Disciplinary

Database Management System

CS3003

Assignment 3

Sheikh Muhammed Tadeeb (AU19B1014)

❖ Problem Statement:

Apply to index on a set of columns and compare the query performance.

❖ Indexing:

An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view. An index contains keys built from one or more columns in the table or view. These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently.

❖ Types of Indexing:

1) **Primary Indexing: -** Indexing done on primary key of table is called primary indexing.

2) **Secondary Indexing:** - Indexing done on Alternate key of table is called primary indexing.

3) **Cluster Indexing: -** Indexing done on some other column of table apart from primary key where records may have repeating values is called cluster indexing.

4) **Multi-level Indexing: -** Indexing inside indexing is called multi-level indexing. If the data is big machine does this.

❖ Table Insights:

```
MariaDB [assignment3]> desc user_details;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| user_id    | int(11)      | NO   | PRI | NULL    | auto_increment |
| username   | varchar(255) | YES  |     | NULL    |                |
| first_name | varchar(50)  | YES  |     | NULL    |                |
| last_name  | varchar(50)  | YES  |     | NULL    |                |
| gender     | varchar(10)  | YES  |     | NULL    |                |
| password   | varchar(50)  | YES  |     | NULL    |                |
| status     | tinyint(10)  | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
7 rows in set (0.025 sec)
```

❖ Indexing on table:

S.no	Primary Indexing	Secondary Indexing	Cluster Indexing
1.	user_id	username	first_name, last_name, status, password and gender
2.	user_id is the primary key of the table	username is the alternate key of the table i.e. it will also have unique plus not null values but it's not declared as the primary key of the table.	On column first_name, last_name, status, password and gender as these columns may have repeating values for two different entries.

❖ Queries (Before Indexing):

1)

```
MariaDB [assignment3]> select first_name,last_name from user_details where user_id = 77777 ;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| james      | michael   |
+------------+-----------+
1 row in set (0.007 sec)
```

2)

```
MariaDB [assignment3]> select COUNT(*) from user_details where username LIKE '%mike%' ;
+----------+
| COUNT(*) |
+----------+
|     2903 |
+----------+
1 row in set (0.031 sec)
```

3)

```
MariaDB [assignment3]> select count(*) from user_details where first_name = 'John';
+----------+
| count(*) |
+----------+
|     4041 |
+----------+
1 row in set (0.015 sec)
```

4)

```
MariaDB [assignment3]> select count(*) from user_details where last_name = 'bell';
+----------+
| count(*) |
+----------+
|     3663 |
+----------+
1 row in set (0.015 sec)
```

5)

```
MariaDB [assignment3]> select count(*) from user_details where user_id BETWEEN 30 AND 30000;
+----------+
| count(*) |
+----------+
|    29971 |
+----------+
1 row in set (0.016 sec)
```

6)

```
MariaDB [assignment3]> select count(*) from user_details where last_name IN ('david','brown');
+----------+
| count(*) |
+----------+
|     8713 |
+----------+
1 row in set (0.030 sec)
```

7)

```
MariaDB [assignment3]> select count(*) from user_details where gender = 'Female';
+----------+
| count(*) |
+----------+
|    51259 |
+----------+
1 row in set (0.016 sec)
```

Time Chart (before):

Query	Time (milli-second)
1) User_id = 77777	7 ms
2) Username like ‘%mike%’	31 ms
3) First_name = ‘John’	15 ms
4) Last_name = ‘Bell’	15 ms
5) User_id Between 30 AND 30000	16 ms
6) Last_name IN (‘david’,’brown’)	30 ms
7) Gender = ‘Female’	16 ms

So I ran 7 queries and the execution time was all very similar as in order to answer each query database had to scan all 1,00,000 records to check each record for a match.

Queries = 7
Row Scanned = 7,00,000
Total Time taken = 130 ms
Average Speed = Row scanned / total time = <u>5,384.6 rows/ms</u>

While this may seem fast but we can even improve this using indexing.

❖ Indexing on table:

1) Primary indexing:

```
MariaDB [assignment3]> CREATE INDEX user_details_userID_idx ON user_details(user_id);
Query OK, 100000 rows affected (0.166 sec)
Records: 100000  Duplicates: 0  Warnings: 0
```

2) Secondary indexing:

```
MariaDB [assignment3]> CREATE INDEX user_details_username_idx ON user_details(username);
Query OK, 100000 rows affected (0.780 sec)
Records: 100000  Duplicates: 0  Warnings: 0
```

3) Cluster indexing:

```
MariaDB [assignment3]> CREATE INDEX user_details_firstname_idx ON user_details(first_name);
Query OK, 100000 rows affected (0.818 sec)
Records: 100000  Duplicates: 0  Warnings: 0
```

Or

```
MariaDB [assignment3]> CREATE INDEX user_details_lastname_idx ON user_details(last_name);
Query OK, 100000 rows affected (0.855 sec)
Records: 100000  Duplicates: 0  Warnings: 0
```

Or

```
MariaDB [assignment3]> CREATE INDEX user_details_gender_idx ON user_details(gender);
Query OK, 100000 rows affected (0.973 sec)
Records: 100000  Duplicates: 0  Warnings: 0
```

4) Multi-Column Indexing: -

- (first_name, last_name)

```
MariaDB [assignment3]> CREATE INDEX user_details_firstname_lastname_idx ON user_details (first_name,last_name);
Query OK, 100000 rows affected (1.507 sec)
Records: 100000  Duplicates: 0  Warnings: 0
```

- (first_name, last_name)

```
MariaDB [assignment3]> CREATE INDEX user_details_lastname_firstname_idx ON user_details (last_name,first_name);
Query OK, 100000 rows affected (2.119 sec)
Records: 100000  Duplicates: 0  Warnings: 0
```

- (first_name, gender)

```
MariaDB [assignment3]> CREATE INDEX user_details_firstname_gender_idx ON user_details (first_name,gender);
Query OK, 100000 rows affected (2.259 sec)
Records: 100000  Duplicates: 0  Warnings: 0
```

- (gender, first_name)

```
MariaDB [assignment3]> CREATE INDEX user_details_gender_firstname_idx ON user_details (gender,first_name);
Query OK, 100000 rows affected (2.244 sec)
Records: 100000  Duplicates: 0  Warnings: 0
```

NOTE: We can make more pairs of multi-column indexes based on our requirements.

❖ Choosing right "INDEXING"

- **Index check**

```
MariaDB [assignment3]> SHOW INDEXES from user_details;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
user_details	0	PRIMARY	1	user_id	A	100000	NULL	NULL		BTREE		
user_details	1	user_details_userID_idx	1	user_id	A	100000	NULL	NULL		BTREE		
user_details	1	user_details_username_idx	1	username	A	675	NULL	NULL	YES	BTREE		
user_details	1	user_details_firstname_idx	1	first_name	A	23	NULL	NULL	YES	BTREE		
user_details	1	user_details_lastname_idx	1	last_name	A	23	NULL	NULL	YES	BTREE		
user_details	1	user_details_gender_idx	1	gender	A	2	NULL	NULL	YES	BTREE		
user_details	1	user_details_firstname_lastname_idx	1	first_name	A	23	NULL	NULL	YES	BTREE		
user_details	1	user_details_firstname_lastname_idx	2	last_name	A	427	NULL	NULL	YES	BTREE		
user_details	1	user_details_lastname_firstname_idx	1	last_name	A	23	NULL	NULL	YES	BTREE		
user_details	1	user_details_lastname_firstname_idx	2	first_name	A	427	NULL	NULL	YES	BTREE		
user_details	1	user_details_firstname_gender_idx	1	first_name	A	23	NULL	NULL	YES	BTREE		
user_details	1	user_details_firstname_gender_idx	2	gender	A	48	NULL	NULL	YES	BTREE		
user_details	1	user_details_gender_firstname_idx	1	gender	A	2	NULL	NULL	YES	BTREE		
user_details	1	user_details_gender_firstname_idx	2	first_name	A	48	NULL	NULL	YES	BTREE		

```
14 rows in set (0.000 sec)
```

Hence we will choose indexing based on our question. Whatever indexing best suits our needs as we can see in the above table "CARDINALITY i.e. the no. of records or rows is different for each indexing".

❖ Queries (after indexing)

1)

```
MariaDB [assignment3]> select first_name,last_name from user_details where user_id = 77777 ;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| james      | michael   |
+------------+-----------+
1 row in set (0.007 sec)
```

2)

```
MariaDB [assignment3]> select COUNT(*) from user_details where username LIKE '%mike%' ;
+----------+
| COUNT(*) |
+----------+
|     2903 |
+----------+
1 row in set (0.024 sec)
```

3)

```
MariaDB [assignment3]> select COUNT(*) from user_details where first_name = 'John' ;
+----------+
| COUNT(*) |
+----------+
|     4041 |
+----------+
1 row in set (0.008 sec)
```

4)

```
MariaDB [assignment3]> select COUNT(*) from user_details where last_name = 'bell' ;
+----------+
| COUNT(*) |
+----------+
|     3663 |
+----------+
1 row in set (0.007 sec)
```

5)

```
MariaDB [assignment3]> select COUNT(*) from user_details where user_id BETWEEN 30 AND 30000 ;
+----------+
| COUNT(*) |
+----------+
|    29971 |
+----------+
1 row in set (0.013 sec)
```

6)

```
MariaDB [assignment3]> select COUNT(*) from user_details where last_name IN ('david','brown');
+----------+
| COUNT(*) |
+----------+
|     8713 |
+----------+
1 row in set (0.008 sec)
```

7)

```
MariaDB [assignment3]> select COUNT(*) from user_details where gender = 'Female';
+----------+
| COUNT(*) |
+----------+
|    51259 |
+----------+
1 row in set (0.016 sec)
```

Time chart (after):

Query	Time (milli-second)
1) User_id = 77777	7 ms
2) Username like '%mike%'	24 ms
3) First_name = 'John'	8 ms
4) Last_name = 'Bell'	7 ms
5) User_id Between 30 AND 30000	13 ms
6) Last_name IN ('david','brown')	8 ms
7) Gender = 'Female'	16 ms

Queries = 7
Row Scanned = 7,00,000
Total Time taken = 83 ms
Relative Speed = Row scanned / total time = <u>8,433 rows/ms</u>

❖ Queries (SET-2):

```
MariaDB [assignment3]> select COUNT(*) from user_details where first_name = 'bell' AND last_name = 'david';
+----------+
| COUNT(*) |
+----------+
|      254 |
+----------+
1 row in set (0.008 sec)

MariaDB [assignment3]> select COUNT(*) from user_details where first_name = 'bell';
+----------+
| COUNT(*) |
+----------+
|     5683 |
+----------+
1 row in set (0.002 sec)

MariaDB [assignment3]> select COUNT(*) from user_details where first_name = 'bell' AND gender = 'Male';
+----------+
| COUNT(*) |
+----------+
|     2781 |
+----------+
1 row in set (0.008 sec)

MariaDB [assignment3]> select COUNT(*) from user_details where status = 1;
+----------+
| COUNT(*) |
+----------+
|   100000 |
+----------+
1 row in set (0.032 sec)

MariaDB [assignment3]> select COUNT(*) from user_details where first_name = 'bell' AND last_name = 'david' AND gender = 'Male';
+----------+
| COUNT(*) |
+----------+
|        0 |
+----------+
1 row in set (0.009 sec)

MariaDB [assignment3]> select COUNT(*) from user_details where first_name = 'bell' AND last_name = 'david' AND Status = 1;
+----------+
| COUNT(*) |
+----------+
|      254 |
+----------+
1 row in set (0.004 sec)
```

NOTE: As there is no indexing on the status column we can see its time is greater as compared to other queries on which indexing is

❖ Queries (SET-3):

```
MariaDB [assignment3]> select user_id from user_details where first_name = 'bell' AND last_name = 'david' limit 5;
+---------+
| user_id |
+---------+
|      42 |
|      51 |
|     834 |
|     843 |
|    1626 |
+---------+
5 rows in set (0.001 sec)

MariaDB [assignment3]> select first_name, last_name from user_details where user_id BETWEEN 2 AND 8;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| rogers     | paul      |
| david      | john      |
| maria      | sanders   |
| morris     | miller    |
| daniel     | michael   |
| sanders    | paul      |
| mark       | mike      |
+------------+-----------+
7 rows in set (0.000 sec)

MariaDB [assignment3]> select first_name, last_name from user_details where gender = 'Female' limit 5;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| david      | john      |
| morris     | miller    |
| daniel     | michael   |
| sanders    | paul      |
| morgan     | maria     |
+------------+-----------+
5 rows in set (0.001 sec)
```

NOTE: As all the queries are indexed hence the time is very less for executionand overall the process is

❖ Conclusion:

1) The most important use for an index is in finding a record or set of records matching a WHERE clause.

2) Consecutive things are ‘adjacent’ on disk therefore efficient in disk Input & Output hence for a very huge data cluster indexing is better option as compared to primary and secondary indexing.

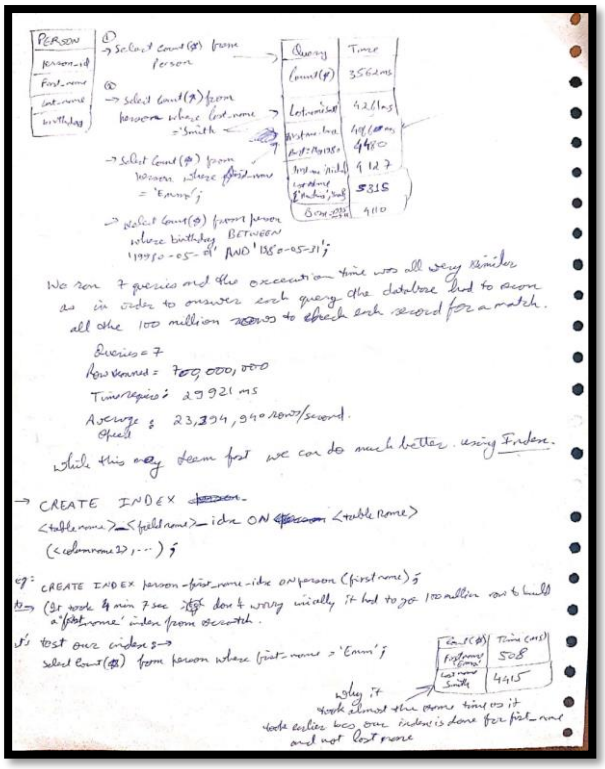
3) WHERE name LIKE ‘%James%’ won’t use indexing.

4) It could fasten Update and Delete command by means of where clause in non-clustered index.

5) When there are no indexes, the database will scan the table and then sort the rows to process the query. However, the index will provide the database with already sorted list of table’s columns. The database can simply scan the index from the first record to the last record and retrieve the rows in sorted order. We can use a GROUP BY clause to group records and aggregate values.

6) If the data is modified on regular intervals then database engine requires updating all the indexes, thus too many indexes will slow down the performance.

❖ Paper Work:



PERSON
person_id
first_name
last_name
birthday

① → Select Count(*) from person

② → select Count(*) from person where last_name = 'Smith'

→ Select Count(*) from person where first_name = 'Emma';

→ Select Count(*) from person where birthday BETWEEN '1990-05-01' AND '1990-05-31';

Query	Time
Count(*)	3562 ms
Lastnames	4261 ms
firstnames	4480
firstnames birthdays	4127
lastnames firstnames birthdays	5315
Count(*)	4110

We ran 7 queries and the execution time was all very similar as in order to answer each query the database had to scan all the 100 million rows to check each record for a match.

Queries = 7
Row Scanned = 700,000,000
Time Required: 29921 ms
Average Speed: 23,394,940 rows/second.

While this may seem fast we can do much better using Indexes.

→ CREATE INDEX
<table name>_<field name>_idx ON <table name>
(<column name 2>, ...);

Q: CREATE INDEX person_first_name_idx ON person (first name);

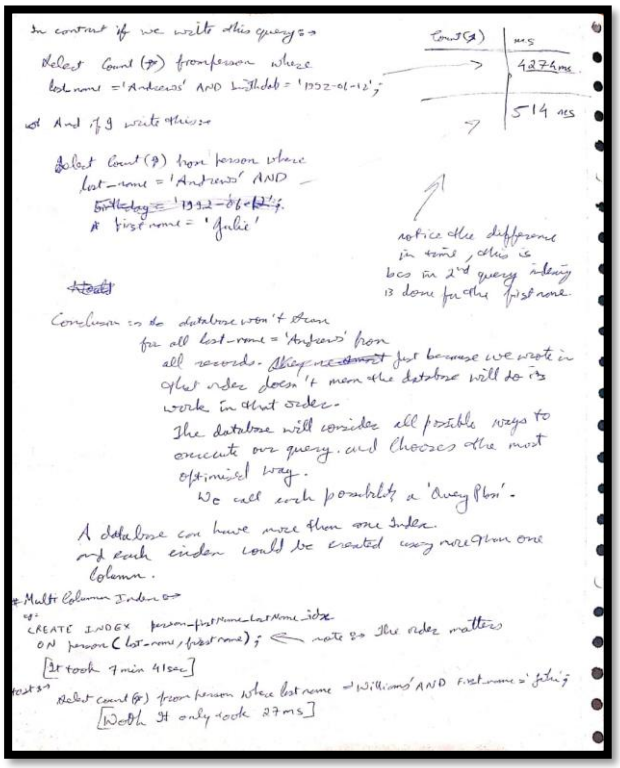
→ (It took 4 min 7 sec but don't worry initially it had to go 100 million rows to build a 'first_name' index from scratch.

Now test our indexes →
Select Count(*) from person where first_name = 'Emma';

Count(*)	Time (ms)
Firstnames (Emma)	508
Last name (Smith)	4415

Why it took almost the same time as it took earlier bcs our index is done for first_name and not last name

①



In contrast if we write this query →

Select Count(*) from person where last_name = 'Andrews' AND birthdate = '1992-01-12';

Count(*)	ms
	4274 ms
	514 ms

Q And if I write this →

Select Count(*) from person where last_name = 'Andrews' AND ~~birthday = '1992-01-12'~~; & first_name = 'Julie'

notice the difference in time, this is bcs in 2nd query indexing is done for the first name.

Conclusion → the database won't scan for all last_name = 'Andrews' from all records. ~~Designer doesn't~~ just because we write in what order doesn't mean the database will do its work in that order.
The database will consider all possible ways to execute our query. and chooses the most optimised way.
We call each possibility a 'Query Plan'.

A database can have more than one index and each index could be created using more than one column.

Multi Column Index →
CREATE INDEX person_firstName_LastName_idx
ON person (last_name, first_name); ← note → the order matters

[It took 7 min 41 sec]

Now test → Select count(*) from person where last_name = 'Williams' AND first_name = 'John';
[Wooh It only took 27 ms]

②