previous screenshot.

AFTER INDEXING

This screenshot shows the MySQL Workbench interface after executing a query. The SQL editor contains the following query:

```
1 create index rental_duration on film(rental_duration);
2 select flm.title , count(*) number_of_actors
3 from film flm
4 left join film_actor flm_act
5 on flm.film_id = flm_act.film_id
6 group by flm.rental_duration
7 order by number_of_actors desc;
```

The Query Statistics panel displays the following information:

- Timing (as measured at client side):** Execution time: 0:00:0.01500000
- Timing (as measured by the server):** Execution time: 0:00:0.00662890, Table lock wait time: 0:00:0.00000200
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 5, Rows examined: 6467
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 1
- Joins per Type:** Full table scans (Select_scan): 1, Joins using table scans (Select_full_join): 0, Joins using range search (Select_range_join): 0, Joins with range checks (Select_range_check): 0, Joins using range (Select_range): 0
- Sorting:** Sorted rows (Sort_rows): 5, Sort merge passes (Sort_merge_passes): 0, Sorts with ranges (Sort_range): 0, Sorts with table scans (Sort_scan): 1
- Index Usage:** No Index used
- Other Info:** Event Id: 279, Thread Id: 134

The left sidebar shows the Schemas tree with the 'film' table selected. The bottom status bar indicates 'Result 15' and 'Result 16'.

4) RIGHT JOIN

This screenshot shows the MySQL Workbench interface with a right join query and its execution plan. The SQL editor contains the following query:

```
15 from film flm
16 right join film_actor flm_act
17 on flm.film_id = flm_act.film_id
18 group by flm.title
19 order by number_of_actors desc;
```

The Visual Explain panel displays the execution plan for the query. The plan shows a 'Full Index Scan' on the 'film_actor' table (556.38, 5.46K rows) and a 'Unique Key Lookup' on the 'film' table (558.55, 1 row). These are joined via a 'nested loop' join. The result is then grouped and ordered. The final output is 'Query cost: 6154.92' and 'query_block #1'.

The left sidebar shows the Schemas tree with the 'sakila' database selected. The bottom status bar indicates 'actor 36', 'Result 37', 'Result 38', and 'Result 39'.

BEFORE INDEXING

This screenshot shows the MySQL Workbench interface with a query executed. The query is:

```
1 select flm.title , count(*) number_of_actors
2 from film flm
3 right join film_actor flm_act
4 on flm.film_id = flm_act.film_id
5 group by flm.title
6 order by number_of_actors desc;
```

The Query Statistics panel shows the following performance metrics:

- Timing (as measured at client side):** Execution time: 0:00:0.000000000
- Timing (as measured by the server):** Execution time: 0:00:0.005629000, Table lock wait time: 0:00:0.000003300
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 997, Rows examined: 11921
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 1
- Joins per Type:** Full table scans (Select_scan): 1, Joins using table scans (Select_full_join): 0, Joins using range search (Select_full_range_join): 0, Joins with range checks (Select_range_check): 0, Joins using range (Select_range): 0
- Sorting:** Sorted rows (Sort_rows): 997, Sort merge passes (Sort_merge_passes): 0, Sorts with ranges (Sort_range): 0, Sorts with table scans (Sort_scan): 1
- Index Usage:** At least one Index was used
- Other Info:** Event ID: 244, Thread ID: 134

The result set shows 14 rows. The interface also displays the database schema on the left and a sidebar with various tools like Result Grid, Form Editor, Field Types, Query Status, and Execution Plan.

AFTER INDEXING

This screenshot shows the MySQL Workbench interface with the same query executed after creating an index on the rental_rate column of the film table. The query is:

```
1 create index rental_rate on film(rental_rate);
2 select flm.title , count(*) number_of_actors
3 from film flm
4 right join film_actor flm_act
5 on flm.film_id = flm_act.film_id
6 group by flm.title
7 order by number_of_actors desc;
```

The Query Statistics panel shows the following performance metrics:

- Timing (as measured at client side):** Execution time: 0:00:0.015000000
- Timing (as measured by the server):** Execution time: 0:00:0.005033120, Table lock wait time: 0:00:0.000003300
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 997, Rows examined: 11921
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 1
- Joins per Type:** Full table scans (Select_scan): 1, Joins using table scans (Select_full_join): 0, Joins using range search (Select_full_range_join): 0, Joins with range checks (Select_range_check): 0, Joins using range (Select_range): 0
- Sorting:** Sorted rows (Sort_rows): 997, Sort merge passes (Sort_merge_passes): 0, Sorts with ranges (Sort_range): 0, Sorts with table scans (Sort_scan): 1
- Index Usage:** At least one Index was used
- Other Info:** Event ID: 241, Thread ID: 134

The result set shows 13 rows. The interface also displays the database schema on the left and a sidebar with various tools like Result Grid, Form Editor, Field Types, Query Status, and Execution Plan.