Course: CS3860 Author: Nigel Nelson Date: 10/19/21

Increasing Memory Allotment:

Initial Memory Values:

Memory Values after queries to increase storage:

- Max heap table size value:



Creating tables using InnoBD engine:

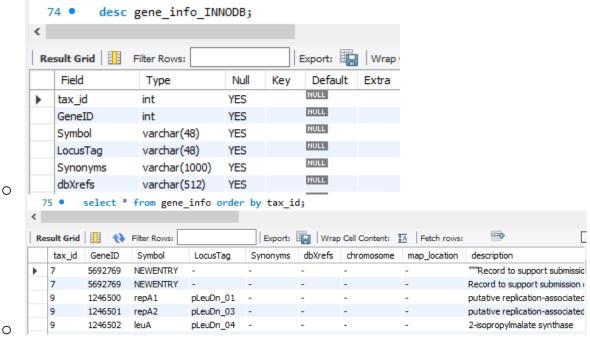
- Loading gene_info5000 data:

```
mysql> Load Data local Infile "C:/Users/nelsonni/OneDrive - Milwaukee School of Engineer ing/Database System/Labs/Lab_6/gene_info50000.csv" into table genomics.gene_info_INNODB fields terminated by ',' lines terminated by '\n' ignore 1 lines;
Query OK, 50000 rows affected (0.74 sec)
Records: 50000 Deleted: 0 Skipped: 0 Warnings: 0
```

- o Records loaded: 50,000
- Verifying gene_info5000 data loaded properly

Lab 6: Indexing

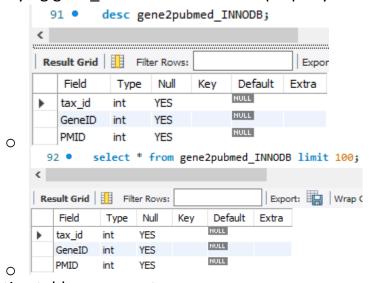
Course: CS3860 Author: Nigel Nelson Date: 10/19/21



Loading gene2pubmed data:

mysql> Load Data local Infile "C:/Users/nelsonni/OneDrive - Milwaukee School of Engineer
ing/Database System/Labs/Lab_6/gene2pubmed" into table genomics.gene2pubmed_INNODB field
s terminated by '\t' lines terminated by '\n' IGNORE 1 LINES;
Query OK, 12917351 rows affected (1 min 52.44 sec)
Records: 12917351 Deleted: 0 Skipped: 0 Warnings: 0

- o Rows loaded: 12,917,351
- Verifying gene_info5000 data loaded properly

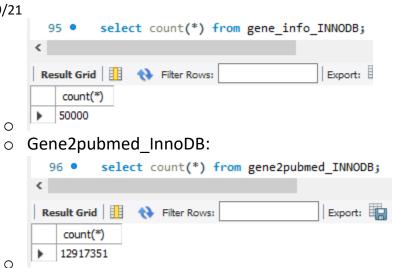


- Noting table row counts:
 - o Gene info InnoDB:

0

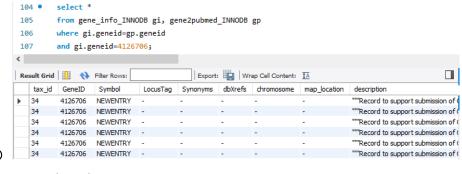
Lab 6: Indexing

Course: CS3860 Author: Nigel Nelson Date: 10/19/21

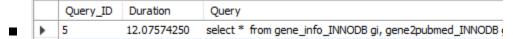


Question 1 - Join

- Joining Tables:



O Execution Time:



Analyzing Query Plan:

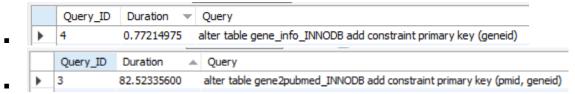
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
•	1	SIMPLE	gi	NULL	ALL	NULL			NULL	49992	10.00	Using where
	1	SIMPLE	αp	NULL	ALL	NULL	NULL	NULL	NULL	12570240	10.00	Using where: Using join buffer (hash join)

- Explanation:
 - This explain statement is showing that for the above join statement, two tables are being joined, where the two are being joined via a hash join. In this join, just under 50k entries from gene_info_InnoDB were iterated through, and just over 1.25 million entries from gene2pubmed_InnoDB. Without any primary

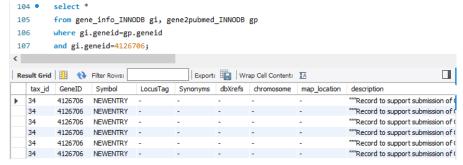
Course: CS3860 Author: Nigel Nelson Date: 10/19/21

keys, SQL has no option but to iterate through just about every element.

- Adding Primary Keys:



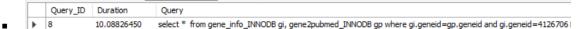
Re-executing join query:



o Execution Time:

0

0



Analyzing Query Plan



- Explanation:
 - The execution of this join is only marginally faster than the previous execution, gaining about a 2 second advantage. The execution plan shows that gene_info_INNODB is able to use its primary key. This allows the query to only attempt to join on a single row for gene_info_INNODB, which is where its geneid attribute equals 4126706. However, gene2pubmed_INNODB is not able to use a primary key on this join because gene2pubmed_INNODB's primary key is a combination key in the order pmid, geneid and the join occurs exclusively on geneid.
- Redefining gene2pubmed_INNODB primary key

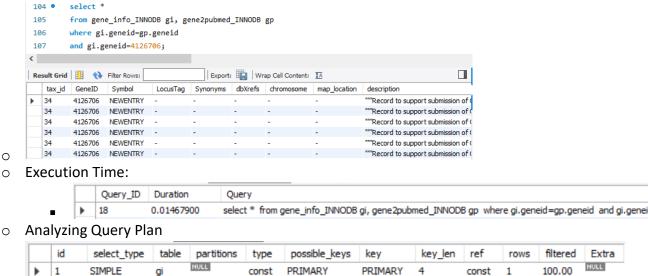
	Query_ID	Duration	Query
•	14	88.22637100	alter table gene2pubmed_INNODB add constraint primary key (geneid, pmid)

- Re-executing join query:

Lab 6: Indexing

Course: CS3860 Author: Nigel Nelson

Date: 10/19/21



Explanation:

 \bigcirc

QD

The execution of this join query is almost 100 times faster than the original query, which originally took about 12 seconds. The execution plan shows that gene info INNODB and gene2pubmed INNODB are able to use their primary key. This allows the query to only attempt to join on a single row for gene info INNODB, which is where its geneid attribute equals 4126706. Also, gene2pubmed INNODB is able to use a primary key on this join and therefore only searches the 6 rows of entries that were referenced by the geneid 4126706. The ability to utilize gene2pubmed INNODB's combination key is likely due to the fact that geneid is now the first portion of the combination key.

PRIMARY

PRIMARY

100.00

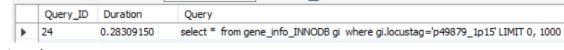
Question 2 - Restrict

Executing select query:

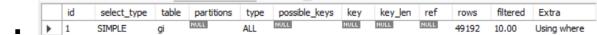


Execution Time:

Course: CS3860 Author: Nigel Nelson Date: 10/19/21

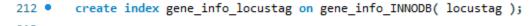


Execution Plan:

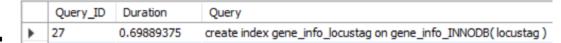


- Explanation:
 - This query filtered through 49,192 rows of entries, no key was used as locustag is not defined as a primary key or index, as such it must iterate through almost all of the entries.
- Creating index for locustag

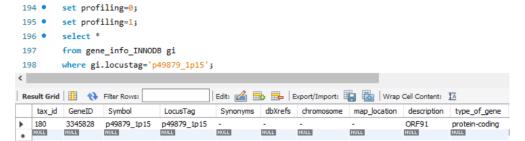
0



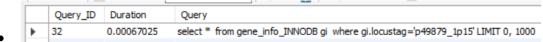
Execution Time:



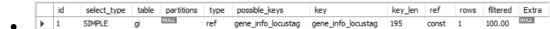
Re-executing select query:



Execution time:



Execution Plan:

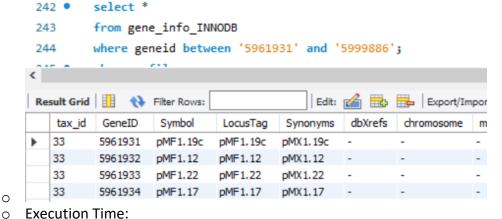


- Explanation:
 - With an index defined, this restrict query is executed in almost a 100th of the original time that it took for the first query. Only a single row needed to be searched, the reason for this that now with an index defined for locustag, sql identifies gene_info_locustag as a key and was able to immediately find the exact entry for the select query.

Question 3 – Range Query

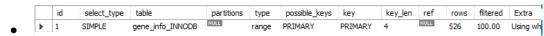
Course: CS3860 Author: Nigel Nelson Date: 10/19/21

Executing select query





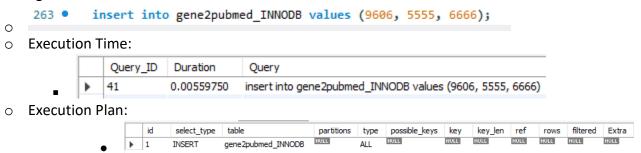
Execution Plan:



- Explanation:
 - 526 rows needed to be searched for this query. This query was able to utilize the primary key, and as such was able to search through less rows of entries.

Question 4 - Insert

Executing insert statement

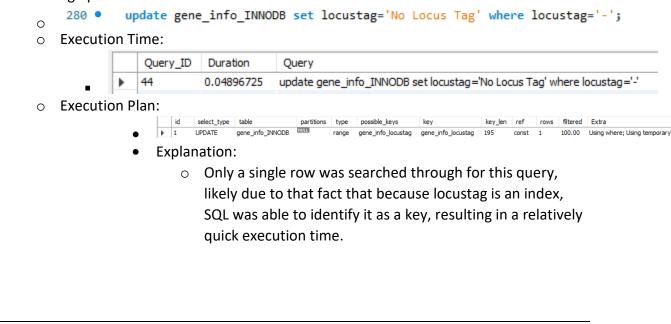


- Explanation:
 - This query is interesting in that it appears that the InnoDB engine was able to insert without utilizing any key, and also able to do it without iterating through any disclosed rows. This is the likely reason for the quick execution time.

Course: CS3860 Author: Nigel Nelson Date: 10/19/21

Question 5 – Update

- Executing update statement



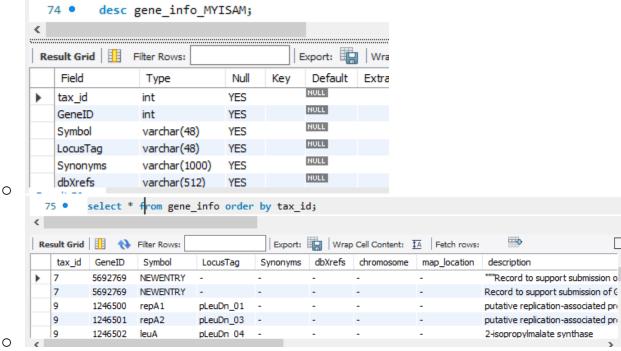
Creating tables using MyISAM engine:

- Loading gene_info5000 data:

```
mysql> Load Data local Infile "C:/Users/nelsonni/OneDrive - Milwaukee School of Engineering/Database System/Labs/Lab_6/gene_info50000.csv" into table genomics.gene_info_MYISAM fields terminated by ',' lines terminated by '\n' ignore 1 lines;
Query OK, 50000 rows affected (0.60 sec)
Records: 50000 Deleted: 0 Skipped: 0 Warnings: 0
```

- Records loaded: 50,000
- Verifying gene_info5000 data loaded properly

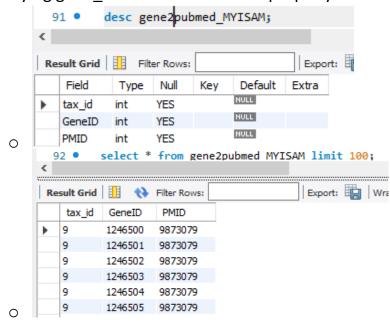
Course: CS3860 Author: Nigel Nelson Date: 10/19/21



Loading gene2pubmed data:

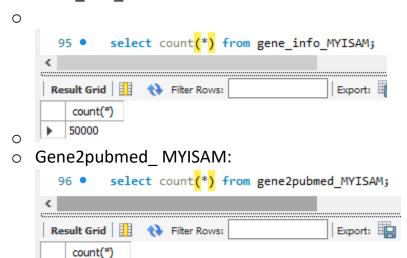
mysql> Load Data local Infile "C:/Users/nelsonni/OneDrive - Milwaukee School of Engineering/Da tabase System/Labs/Lab_6/gene2pubmed" into table genomics.gene2pubmed_MYISAM fields terminated by '\t' lines terminated by '\n' IGNORE 1 LINES;
Query OK, 12917351 rows affected (1 min 31.45 sec)
Records: 12917351 Deleted: 0 Skipped: 0 Warnings: 0

- o Rows loaded: 12,917,351
- Verifying gene_info5000 data loaded properly



Course: CS3860 Author: Nigel Nelson Date: 10/19/21

- Noting table row counts:
 - o Gene info MYISAM:



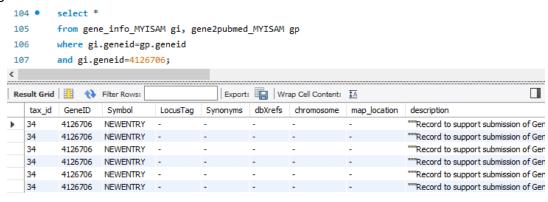
Question 1 - Join

12917351

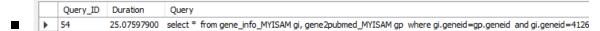
Joining Tables:

0

0



O Execution Time:



Analyzing Query Plan:

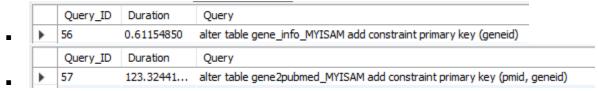
 						_						
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
•	1	SIMPLE	gi	NULL	ALL	NULL	NULL	NULL	NULL	50000	10.00	Using where
	1	SIMPLE	gp	NULL	ALL	NULL	NULL	NULL	NULL	12917351	10.00	Using where; Using join buffer (hash join)

Course: CS3860 Author: Nigel Nelson Date: 10/19/21

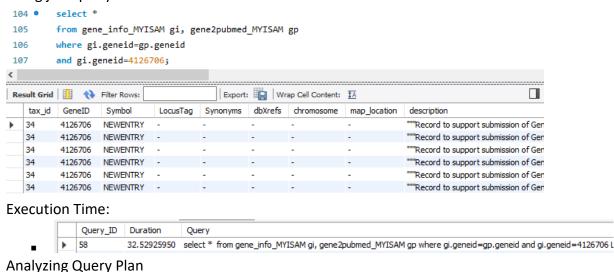
Explanation:

• This explain statement is showing that for the above join statement, two tables are being joined, where the two are being joined via a hash join. In this join, 50k entries from gene_info_InnoDB were iterated through, and just over 1.29 million entries from gene2pubmed_InnoDB. Without any primary keys, SQL has no option but to iterate through just about every element. However, this execution time of 25 seconds is much slower than InnoDB's original execution. This is likely due to the fact that without any indexes or keys defined, MyISAM is only able to use the file that contains the data, and has no way to utilize its index file.

- Adding Primary Keys:



Re-executing join query:



Explanation:

1

1

0

select type

SIMPLE

SIMPLE

partitions

NULL

NULL

αp

type

const

ALL

This execution plan shows that gene_info_MyISAM is able to use its primary key. This allows the query to only attempt to join on a single row for gene_info_INNODB, which is where its geneid attribute equals 4126706. However, gene2pubmed INNODB is not able to use a primary

possible keys

PRIMARY

kev

NULL

PRIMARY

kev len | ref

const

NULL

rows

1

filtered

100.00

12917351 10.00

Extra

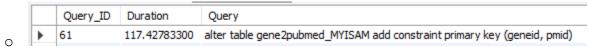
Using where

NULL

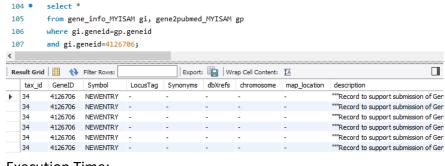
Course: CS3860 Author: Nigel Nelson Date: 10/19/21

key on this join because gene2pubmed_INNODB's primary key is a compound key in the order pmid, geneid and the join occurs exclusively on geneid. This results in 1.29 million rows needed to be iterated through, which is the likely reason for the high execution time of 32 seconds.

Redefining gene2pubmed INNODB primary key

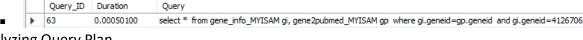


Re-executing join query:

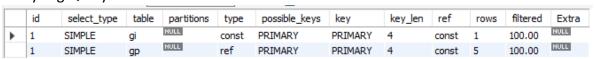


Execution Time:

0



Analyzing Query Plan



- Explanation:
 - This execution plan shows that gene_info_MySAM and gene2pubmed_MySAM are able to use their primary key. This allows the query to only attempt to join on a single row for gene_info_INNODB, which is where its geneid attribute equals 4126706. Also, gene2pubmed_INNODB is able to use a primary key on this join and therefore only searches the 5 rows of entries that were referenced by the geneid 4126706. This is a major improvement in the original execution time which was 22 seconds, resulting in above a 4,000x speed up.

Question 2 - Restrict:

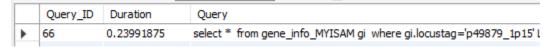
- Executing select query:

Course: CS3860 Author: Nigel Nelson Date: 10/19/21

197 from gene_info_MYISAM gi
198 where gi.locustag='p49879_1p15';



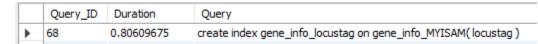
o Execution Time:



o Execution Plan:



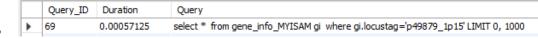
- Explanation:
 - This query filtered through 50k rows of entries, no key was used as locustag is not defined as a primary key or index, as such it must iterate through almost all of the entries. This time is comparable to InnoDB's initial time for execution as well.
- Creating index for locustag
 - 212 create index gene_info_locustag on gene_info_MYISAM(locustag);
 - Execution Time:



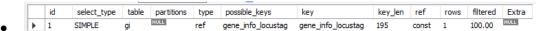
Re-executing select query:



Execution time:



Execution Plan:



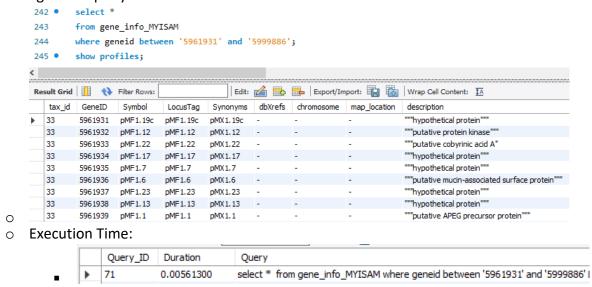
- Explanation:
 - Only a single row needed to be searched, the reason for this that with an index defined, the key

Course: CS3860 Author: Nigel Nelson Date: 10/19/21

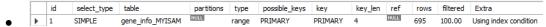
gene_info_locustag was used to immediately find the exact entry for the select query.

Question 3 - Range

- Executing select query



Execution Plan:

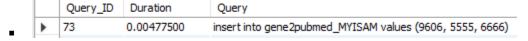


- Explanation:
 - 695 rows needed to be searched for this query, with the utilization of the of the primary key assisting in this smaller search size. However, for the same query in InnoDB, MyISAM must search through over 100 more entries, this is likely due to the fact that InnoBD utilizes a B+ tree while MyISAM uses a data page and an index page.

Question 4 - Insert

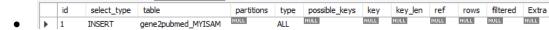
Executing insert statement

- 263 insert into gene2pubmed_MYISAM values (9606, 5555, 6666);
- Execution Time:



Course: CS3860 Author: Nigel Nelson Date: 10/19/21

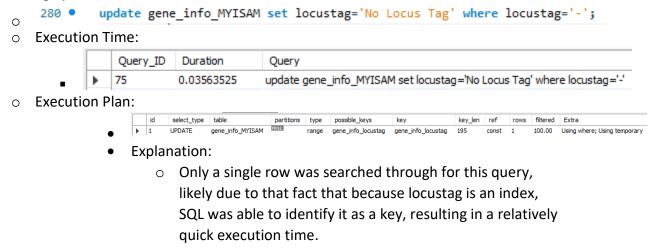
Execution Plan:



- Explanation:
 - This query is interesting in that it appears that the MylSAM engine was able to insert without utilizing any key, and also able to do it without iterating through any disclosed rows.

Question 5 – Update

Executing update statement



Creating tables using Memory engine:

- Loading gene info5000 data:

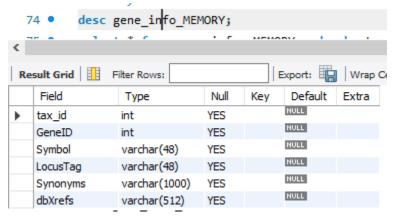
```
mysql> Load Data local Infile "C:/Users/nelsonni/OneDrive - Milwaukee School of Engineering/Da tabase System/Labs/Lab_6/gene_info50000.csv" into table genomics.gene_info_MEMORY fields terminated by ',' lines terminated by '\n' ignore 1 lines;
Query OK, 50000 rows affected (1.65 sec)
Records: 50000 Deleted: 0 Skipped: 0 Warnings: 0
```

- Records loaded: 50,000
- Verifying gene_info5000 data loaded properly

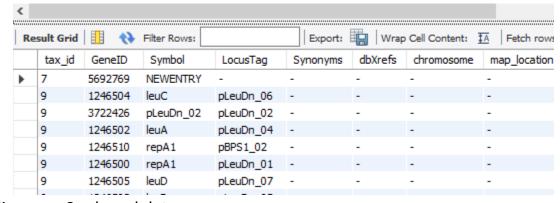
Lab 6: Indexing

Course: CS3860 Author: Nigel Nelson Date: 10/19/21

0



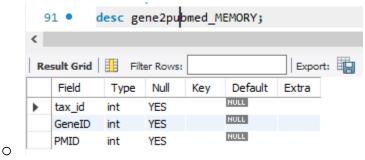
75 • select * from gene_info_MEMORY order by tax_id;



Loading gene2pubmed data:

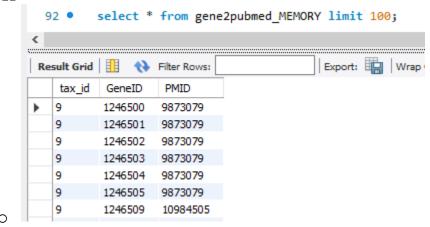
mysql> Load Data local Infile "C:/Users/nelsonni/OneDrive - Milwaukee School of Engineering/Da
tabase System/Labs/Lab_6/gene2pubmed" into table genomics.gene2pubmed_MEMORY fields terminated
by '\t' lines terminated by '\n' IGNORE 1 LINES;
Query OK, 12917351 rows affected (27.72 sec)
Records: 12917351 Deleted: 0 Skipped: 0 Warnings: 0

- o Rows loaded: 12,917,351
- Verifying gene_info5000 data loaded properly

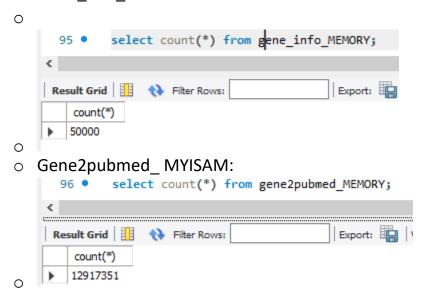


Lab 6: Indexing

Course: CS3860 Author: Nigel Nelson Date: 10/19/21



- Noting table row counts:
 - o Gene_info_MYISAM:

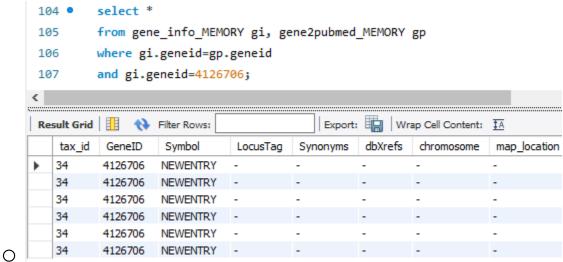


Question 1 - Join

- Joining Tables:

Course: CS3860 Author: Nigel Nelson

Date: 10/19/21



O Execution Time:

	Query_ID	Duration	Query
•	132	0.65525125	select * from gene_info_MEMORY gi, gene2pubmed_MEMORY gp_where gi.geneid=gp.geneid and gi.geneid=4126706

Analyzing Query Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
>	1	SIMPLE	gi	NULL	ALL	NULL	NULL	NULL	NULL	50000	10.00	Using where
	1	SIMPLE	qp	NULL	ALL	NULL	NULL	NULL	NULL	12917351	10.00	Using where; Using join buffer (hash join)

Explanation:

This explain statement is showing that for the above join statement, two tables are being joined, where the two are being joined via a hash join. In this join, 50k entries from gene info InnoDB were iterated through, and just over 1.29 million entries from gene2pubmed InnoDB. Without any primary keys, SQL has no option but to iterate through just about every element. However, this execution time of 0.6553 seconds is much faster than either of the other two engines. This is likely due to the fact that the data is stored on RAM rather than on the disk, which allows much faster lookup times.

Adding Primary Keys:

		Query_ID	Duration	Query
•	•	134	1.66185725	alter table gene_info_MEMORY add constraint primary key (geneid)
		Query_ID	Duration	Query
	•	135	7.33267675	alter table gene2pubmed_MEMORY add constraint primary key (pmid, geneid)

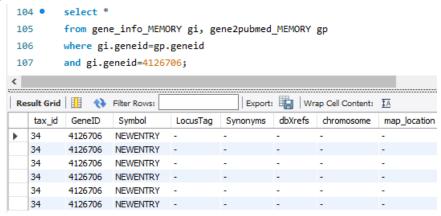
Re-executing join query:

Course: CS3860 Author: Nigel Nelson

 \bigcirc

0

Date: 10/19/21



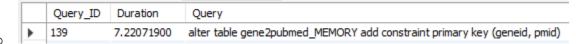
Execution Time:



Analyzing Query Plan



- Explanation:
 - This execution plan shows that gene info Memory is able to use its primary key. This allows the guery to only attempt to join on a single row for gene info Memory, which is where its geneid attribute equals 4126706. However, gene2pubmed Memory is not able to use a primary key on this join because gene2pubmed Memory's primary key is a compound key in the order pmid, geneid and the join occurs exclusively on geneid. This results in 1.29 million rows needed to be iterated through, which is the likely reason for the similar execution time as the initial execution.
- Redefining gene2pubmed INNODB primary key

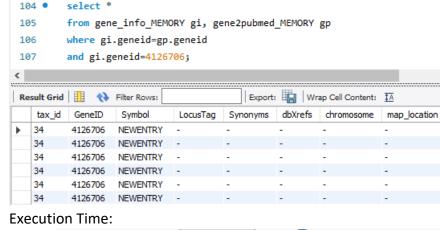


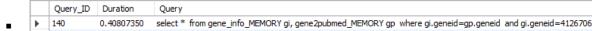
Re-executing join query:

Course: CS3860 Author: Nigel Nelson

 \bigcirc

Date: 10/19/21





Analyzing Query Plan

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
•	1	SIMPLE	gi	NULL	const	PRIMARY	PRIMARY	4	const	1	100.00	NULL
	1	SIMPLE	gp	NULL	ALL	PRIMARY	NULL	NULL	NULL	12917351	10.00	Using where

- **Explanation:**
 - This execution plan shows that gene info Memory is able to use the primary key to only iterate over 1 entry. However, gene2pubmed Memory is still unable to utilize the primary key, and still must iterate over 1.29 million entries, resulting in a comparatively slower execution time to the other storage engines. This is likely an artifact of the RAM storage utilized by the Memory engine.

Question 2 - Restrict

Executing select query:

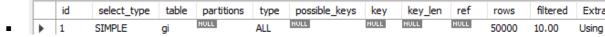


Execution Time:

Course: CS3860 Author: Nigel Nelson Date: 10/19/21



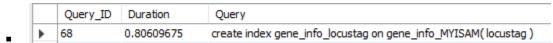
Execution Plan:



- Explanation:
 - This query filtered through 50k rows of entries, no key was used as locustag is not defined as a primary key or index, as such it must iterate through almost all of the entries. This time is comparable to the other storage engine's initial time for execution as well.
- Creating index for locustag
 - create index gene_info_locustag on gene_info_MEMORY(locustag); 212 • 0

p49879_1p15

Execution Time:

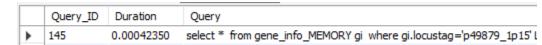


Re-executing select query:



p49879_1p15

Execution time:



chromosome

map_location

description

ORF91

type_

protei

Execution Plan:



Explanation:

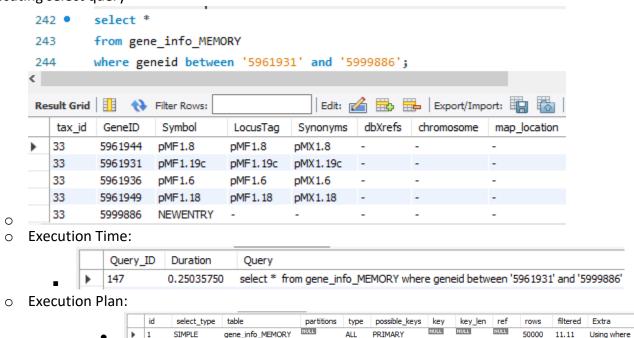
3345828

Only 2 rows were needed to be searched, the reason for this that the key gene info locustag was able to be used as a primary key. However, with the same index, the other engines were able to only check a single entry, which shows a downside for the Memory engine in terms of indexing. Regardless of this, Memory executed this statement faster than the other engines, again likely due to its RAM storage.

Course: CS3860 Author: Nigel Nelson Date: 10/19/21

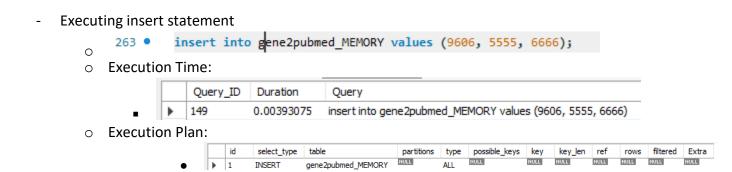
Question 3 - Range

- Executing select query



- Explanation:
 - 50k rows needed to be searched for this query due to the fact that the primary key was unable to be used for this query. This results in the Memory engine taking close to 100 times longer to execute this query when compared to the other two engines.

Question 4 - Insert



Course: CS3860 Author: Nigel Nelson Date: 10/19/21

Explanation:

 This query is interesting in that it appears that the Memory engine was able to insert without utilizing any key, and also able to do it without iterating through any disclosed rows.

Question 5 – Update

- Executing update statement
 - 280 update gene_info_MEMORY set locustag='No Locus Tag' where locustag='-';
 - o Execution Time:



Execution Plan:



- Explanation:
 - 2 rows needed to be iterated through in this query, which again is in contrast to the other two engines that were able to only search through a single row for their identical queries. Interestingly though, the Memory engine has the fastest execution time of the 3 engines, likely due to its utilization of RAM.

Execution Timetable:

	InnoDB	MyISAM	MEMORY
Q1 – join	0.0147 s	0.0005 s	0.4081 s
Q2 - restrict	0.0007 s	0.0006 s	0.0004 s
Q3 – range query	0.0036 s	0.0056 s	0.2504 s
Q4 - insert	0.0056 s	0.0048 s	0.0039 s
Q5 – update	0.0490 s	0.0356 s	0.0171 s

Course: CS3860 Author: Nigel Nelson Date: 10/19/21

Observations:

Overall, it appears that in this suite of experiments, that the MyISAM engine performed the best overall. When joining tables, all but the Memory engine, which stores tables in RAM, seemed to take advantage of the primary keys. As for the restrict query, all engines gain approximately the same efficiency once an index is defined for the restricted attribute. The range query once again was a disadvantage for the Memory engine, showing that it struggled to utilize the primary key to the utmost of its potential. The insert statement also saw similar performance across all engines, likely due to the fact that one of the inserted attributes was a primary key for the tables. Lastly, updating attributes had similar performances across the engines, which was likely due to the fact that this inserted attribute was assigned an index previously. However, a general observation that was noted in these experiments is that if no keys or indexes are defined, the Memory storage engine is likely the best choice. The reason for this is that it was noted that the Memory engine consistently had the fastest execution time when no keys or indexes were used for all storage engines. The likely reason for this is that it stores tables in RAM, instead of on the disk, which allows much faster read and write times. However, if primary keys and indexes are to be used in a database, it is likely not to be the best choice as there were several instances where the Memory engine was unable to take advantage of these indexes to the same degree that the other engines were able to. This contrasts with the MySAM storage engine that was extremely inefficient without any indexes or keys defined, which is likely due to the fact that it stores data in one file and indexes in another. However, with keys and indexes defined, it was the absolute fastest engine to join tables, and consistently the 2nd fastest for all other queries. To summarize, InnoDB appeared to handle all situations to a moderate degree of efficiency, where it did not suffer too bad when keys and indexes were not defined, and gained substantial speedup with keys and indexes. The MyISAM storage engine exceeds when it is supplied keys and indexes to work with, but suffers greatly without the uses of keys and indexes. Lastly, the Memory Engine did extremely well in niche queries where single values were being inserted or selected for, but suffers in comparison to the other engines when keys are defined and multiple values are being queried.

What I learned from the lab:

- The first lesson that I learned from this lab is the ability to change the storage engine used by SQL. Previously, I was unaware that this was a feature that could be changed.

Course: CS3860 Author: Nigel Nelson Date: 10/19/21

Coinciding with this is I learned the importance of specifying the proper storage engine for the problem space. I learned that the InnoDB engine does fairly well in all situations. I also realized that the MyISAM storage engine exceeds with keys and indexes defined, but suffers to a much higher degree than other engines when they are not defined. Lastly, I learned that the Memory engine is the fastest engine to use if keys and indexes are not going to be defined, but if keys and indexes are used, the Memory engine doesn't benefit to the same degree as other engines.