

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

Increasing Memory Allotment:

Initial Memory Values:

- Max_heap_table_size value:

	Variable_name	Value
▶	max_heap_table_size	16777216

- TMP_table_size value:

	Variable_name	Value
▶	tmp_table_size	69206016

Memory Values after queries to increase storage:

- Max_heap_table_size value:

	@@max_heap_table_size
▶	2147483648

- TMP_table_size value:

	@@tmp_table_size
▶	2147483648

Creating tables using InnoDB 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_INNOB
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
```

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

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

74 • desc gene_info_INnoDB;

Result Grid | Filter Rows: | Export: | Wrap

	Field	Type	Null	Key	Default	Extra
▶	tax_id	int	YES		NULL	
	GeneID	int	YES		NULL	
	Symbol	varchar(48)	YES		NULL	
	LocusTag	varchar(48)	YES		NULL	
	Synonyms	varchar(1000)	YES		NULL	
	dbXrefs	varchar(512)	YES		NULL	

75 • select * from gene_info order by tax_id;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description
▶	7	5692769	NEWENTRY	-	-	-	-	-	Record to support submission
	7	5692769	NEWENTRY	-	-	-	-	-	Record to support submission
	9	1246500	repA1	pLeuDn_01	-	-	-	-	putative replication-associated
	9	1246501	repA2	pLeuDn_03	-	-	-	-	putative replication-associated
	9	1246502	leuA	pLeuDn_04	-	-	-	-	2-isopropylmalate synthase

- Loading gene2pubmed data:

```
mysql> Load Data local Infile "C:/Users/nelsonni/OneDrive - Milwaukee School of Engineering/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
```

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

91 • desc gene2pubmed_INnoDB;

Result Grid | Filter Rows: | Export:

	Field	Type	Null	Key	Default	Extra
▶	tax_id	int	YES		NULL	
	GeneID	int	YES		NULL	
	PMID	int	YES		NULL	

92 • select * from gene2pubmed_INnoDB limit 100;

Result Grid | Filter Rows: | Export: | Wrap

	Field	Type	Null	Key	Default	Extra
▶	tax_id	int	YES		NULL	
	GeneID	int	YES		NULL	
	PMID	int	YES		NULL	

- Noting table row counts:
 - Gene_info_InnoDB:
 -

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

```
95 • select count(*) from gene_info_INnoDB;
```

count(*)
50000

-
- Gene2pubmed_InnoDB:

```
96 • select count(*) from gene2pubmed_INnoDB;
```

count(*)
12917351

-

Question 1 - Join

- Joining Tables:

```
104 • select *
105 from gene_info_INnoDB gi, gene2pubmed_INnoDB gp
106 where gi.geneid=gp.geneid
107 and gi.geneid=4126706;
```

tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of

-
- Execution Time:

Query_ID	Duration	Query
5	12.07574250	select * from gene_info_INnoDB gi, gene2pubmed_INnoDB gp

- 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	49992	10.00	Using where
1	SIMPLE	gp	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

Lab 6: Indexing

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:

Query_ID	Duration	Query
4	0.77214975	alter table gene_info_INNODB add constraint primary key (geneid)
Query_ID	Duration	Query
3	82.52335600	alter table gene2pubmed_INNODB add constraint primary key (pmid, geneid)

- Re-executing join query:

```
104 • select *
105 from gene_info_INNODB gi, gene2pubmed_INNODB gp
106 where gi.geneid=gp.geneid
107 and gi.geneid=4126706;
```

tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of

○

- Execution Time:

Query_ID	Duration	Query
8	10.08826450	select * from gene_info_INNODB gi, gene2pubmed_INNODB 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	HULL	const	PRIMARY	PRIMARY	4	const	1	100.00	HULL
1	SIMPLE	gp	HULL	ALL	HULL	HULL	HULL	HULL	12891572	10.00	Using where

○

- 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

```
104 • select *
105 from gene_info_INNODB gi, gene2pubmed_INNODB gp
106 where gi.geneid=gp.geneid
107 and gi.geneid=4126706;
```

Result Grid										Filter Rows:	Export:	Wrap Cell Content:
	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description			
▶	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of			
	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of			
	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of			
	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of			
	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of			
	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of			

- Execution Time:

Query_ID	Duration	Query
18	0.01467900	select * from gene_info_INNODB gi, gene2pubmed_INNODB 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	const	PRIMARY	PRIMARY	4	const	1	100.00	NULL
	1	SIMPLE	gp	NULL	ref	PRIMARY	PRIMARY	4	const	6	100.00	NULL






- Explanation:

- 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.

Question 2 - Restrict

- Executing select query:

```
196 • select *
197 from gene_info_INNODB gi
198 where gi.locustag='p49879_1p15';
```

<div>Result Grid</div> <div>Filter Rows: <input type="text"/></div> <div>Edit:   </div> <div>Export/Import:  </div> <div>Wrap Cell Content: <input checked="" type="checkbox"/></div>										
	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description	type_of_gene
▶	180	3345828	p49879_1p15	p49879_1p15	-	-	-	-	ORF91	protein-coding
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Execution Time:

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

	Query_ID	Duration	Query
▶	24	0.28309150	select * from gene_info_INNOB gi where gi.locustag='p49879_1p15' LIMIT 0, 1000

Execution 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	49192	10.00	Using where

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

```
212 • create index gene_info_locustag on gene_info_INNOB( locustag );
```

Execution Time:

	Query_ID	Duration	Query
▶	27	0.69889375	create index gene_info_locustag on gene_info_INNOB(locustag)

Re-executing select query:

```
194 • set profiling=0;
195 • set profiling=1;
196 • select *
197 from gene_info_INNOB gi
198 where gi.locustag='p49879_1p15';
```

	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description	type_of_gene
▶	180	3345828	p49879_1p15	p49879_1p15	-	-	-	-	ORF91	protein-coding
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Execution time:

	Query_ID	Duration	Query
▶	32	0.00067025	select * from gene_info_INNOB gi where gi.locustag='p49879_1p15' LIMIT 0, 1000

Execution Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	gi	NULL	ref	gene_info_locustag	gene_info_locustag	195	const	1	100.00	NULL

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

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

- Executing select query

```
242 • select *
243 from gene_info_INNOdb
244 where geneid between '5961931' and '5999886';
```

Result Grid								
Filter Rows: <input type="text"/>								
Edit: Export/Import								
	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	m
▶	33	5961931	pMF1.19c	pMF1.19c	pMX1.19c	-	-	-
	33	5961932	pMF1.12	pMF1.12	pMX1.12	-	-	-
	33	5961933	pMF1.22	pMF1.22	pMX1.22	-	-	-
	33	5961934	pMF1.17	pMF1.17	pMX1.17	-	-	-

-
- Execution Time:

	Query_ID	Duration	Query
▶	36	0.00356300	select * from gene_info_INNOdb where geneid between '5961931' and '5999886'

- Execution Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	gene_info_INNOdb	NULL	range	PRIMARY	PRIMARY	4	NULL	526	100.00	Using wh

- 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

```
263 • insert into gene2pubmed_INNOdb values (9606, 5555, 6666);
```

-
- Execution Time:

	Query_ID	Duration	Query
▶	41	0.00559750	insert into gene2pubmed_INNOdb values (9606, 5555, 6666)

- Execution Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	INSERT	gene2pubmed_INNOdb	NULL	ALL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- 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.

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

Question 5 – Update

- Executing update statement

- 280 • `update gene_info_INNODB set locustag='No Locus Tag' where locustag='-';`
- Execution Time:

	Query_ID	Duration	Query
■ ▶	44	0.04896725	update gene_info_INNODB set locustag='No Locus Tag' where locustag='-'

- Execution Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
● ▶	1	UPDATE	gene_info_INNODB	NULL	range	gene_info_locustag	gene_info_locustag	195	const	1	100.00	Using where; Using temporary

- Explanation:

- Only a single row was searched through for this query, likely due to that fact that because locustag is an index, SQL was able to identify it as a key, resulting in a relatively quick execution time.

Creating tables using MyISAM 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_MYISAM fields termi
nated 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

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

74 • desc gene_info_MYISAM;

Field	Type	Null	Key	Default	Extra
tax_id	int	YES		NULL	
GeneID	int	YES		NULL	
Symbol	varchar(48)	YES		NULL	
LocusTag	varchar(48)	YES		NULL	
Synonyms	varchar(1000)	YES		NULL	
dbXrefs	varchar(512)	YES		NULL	

75 • select * from gene_info order by tax_id;

tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description
7	5692769	NEWENTRY	-	-	-	-	-	Record to support submission o
7	5692769	NEWENTRY	-	-	-	-	-	Record to support submission of C
9	1246500	repA1	pLeuDn_01	-	-	-	-	putative replication-associated pr
9	1246501	repA2	pLeuDn_03	-	-	-	-	putative replication-associated pr
9	1246502	leuA	pLeuDn_04	-	-	-	-	2-isopropylmalate synthase

- 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
```

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

91 • desc gene2pubmed_MYISAM;

Field	Type	Null	Key	Default	Extra
tax_id	int	YES		NULL	
GeneID	int	YES		NULL	
PMID	int	YES		NULL	

92 • select * from gene2pubmed MYISAM limit 100;

tax_id	GeneID	PMID
9	1246500	9873079
9	1246501	9873079
9	1246502	9873079
9	1246503	9873079
9	1246504	9873079
9	1246505	9873079

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

- Noting table row counts:

- Gene_info_MYISAM:

○

```
95 • select count(*) from gene_info_MYISAM;
```

count(*)
50000

○

- Gene2pubmed_MYISAM:

```
96 • select count(*) from gene2pubmed_MYISAM;
```

count(*)
12917351



○

Question 1 - Join

- Joining Tables:

```
104 • select *
105 from gene_info_MYISAM gi, gene2pubmed_MYISAM gp
106 where gi.geneid=gp.geneid
107 and gi.geneid=4126706;
```

○

<div>Result Grid</div> <div>Filter Rows: <input type="text"/></div> <div>Export: </div> <div>Wrap Cell Content: </div>									
	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description
▶	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger
	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger
	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger
	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger
	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger
	34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger

○

○ Execution Time:

Query_ID	Duration	Query
54	25.07597900	select * from gene_info_MYISAM gi, gene2pubmed_MYISAM gp where gi.geneid=gp.geneid and gi.geneid=4126

○ 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)

Lab 6: Indexing

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:

	Query_ID	Duration	Query
▶	56	0.61154850	alter table gene_info_MYISAM add constraint primary key (geneid)
	Query_ID	Duration	Query
▶	57	123.32441...	alter table gene2pubmed_MYISAM add constraint primary key (pmid, geneid)

- Re-executing join query:

```
104 • select *
105 from gene_info_MYISAM gi, gene2pubmed_MYISAM gp
106 where gi.geneid=gp.geneid
107 and gi.geneid=4126706;
```

tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description
▶ 34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Gen
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Gen
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Gen
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Gen
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Gen
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Gen

○

○ Execution Time:

	Query_ID	Duration	Query
▶	58	32.52925950	select * from gene_info_MYISAM gi, gene2pubmed_MYISAM 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	const	PRIMARY	PRIMARY	4	const	1	100.00	NULL
	1	SIMPLE	gp	NULL	ALL	NULL	NULL	NULL	NULL	12917351	10.00	Using where

○

○ Explanation:

- 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_INNOBDB, which is where its geneid attribute equals 4126706. However, gene2pubmed_INNOBDB is not able to use a primary

Lab 6: Indexing

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

	Query_ID	Duration	Query
▶	61	117.42783300	alter table gene2pubmed_MYISAM add constraint primary key (geneid, pmid)

- Re-executing join query:

```
104 • select *
105 from gene_info_MYISAM gi, gene2pubmed_MYISAM gp
106 where gi.geneid=gp.geneid
107 and gi.geneid=4126706;
```

tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description
▶ 34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger
34	4126706	NEWENTRY	-	-	-	-	-	""Record to support submission of Ger

- Execution Time:

	Query_ID	Duration	Query
■ ▶	63	0.00050100	select * from gene_info_MYISAM gi, gene2pubmed_MYISAM 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	const	PRIMARY	PRIMARY	4	const	1	100.00	NULL
○	1	SIMPLE	gp	NULL	ref	PRIMARY	PRIMARY	4	const	5	100.00	NULL

- 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:

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

```
196 • select *
197 from gene_info_MYISAM gi
198 where gi.locustag='p49879_1p15';
```

Result Grid											
Filter Rows: [] Edit: [] Export/Import: [] Wrap Cell Content: []											
	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description	type_of_gene	Symbol_fro
▶	180	3345828	p49879_1p15	p49879_1p15	-	-	-	-	ORF91	protein-coding	-
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Execution Time:

	Query_ID	Duration	Query
▶	66	0.23991875	select * from gene_info_MYISAM gi where gi.locustag='p49879_1p15' l

- Execution 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

- 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:

	Query_ID	Duration	Query
▶	68	0.80609675	create index gene_info_locustag on gene_info_MYISAM(locustag)

- Re-executing select query:

```
196 • select *
197 from gene_info_MYISAM gi
198 where gi.locustag='p49879_1p15';
```

Result Grid											
Filter Rows: [] Edit: [] Export/Import: [] Wrap Cell Content: []											
	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description	type_of_gene	Symbol_fro
▶	180	3345828	p49879_1p15	p49879_1p15	-	-	-	-	ORF91	protein-coding	-
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Execution time:

	Query_ID	Duration	Query
▶	69	0.00057125	select * from gene_info_MYISAM gi where gi.locustag='p49879_1p15' LIMIT 0, 1000

- Execution Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	gi	NULL	ref	gene_info_locustag	gene_info_locustag	195	const	1	100.00	NULL

- Explanation:

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

Lab 6: Indexing

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

```
242 • select *
243 from gene_info_MYISAM
244 where geneid between '5961931' and '5999886';
245 • show profiles;
```

tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description
33	5961931	pMF1.19c	pMF1.19c	pMX1.19c	-	-	-	""hypothetical protein""
33	5961932	pMF1.12	pMF1.12	pMX1.12	-	-	-	""putative protein kinase""
33	5961933	pMF1.22	pMF1.22	pMX1.22	-	-	-	""putative cobyrinic acid A"
33	5961934	pMF1.17	pMF1.17	pMX1.17	-	-	-	""hypothetical protein""
33	5961935	pMF1.7	pMF1.7	pMX1.7	-	-	-	""hypothetical protein""
33	5961936	pMF1.6	pMF1.6	pMX1.6	-	-	-	""putative mucin-associated surface protein""
33	5961937	pMF1.23	pMF1.23	pMX1.23	-	-	-	""hypothetical protein""
33	5961938	pMF1.13	pMF1.13	pMX1.13	-	-	-	""hypothetical protein""
33	5961939	pMF1.1	pMF1.1	pMX1.1	-	-	-	""putative APEG precursor protein""

- Execution Time:

Query_ID	Duration	Query
71	0.00561300	select * from gene_info_MYISAM where geneid between '5961931' and '5999886'

- Execution Plan:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	gene_info_MYISAM	NULL	range	PRIMARY	PRIMARY	4	NULL	695	100.00	Using index condition

- 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, MySQL must search through over 100 more entries, this is likely due to the fact that InnoDB utilizes a B+ tree while MySQL 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:

Query_ID	Duration	Query
73	0.00477500	insert into gene2pubmed_MYISAM values (9606, 5555, 6666)

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

- Execution Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	INSERT	gene2pubmed_MYISAM	NULL	ALL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Explanation:

- This query is interesting in that it appears that the MyISAM 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_MYISAM set locustag='No Locus Tag' where locustag='-';`

- Execution Time:

	Query_ID	Duration	Query
▶	75	0.03563525	update gene_info_MYISAM set locustag='No Locus Tag' where locustag='-'

- Execution Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	UPDATE	gene_info_MYISAM	NULL	range	gene_info_locustag	gene_info_locustag	195	const	1	100.00	Using where; Using temporary

- Explanation:

- Only a single row was searched through for this query, likely due to that fact that because locustag is an index, SQL was able to identify it as a key, resulting in a relatively quick execution time.

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 termi
nated 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

```
74 • desc gene_info_MEMORY;
```

Field	Type	Null	Key	Default	Extra
tax_id	int	YES		NULL	
GeneID	int	YES		NULL	
Symbol	varchar(48)	YES		NULL	
LocusTag	varchar(48)	YES		NULL	
Synonyms	varchar(1000)	YES		NULL	
dbXrefs	varchar(512)	YES		NULL	

```
75 • select * from gene_info_MEMORY order by tax_id;
```

tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location
7	5692769	NEWENTRY	-	-	-	-	-
9	1246504	leuC	pLeuDn_06	-	-	-	-
9	3722426	pLeuDn_02	pLeuDn_02	-	-	-	-
9	1246502	leuA	pLeuDn_04	-	-	-	-
9	1246510	repA1	pBPS1_02	-	-	-	-
9	1246500	repA1	pLeuDn_01	-	-	-	-
9	1246505	leuD	pLeuDn_07	-	-	-	-

- 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
```

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

```
91 • desc gene2pubmed_MEMORY;
```

Field	Type	Null	Key	Default	Extra
tax_id	int	YES		NULL	
GeneID	int	YES		NULL	
PMID	int	YES		NULL	

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

```
92 • select * from gene2pubmed_MEMORY limit 100;
```

	tax_id	GeneID	PMID
▶	9	1246500	9873079
	9	1246501	9873079
	9	1246502	9873079
	9	1246503	9873079
	9	1246504	9873079
	9	1246505	9873079
	9	1246509	10984505

- Noting table row counts:

- Gene_info_MYISAM:

```
95 • select count(*) from gene_info_MEMORY;
```

	count(*)
▶	50000

- Gene2pubmed_MYISAM:

```
96 • select count(*) from gene2pubmed_MEMORY;
```

	count(*)
▶	12917351

Question 1 - Join

- Joining Tables:

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

```
104 • select *
105 from gene_info_MEMORY gi, gene2pubmed_MEMORY gp
106 where gi.geneid=gp.geneid
107 and gi.geneid=4126706;
```

Result Grid								
Filter Rows: <input type="text"/> Export: Wrap Cell Content:								
	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location
▶	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-

○

○ 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	gp	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:

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

```
104 • select *
105 from gene_info_MEMORY gi, gene2pubmed_MEMORY gp
106 where gi.geneid=gp.geneid
107 and gi.geneid=4126706;
```

Result Grid								
Filter Rows: <input type="text"/>								
Export: <input type="button" value="Export"/> Wrap Cell Content: <input type="button" value="IA"/>								
	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location
▶	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-

- Execution Time:

	Query_ID	Duration	Query
▶	136	0.42916950	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	const	PRIMARY	PRIMARY	4	const	1	100.00	NULL
	1	SIMPLE	gp	NULL	ALL	NULL	NULL	NULL	NULL	12917351	10.00	Using where

- Explanation:

- This execution plan shows that gene_info_Memory is able to use its primary key. This allows the query 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

	Query_ID	Duration	Query
▶	139	7.22071900	alter table gene2pubmed_MEMORY add constraint primary key (geneid, pmid)

- Re-executing join query:

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

```
104 • select *
105 from gene_info_MEMORY gi, gene2pubmed_MEMORY gp
106 where gi.geneid=gp.geneid
107 and gi.geneid=4126706;
```

Result Grid								
Filter Rows: <input type="text"/> Export: Wrap Cell Content:								
	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location
▶	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-
	34	4126706	NEWENTRY	-	-	-	-	-

-
- Execution Time:

	Query_ID	Duration	Query
■ ▶	140	0.40807350	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	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:

```
196 • select *
197 from gene_info_MEMORY gi
198 where gi.locustag='p49879_1p15';
```

Result Grid										
Filter Rows: <input type="text"/> Edit: Export/Import: Wrap Cell Content:										
	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description	type_of_gene
▶	180	3345828	p49879_1p15	p49879_1p15	-	-	-	-	ORF91	protein-coding
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Execution Time:

Lab 6: Indexing

Course: CS3860

Author: Nigel Nelson

Date: 10/19/21

Query_ID	Duration	Query
142	0.26420650	select * from gene_info_MEMORY gi where gi.locustag='p49879_1p15' LIMIT 0, 1000

Execution 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

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

- 212 • `create index gene_info_locustag on gene_info_MEMORY(locustag);`

Execution Time:

Query_ID	Duration	Query
68	0.80609675	create index gene_info_locustag on gene_info_MYISAM(locustag)

Re-executing select query:

```
196 • select *
197 from gene_info_MEMORY gi
198 where gi.locustag='p49879_1p15';
```

tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location	description	type
180	3345828	p49879_1p15	p49879_1p15	-	-	-	-	ORF91	protein

Execution time:

Query_ID	Duration	Query
145	0.00042350	select * from gene_info_MEMORY gi where gi.locustag='p49879_1p15' l

Execution Plan:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	gi	NULL	ref	gene_info_locustag	gene_info_locustag	195	const	2	100.00	NULL

Explanation:

- 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

```
242 • select *
243 from gene_info_MEMORY
244 where geneid between '5961931' and '5999886';
```

	tax_id	GeneID	Symbol	LocusTag	Synonyms	dbXrefs	chromosome	map_location
▶	33	5961944	pMF1.8	pMF1.8	pMX1.8	-	-	-
	33	5961931	pMF1.19c	pMF1.19c	pMX1.19c	-	-	-
	33	5961936	pMF1.6	pMF1.6	pMX1.6	-	-	-
	33	5961949	pMF1.18	pMF1.18	pMX1.18	-	-	-
	33	5999886	NEWENTRY	-	-	-	-	-

- **Execution Time:**

	Query_ID	Duration	Query
▶	147	0.25035750	select * from gene_info_MEMORY where geneid between '5961931' and '5999886'

- Execution Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	gene_info_MEMORY	NULL	ALL	PRIMARY	NULL	NULL	NULL	50000	11.11	Using where

- 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

- Executing insert statement

```
263 • insert into gene2pubmed_MEMORY values (9606, 5555, 6666);
```

- Execution Time:

	Query_ID	Duration	Query
▶	149	0.00393075	insert into gene2pubmed_MEMORY values (9606, 5555, 6666)

- Execution Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	INSERT	gene2hubmed	MEMORY	NULL	ALL	NULL	NULL	NULL	NULL	NULL	NULL

Lab 6: Indexing

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='-';`
- Execution Time:

	Query_ID	Duration	Query
▶	151	0.01705725	update gene_info_MEMORY set locustag='No Locus Tag' where locustag='-';

- Execution Plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	UPDATE	gene_info_MEMORY	NULL	range	gene_info_locustag	gene_info_locustag	195	const	2	100.00	Using where; Using temporary

- 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

Lab 6: Indexing

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.

Lab 6: Indexing

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.