Lab Manual on Indexes

(May 2022)

Objectives

By the end of this lab, you will be able to:

1) **List** the indexes that are associated with a relation
2) **Determine** which index a SELECT query is using by executing the EXPLAIN command
3) **Determine** which index a SELECT query is using by viewing the "Execution Plan" window in MySQL Workbench
4) **Create** an unclustered BTREE index
5) **Compare** the cost of executing queries by using the "query cost" statistic and the "number of rows examined per scan" statistic

Tools

MySQL DBMS (>=MySQL DBMS 8.0)

MySQL Workbench (>=MySQL Workbench 8.0)

Time

1 hour 30 minutes

Prerequisites

1. You should have installed MySQL DBMS (>= MySQL 8.0) and MySQL Workbench (>=MySQL Workbench 8.0). They are available via this link.
2. You should have imported the sample database called "classicmodels" into your database. It is available on the e-learning portal.

Narrative

The classicmodels sample database contains data of a retail business. The retail business, in this case, is engaged in the sale of models of classic cars. It contains typical business data such as customers, products, product lines (categories of products), orders, order line items (details of an order), payments received, employees, and branch details.

STEP 1. List the current indexes in the relation

Use the following command to list the current indexes in the table:

```
SHOW INDEXES FROM employees;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
employees	1	reportsTo	1	reportsTo	A	7	NULL	NULL	YES	BTREE			YES	NULL
employees	1	officeCode	1	officeCode	A	7	NULL	NULL		BTREE			YES	NULL
employees	0	PRIMARY	1	employeeNumber	A	23	NULL	NULL		BTREE			YES	NULL

Notice that there are only 3 indexes on the columns "reportsTo", "officeCode", and "employeeNumber". Out of these 3 indexes, the one on "employeeNumber" is a clustered index assigned the name "PRIMARY".

STEP 2. Query the database

Execute the following query to get a result set:

```
SELECT
    employeeNumber, lastName, firstName
FROM
    employees
WHERE
    jobTitle = 'Sales Rep';
```

employeeNumber	lastName	firstName
1165	Jennings	Leslie
1166	Thompson	Leslie
1188	Firrelli	Julie
1216	Patterson	Steve
1286	Tseng	Foon Yue
1323	Vanauf	George
1337	Bondur	Loui
1370	Hernandez	Gerard
1401	Castillo	Pamela
1501	Bott	Larry
1504	Jones	Barry
1611	Fixter	Andy
1612	Marsh	Peter

The query searches for employees using the search key as "jobTitle". This is specified in the WHERE clause. However, there is no index on the "jobTitle" attribute as noted in STEP 1. This is a potential cause of slow data retrieval assuming that the job title is a common search

key for most queries issued on the “employees” relation. To address it, we can add an index on the “jobTitle” attribute.

STEP 3. Confirm that the query does not use any index

We can use the EXPLAIN command to identify the indexes that a SELECT statement uses. To use the EXPLAIN command, add the word “EXPLAIN” before the SELECT statement as follows:

```
EXPLAIN SELECT
    employeeNumber, lastName, firstName
FROM
    employees
WHERE
    jobTitle = 'Sales Rep';
```

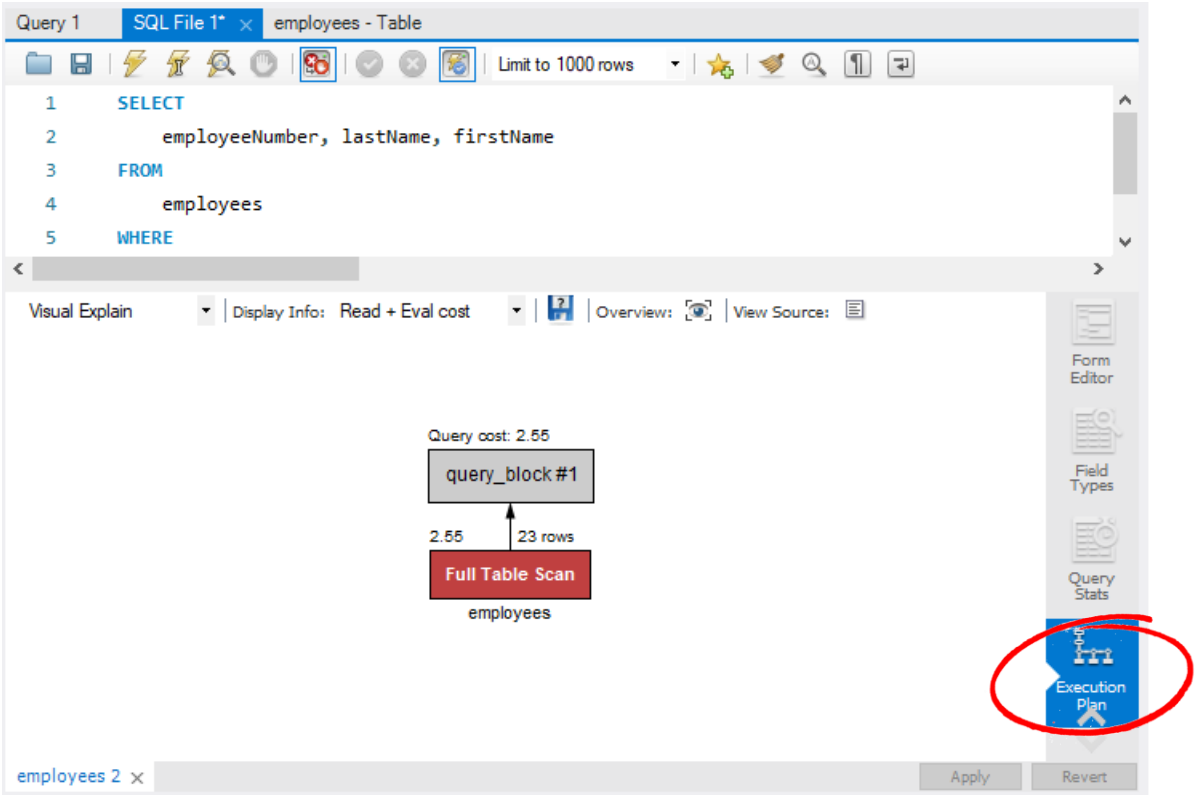
The output shows the following result set:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	employees	NULL	ALL	NULL	NULL	NULL	NULL	23	10.00	Using where

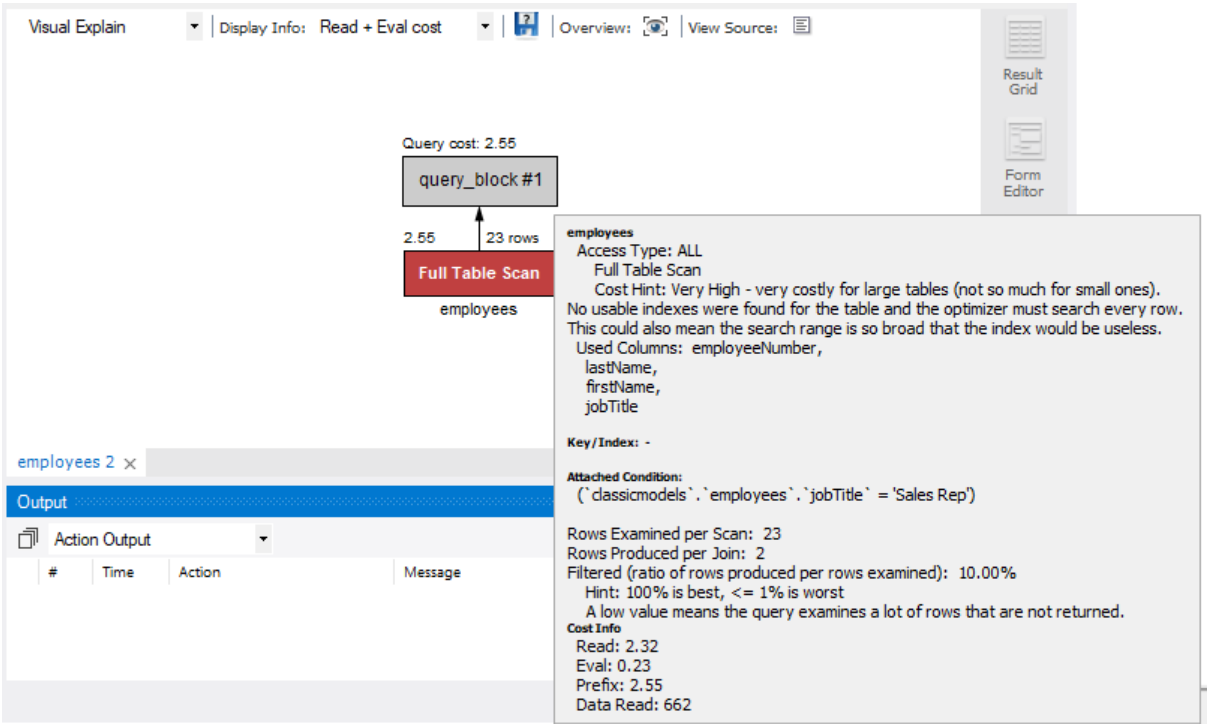
id	Indicates the sequential number of the SELECT statement within the query.
select_type	Indicates the type of SELECT statement. The type can be SIMPLE, which means it does not have any subqueries or it can be PRIMARY, which means it is the outermost query in a nested query. A full list of the select types is available here: https://dev.mysql.com/doc/refman/8.0/en/explain-output.html#explain_select_type
table	Indicates the name of the relation that the output is referring to
partitions	Data in a relation can be grouped into partitions. This attribute indicates the partitions from which records would be matched by the query. The value is NULL for nonpartitioned relations.
type	Indicates how a query combines data from multiple relations using a join. A full list of join types is available here:

	https://dev.mysql.com/doc/refman/8.0/en/explain-output.html#explain-join-types
possible_keys	Indicates the indexes from which the DBMS can choose to find the tuples in the relation. If this attribute is NULL, then there are no relevant indexes for the query to use. In such a case, you may be able to improve the performance of your query by examining the WHERE clause to check whether it refers to some attribute(s) that would be suitable for indexing.
key	Indicates the key (index) that the DBMS decided to use
key_len	Indicates the length of the key that the DBMS decided to use. The value of key_len enables you to determine how many parts of a multiple-part index the DBMS uses.
ref	Indicates whether columns or constants are compared to the index named in the "key" attribute to select tuples from the relation.
rows	Indicates the estimated number of rows the DBMS must examine to answer the query
filtered	Indicates an estimated percentage of tuples that will be filtered by the table condition. The maximum value is 100% which means no filtering of tuples occurred. Values decreasing from 100% indicate increasing amounts of filtering. The "rows" attribute shows the estimated number of tuples examined and "rows" × "filtered" shows the number of tuples that will be included in the result set. For example, if "rows" is 1,000 tuples and "filtered" is 50.00 (50%), the number of tuples to be included in the result set will be 1,000 tuples × 50% = 500 tuples in the result set.
extra	Indicates additional information about how the DBMS resolves the query. A full list of the possible additional information is available here: https://dev.mysql.com/doc/refman/8.0/en/explain-output.html#explain-extra-information

The information in the first diagram of STEP 3 can also be viewed using the graphical user interface of MySQL Workbench. This is done by selecting the "Execution Plan" option in the output section.



The following information is displayed when you hover the mouse on the diagram:



STEP 4. Create an unclustered BTREE index on the "jobTitle" attribute

Execute the following command to create an unclustered BTREE index on the "jobTitle" attribute:

```
CREATE INDEX `IDX_employees_jobTitle` USING BTREE ON
employees (jobTitle ASC);
```

Execute the following command to show the list of indexes associated with the "employees" relation:

```
SHOW INDEXES FROM employees;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
employees	0	PRIMARY	1	employeeNumber	A	23	NULL	NULL		BTREE			YES	NULL
employees	1	reportsTo	1	reportsTo	A	7	NULL	NULL	YES	BTREE			YES	NULL
employees	1	officeCode	1	officeCode	A	7	NULL	NULL		BTREE			YES	NULL
employees	1	IDX_employees_jobTitle	1	jobTitle	A	7	NULL	NULL		BTREE			YES	NULL

Notice that the new index called "IDX_employees_jobTitle" has been added. It is good practice to include a prefix that identifies the type of object created. E.g.

i.) "PK_" represents a primary key

ii.) "FK_" represents a foreign key

iii.) "IDX_" represents an index

iv.) "_UNIQUE" represents a unique constraint (included as a suffix because the prefix would be "IDX" for a "unique index")

v.) "FUNC_" represents a function

vi.) "PROC_" represents a procedure

vii.) "TRG_" represents a trigger

viii.) "EVN_" represents a scheduled event, and so on.

STEP 5. Query the database

Execute the following query again:

```
SELECT
    employeeNumber, lastName, firstName
```

```
FROM
    employees
WHERE
    jobTitle = 'Sales Rep';
```

	employeeNumber	lastName	firstName
▶	1165	Jennings	Leslie
	1166	Thompson	Leslie
	1188	Firrelli	Julie
	1216	Patterson	Steve
	1286	Tseng	Foon Yue
	1323	Vanauf	George
	1337	Bondur	Loui
	1370	Hernandez	Gerard
	1401	Castillo	Pamela
	1501	Bott	Larry
	1504	Jones	Barry
	1611	Fixter	Andy
	1612	Marsh	Peter

Notice that you will still get the same data as you got in STEP 2. However, the difference is in the technique that was used to get the data (HOW the data was retrieved as opposed to WHAT data was retrieved).

STEP 6. Confirm that the query makes use of the created index

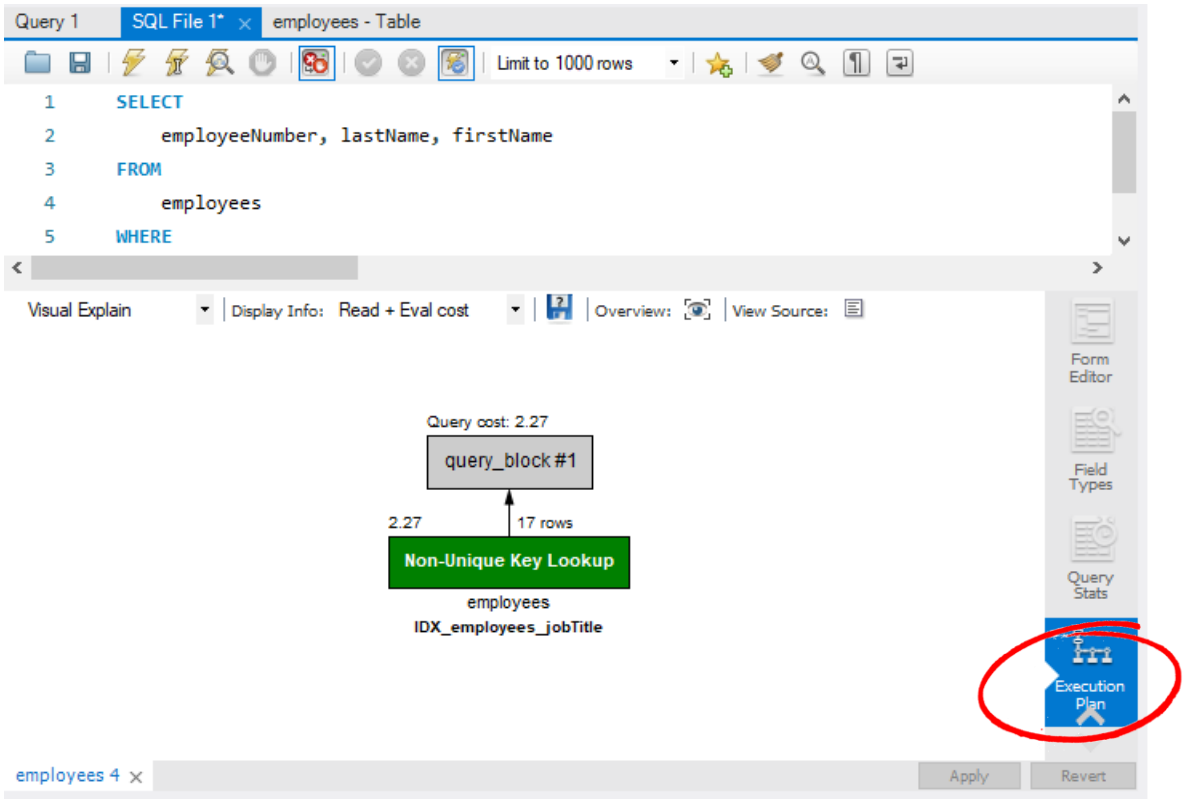
Use the EXPLAIN command to identify the index that the SELECT statement used.

```
EXPLAIN SELECT
    employeeNumber, lastName, firstName
FROM
    employees
WHERE
    jobTitle = 'Sales Rep';
```

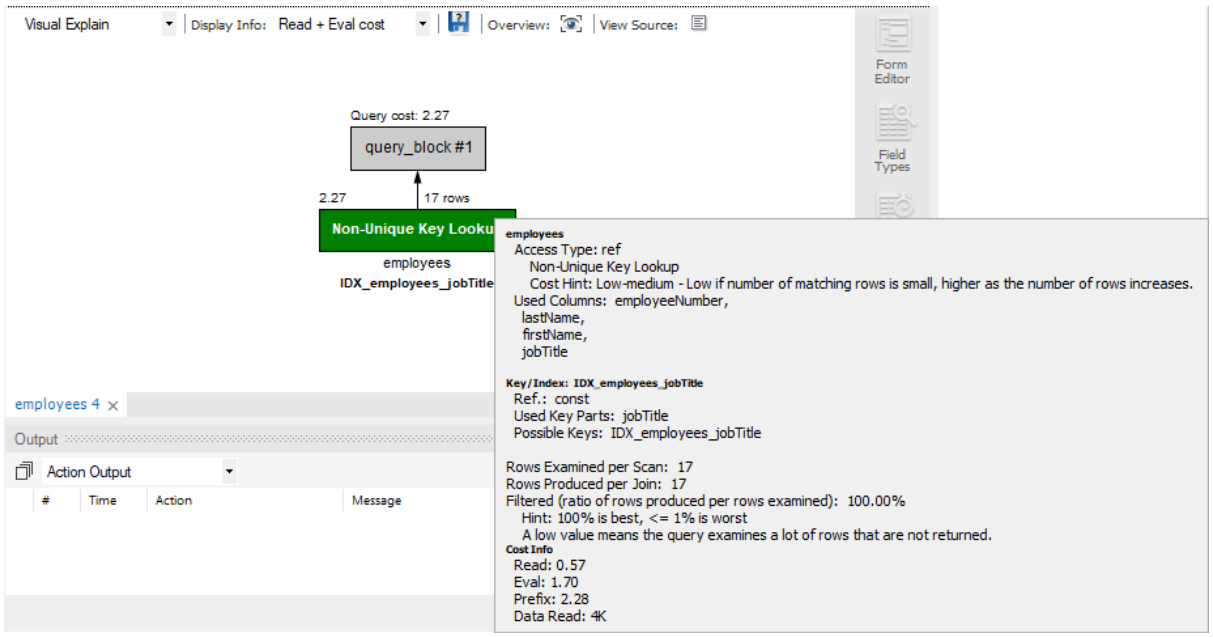
The output shows the following result set:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	employees	NULL	ref	IDX_employees_jobTitle	IDX_employees_jobTitle	52	const	17	100.00	NULL

The information in the first diagram of STEP 6 can also be viewed using the graphical user interface of MySQL Workbench. This is done by selecting the “Execution Plan” option in the output section.



The following information is displayed when you hover the mouse on the diagram:



Notice that 17 tuples were examined to answer the query. This is unlike the previous case in STEP 3 where 23 tuples were examined to answer the query. The index, therefore, made the execution of the query more efficient because the result set was obtained by performing a lower number of scans (reads) for tuples stored in pages on the HDD. The cost of executing the query was 2.55 as shown in STEP 3 but after creating and using the index, the cost reduced to 2.27 as shown in STEP 6.

STEP 7. Deleting an Index

You can execute the following command to delete the index:

```
DROP INDEX `IDX_employees_jobTitle` ON employees;
```

This allows you, if you wish, to go through the lab manual from STEP 1 again.

References

MySQL Index—Ultimate Guide to Indexes in MySQL By Practical Examples. (n.d.). MySQL Tutorial. Retrieved 10 June 2020, from https://www.mysqltutorial.org/mysql-index/