Cairo University Faculty of Engineering

Database Tuning, Optimization and Performance Analysis

Advanced Database Systems

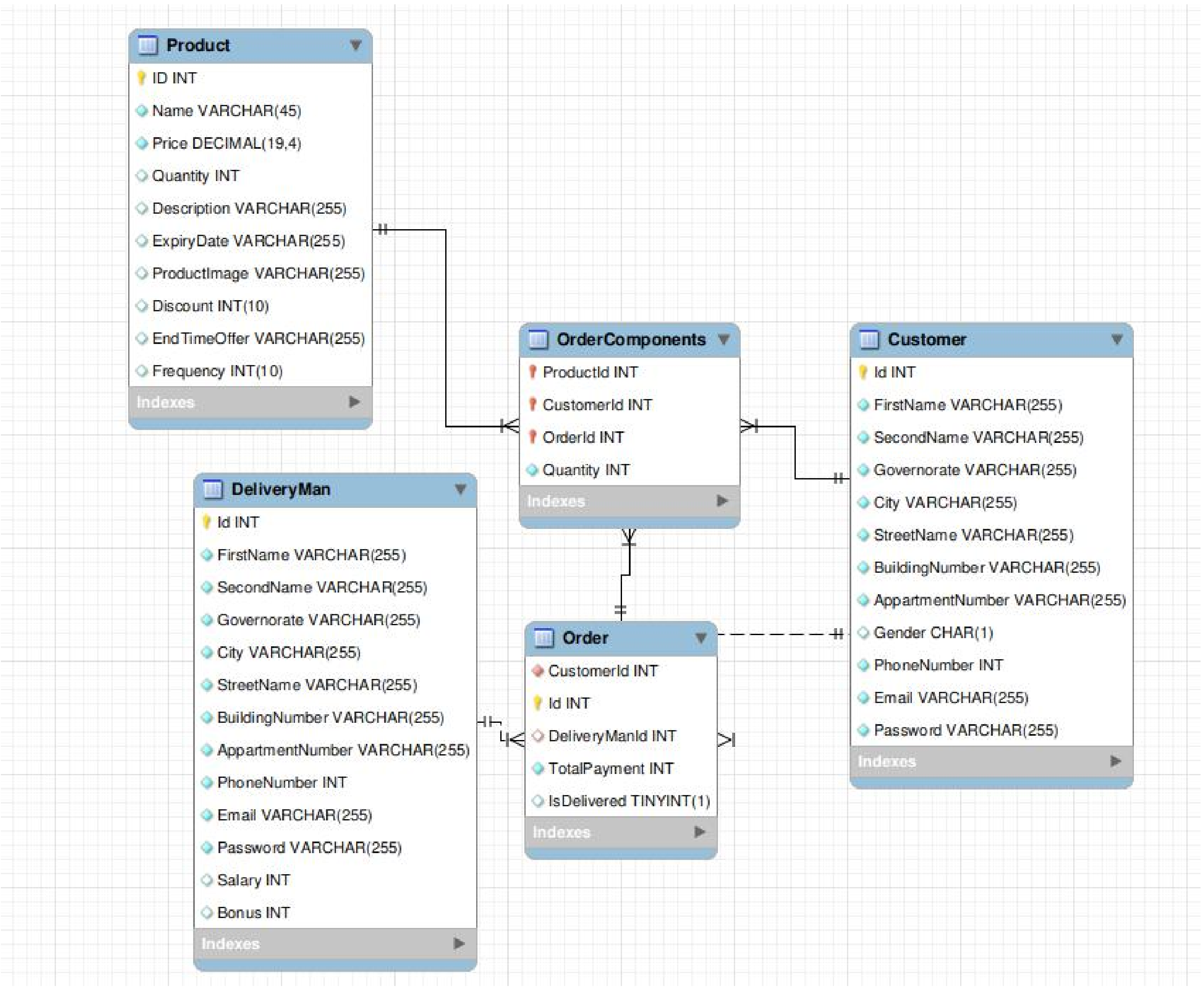
Presented by: Team 11

Passant AbdelAzim Mohamed	Sec: 1	BN: 21
Reem Emad	Sec: 1	BN: 33
Mariam Naser	Sec: 2	BN: 23
Mariam Mohamed Zien	Sec: 2	BN: 22

2023

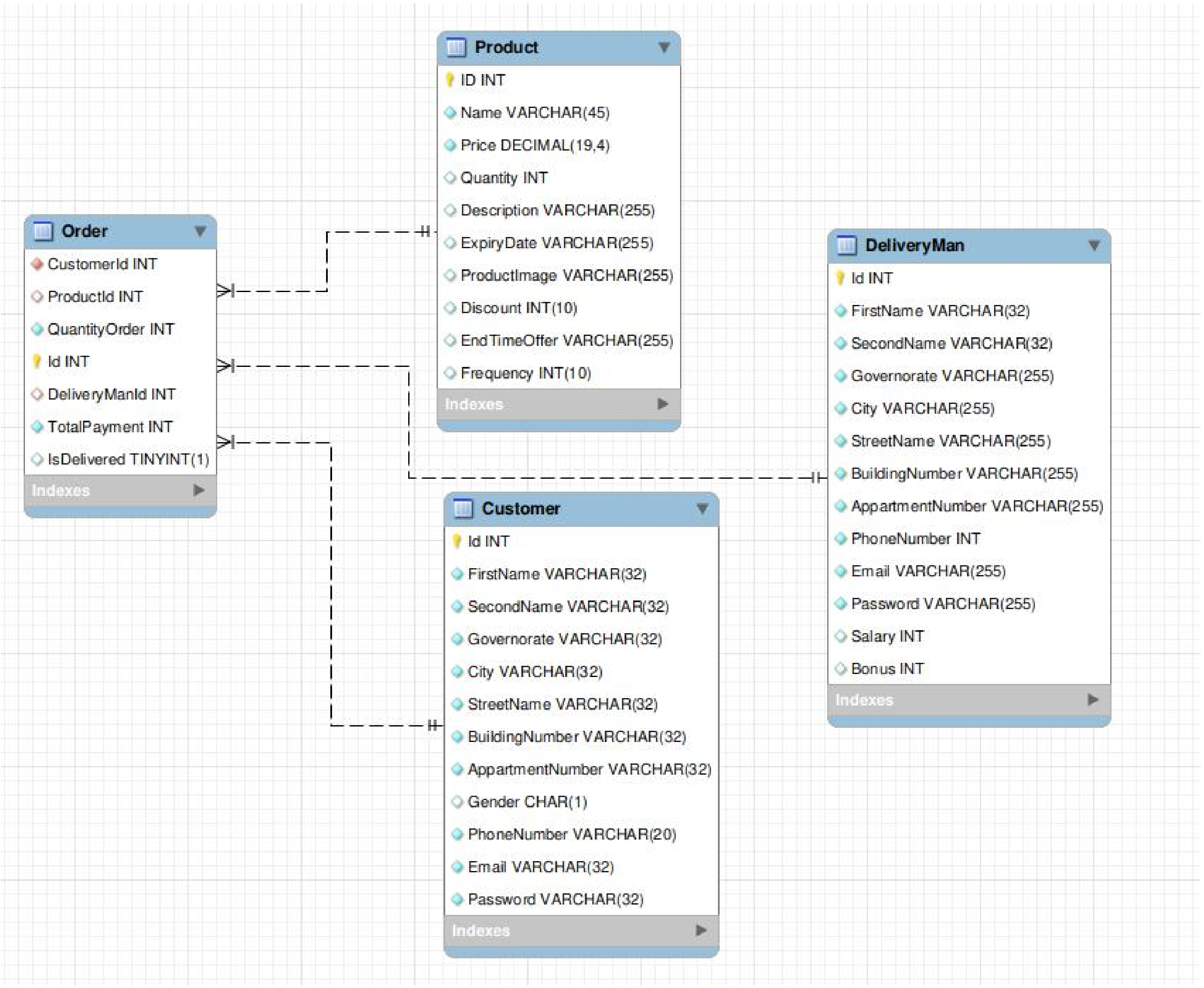
Schema Optimization:

Previous Schema:



We discovered that the OrderComponents table is in no need and can be merged to the Order table and will satisfy the required functionality. So the new schema is:

NewSchema:



DB Number of rows for each table of the new Schema:

Table Name	Number of Rows
Customer	1 millilon record
Product	2 million record
OrderOptimized	5 million record
DeliveryMan	16 thousand record

Set of used queries:

Query One: denoted by Q1

```
explain analyze select c.FirstName, c.SecondName, p.name as
product_name
from `OrderOptimized` as o, `Product` as p, `Customer` as c
where o.ProductId = p.ID  and o.CustomerId = c.Id ;
```

Query Two: denoted by Q2

```
select c.FirstName, c.SecondName
from `OrderOptimized` as o, `Customer` as c  ,`DeliveryMan` as d
where o.CustomerId = c.Id and o.DeliveryManId = d.Id;
```

Query Three: denoted by Q3

```
EXPLAIN ANALYZE select c.FirstName, c.SecondName, p.name as
product_name, d.PhoneNumber as delivery_phone
from `OrderOptimized` as o, `Customer` as c  ,`DeliveryMan` as d,
`Product` as p
where o.CustomerId = c.Id and o.ProductId = p.Id and
o.DeliveryManId = d.Id;
```

Query Four: denoted by Q4

```
select c.city, d.email , d.PhoneNumber
from `OrderOptimized` as o, `Customer` as c ,`DeliveryMan` as d
where c.city = d.city and c.Id = o.CustomerId and o.DeliveryManId
= d.Id;
```

Query Five: denoted by Q5

```
select c.FirstName, c.SecondName
from `OrderOptimized` as o, `Customer` as c  ,`DeliveryMan` as d
where o.CustomerId = c.Id and o.DeliveryManId = d.Id and
o.QuantityOrder = ‘7’
```

Query Optimization

We tried to write a query with nested selects to see its performance before and after query writing optimization. We executed the query on the **10k-database**

Before Optimization:

The query is:

```
select c.FirstName, p.name, d.Email from OrderComponents as oc, Order as o,
Customer as c ,DeliveryMan as d, Product as p where d.FirstName='Andrew' in (select
d.Id from DeliveryMan as d where d.Id in (select o.DeliveryManId from Order as o
where o.Id in (select p.Id from Product as p where p.Id in (select oc.ProductId
from OrderComponents as oc where oc.CustomerId in (select o.CustomerId from
Order as o )))));
```

Result : Ran out of memory after 1 hour

After Optimization:

The query is:

```
select c.FirstName, p.name, d.Email from OrderComponents as oc,
Order as o, Customer as c ,DeliveryMan as d, Product as p
where c.Id in (select o.CustomerId from Order as o where
o.CustomerId in (select oc.CustomerId from OrderComponents as oc
where oc.ProductId in (select p.Id from Product as p
where p.Id in (select o.Id from Order as o where o.DeliveryManId in
(select d.Id from DeliveryMan as d where d.FirstName='Andrew' )))));
```

Result : 36 Sec

Memory and Cache Optimization:

Using Stored Procedure

We investigated the queries we chose and realized that almost all of them use the join between OrderOprimized and Customer tables, so we discussed whether to create a stored procedure or not, and decided that there's no need for it since it will require to store the reslut of the procedure call in a table and use that table in later joins because a procedure call can not be used directly inside other queries.

Changing the block size

At first, we used the default cache memory size value which is **8M**, but then changed the block size to be **16M** and optained the results for the queries as illustrated in the table: to change the cache memorysize we set the system variable innodb_bufer_pool_size

Query Number	Execution Time	
	Before	After
Q3	1.5 hour	31 min

Index Tuning

We added an index in one table as a way of optimization, we realized that the mostly used realtion/table is the order table, so we added a multiple-column index to speed up the query. The index keys are (CustomerId, ProductId, DeliveryManId).

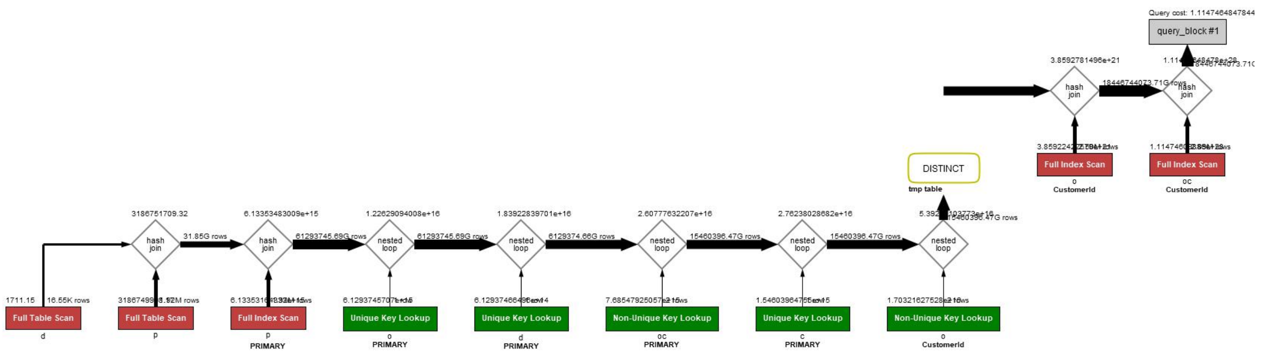
We recorded the execution time after adding the index on the next queries:

Q5

On DB with shown statistics in page 3

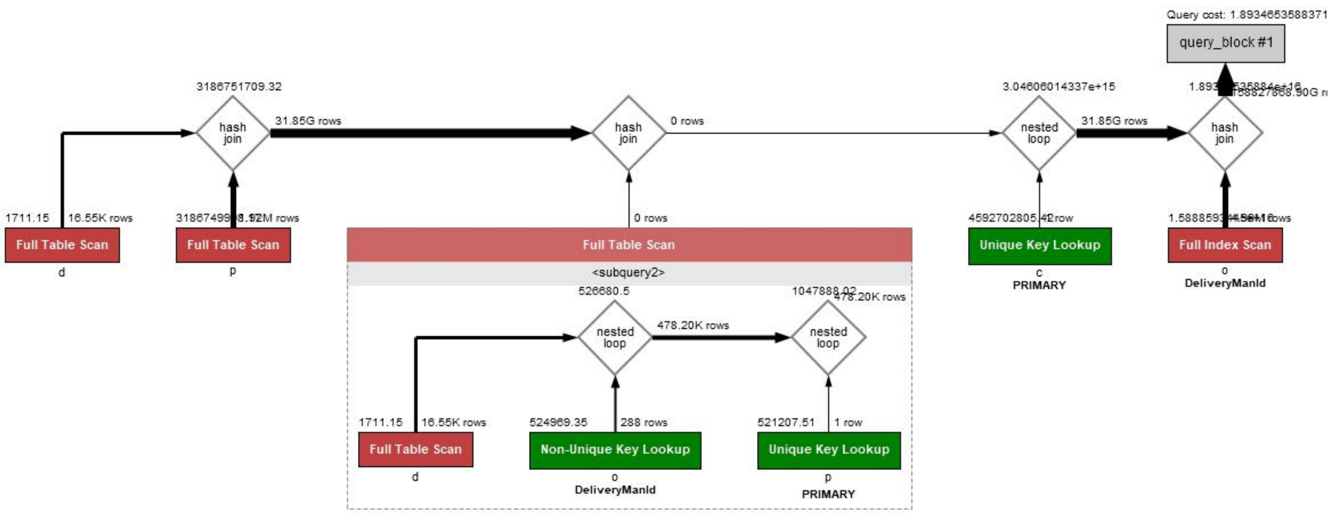
Time		Cost [1 per 8KB]	
Before Any optimization	After Index	Before Any optimization	After Index
30 Sec	30 Sec	1224916.41	1026290.39

Query Tree Before Index



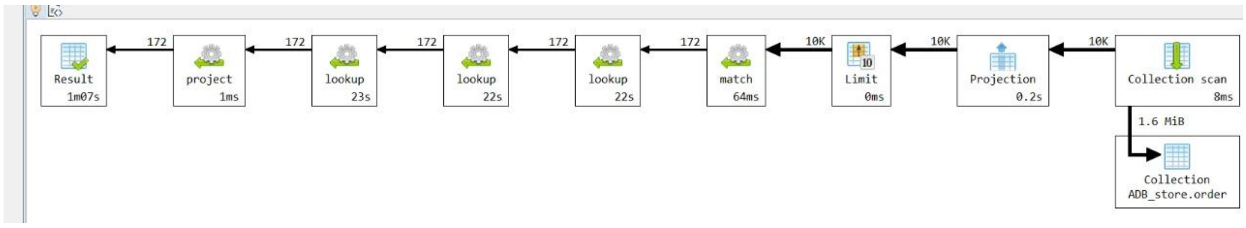
```
-- EXPLAIN -- "-> Nested loop inner join (cost=1026290.39 rows=288799) (actual
time=1.420..26273.064 rows=48421 loops=1) -- -> Nested loop inner join
(cost=925210.60 rows=288799) (actual time=1.407..25855.242 rows=48421
loops=1) -- -> Nested loop inner join (cost=613107.52 rows=288799) (actual
time=1.145..15401.256 rows=51060 loops=1) -- -> Filter: (oc.Quantity = 7)
(cost=299834.31 rows=288799) (actual time=0.875..4227.396 rows=51060
loops=1) -- -> Table scan on oc (cost=299834.31 rows=2887994) (actual
time=0.859..4017.717 rows=3000000 loops=1) -- -> Single-row index lookup on c
using PRIMARY (Id=oc.CustomerId) (cost=0.98 rows=1) (actual time=0.218..0.218
rows=1 loops=51060) -- -> Filter: (o.DeliveryManId is not null) (cost=0.98 rows=1)
(actual time=0.204..0.204 rows=1 loops=51060) -- -> Single-row index lookup on
o using PRIMARY (Id=oc.OrderId) (cost=0.98 rows=1) (actual time=0.203..0.203
rows=1 loops=51060) -- -> Single-row index lookup on d using PRIMARY
```

Query Tree After Index



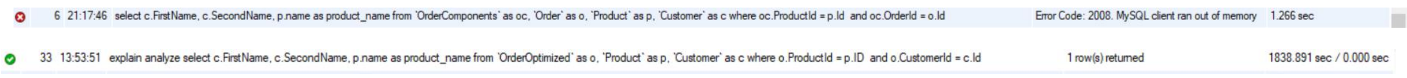
```
-- EXPLAIN -- "-> Nested loop inner join (cost=1224916.41 rows=498692) (actual
time=2.734..37994.760 rows=85411 loops=1) -- -> Nested loop inner join
(cost=687851.19 rows=498692) (actual time=2.368..3964.943 rows=85411
loops=1) -- -> Filter: ((o.QuantityOrder = 7) and (o.DeliveryManId is not null))
(cost=513308.84 rows=498692) (actual time=2.350..3103.431 rows=85411
loops=1) -- -> Table scan on o (cost=513308.84 rows=4986924) (actual
time=1.963..2540.132 rows=5000000 loops=1) -- -> Single-row index lookup on
d using PRIMARY (Id=o.DeliveryManId) (cost=0.25 rows=1) (actual
time=0.009..0.009 rows=1 loops=85411) -- -> Single-row index lookup on c
using PRIMARY (Id=o.CustomerId) (cost=0.98 rows=1) (actual
time=0.398..0.398 rows=1 loops=85411) -- "
```

MongoDB Query Tree



Q1

On DB with shown statistics in page 3



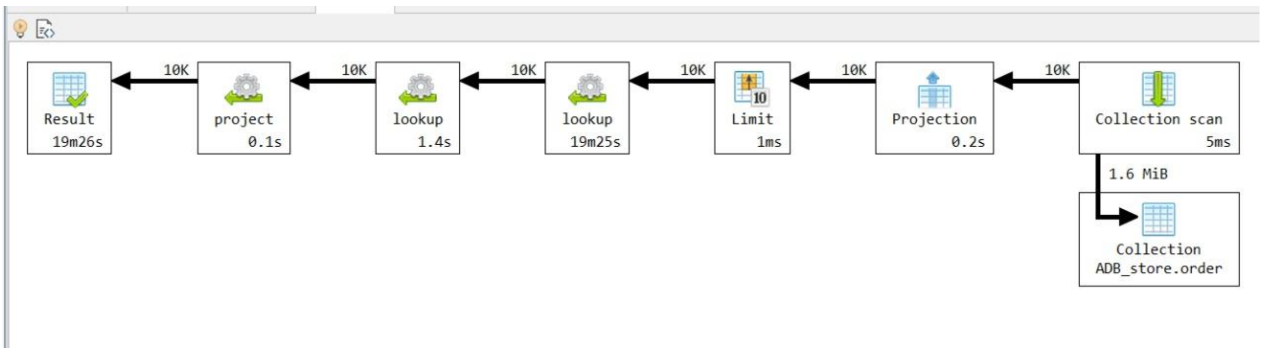
```
6  21:17:46  select c.FirstName, c.SecondName, p.name as product_name from `OrderComponents` as oc, `Order` as o, `Product` as p, `Customer` as c where oc.ProductId = p.Id  and oc.OrderId = o.Id    Error Code: 2008. MySQL client ran out of memory   1.266 sec
33 13:53:51  explain analyze select c.FirstName, c.SecondName, p.name as product_name from `OrderOptimized` as o, `Product` as p, `Customer` as c where o.ProductId = p.ID  and o.CustomerId = c.Id    1 row(s) returned   1838.891 sec / 0.000 sec
```

Time		Cost [1 per 8KB]	
Before Any optimization	After Index	Before Any optimization	After Index
Doesn't run (ran out of memory)	30 min	Doesn't run (ran out of memory)	7062568.70

Query Tree with cost

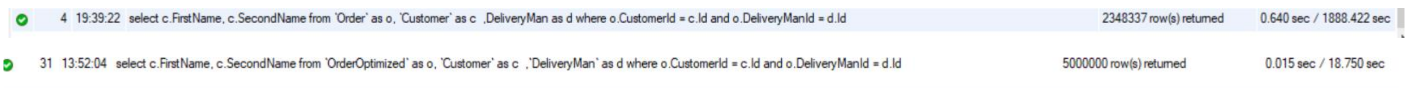
```
-- EXPLAIN ANALYZE AFTER INDEX
-- "-> Nested loop inner join  (cost=7062568.70 rows=4931772) (actual time=19.547..1831656.841 rows=5000000 loops=1)
--     -> Nested loop inner join  (cost=1637748.78 rows=4931772) (actual time=17.648..43446.913 rows=5000000 loops=1)
--         -> Table scan on c  (cost=113277.12 rows=1022722) (actual time=16.446..7941.466 rows=1033612 loops=1)
--         -> Filter: (o.ProductId is not null)  (cost=1.01 rows=5) (actual time=0.017..0.033 rows=5 loops=1033612)
--             -> Index lookup on o using ordering_index_FK (CustomerId=c.Id)  (cost=1.01 rows=5) (actual time=0.016..0.029 rows=5 loops=1033612)
--     -> Single-row index lookup on p using PRIMARY (ID=o.ProductId)  (cost=1.00 rows=1) (actual time=0.357..0.357 rows=1 loops=5000000)
-- "
```

MongoDB Query Tree



Q2

On DB with shown statistics in page 3



```
4  19:39:22  select c.FirstName, c.SecondName from `Order` as o, `Customer` as c  ,DeliveryMan as d where o.CustomerId = c.Id and o.DeliveryManId = d.Id    2348337 row(s) returned    0.640 sec / 1888.422 sec
31  13:52:04  select c.FirstName, c.SecondName from `OrderOptimized` as o, `Customer` as c  ,`DeliveryMan` as d where o.CustomerId = c.Id and o.DeliveryManId = d.Id    5000000 row(s) returned    0.015 sec / 18.750 sec
```

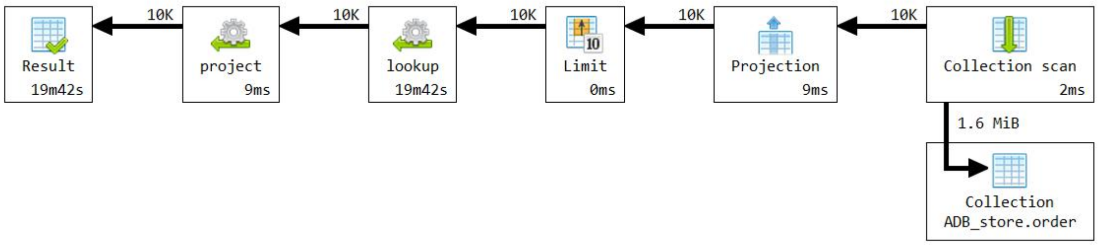
Time		Cost [1 per 8KB]	
Before Any optimization	After Index	Before Any optimization	After Index
31 min	20 Sec	5014060.45	3363868.91

Q2 Query Tree with cost

```
-- EXPLAIN ANALYZE BEFORE INDEX`
-> Nested loop inner join (cost=5014060.45 rows=2495465) (actual time=3.694..655541.403 rows=2348337 loops=1) -- -> Nested loop inner join (cost=2281460.30 rows=2495465) (actual time=2.662..123760.391 rows=2348337 loops=1) -- -> Filter: (o.DeliveryManId is not null) (cost=255536.74 rows=2495465) (actual time=1.833..6759.057 rows=2348337 loops=1) -- -> Table scan on o (cost=255536.74 rows=2495465) (actual time=1.831..5724.464 rows=2500000 loops=1) -- -> Single-row index lookup on d using PRIMARY (Id=o.DeliveryManId) (cost=0.71 rows=1) (actual time=0.049..0.049 rows=1 loops=2348337) -- -> Single-row index lookup on c using PRIMARY (Id=o.CustomerId) (cost=1.00 rows=1) (actual time=0.226..0.226 rows=1 loops=2348337) --
```

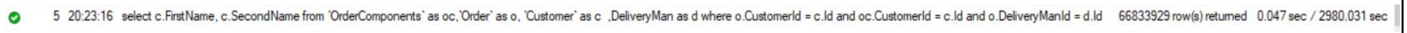
```
-- EXPLAIN ANALYZE AFTER INDEX
-- "-> Nested loop inner join  (cost=3363868.91 rows=4931772) (actual time=4.653..21522.098 rows=5000000 loops=1)
--     -> Nested loop inner join  (cost=1637748.78 rows=4931772) (actual time=4.640..10064.638 rows=5000000 loops=1)
--         -> Table scan on c  (cost=113277.12 rows=1022722) (actual time=3.957..1633.932 rows=1033612 loops=1)
--         -> Filter: (o.DeliveryManId is not null)  (cost=1.01 rows=5) (actual time=0.005..0.008 rows=5 loops=1033612)
--             -> Index lookup on o using ordering_index_FK (CustomerId=c.Id)  (cost=1.01 rows=5) (actual time=0.005..0.007 rows=5 loops=1033612)
--     -> Single-row index lookup on d using PRIMARY (Id=o.DeliveryManId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=5000000)
-- "
```

MongoDB Query Tree



Q3

On DB with shown statistics in page 3



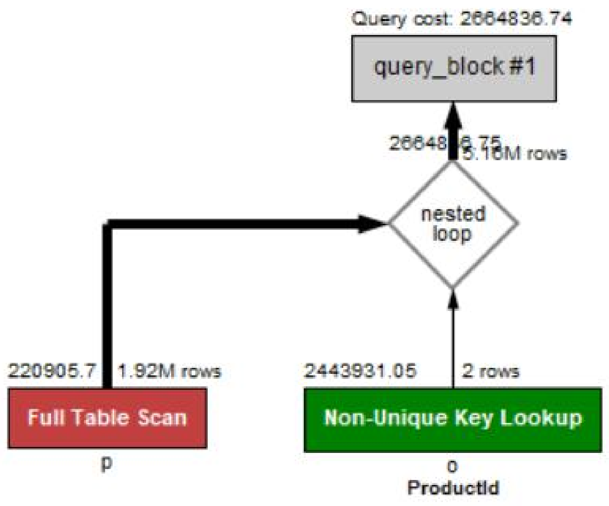
```
5  20:23:16  select c.FirstName, c.SecondName from `OrderComponents` as oc,`Order` as o, `Customer` as c  ,DeliveryMan as d where o.CustomerId = c.Id and oc.CustomerId = c.Id and o.DeliveryManId = d.Id    66833929 row(s) returned   0.047 sec / 2980.031 sec
```

Time		Cost [1 per 8KB]	
Before Any optimization	After Index	Before Any optimization	After Index
1.5 hour	45 min	7430656.76	2664830.74

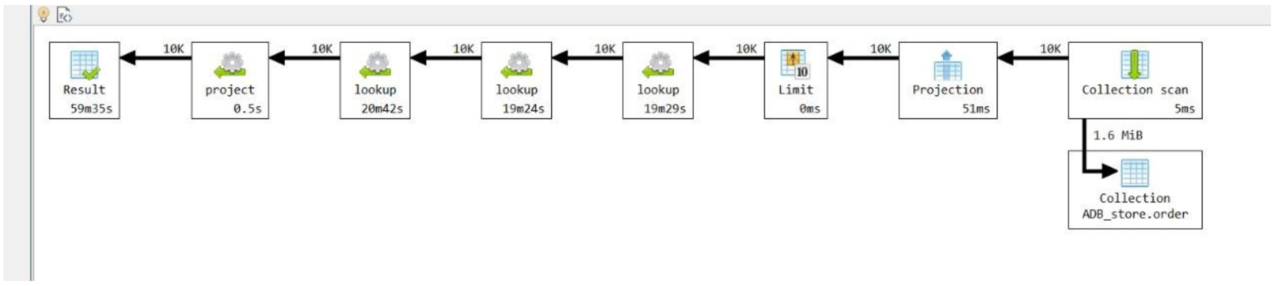
Query Tree before Index

```
-- EXPLAIN ANALYZE BEFORE INDEX
-- "-> Nested loop inner join (cost=7430656.76 rows=8972283) (actual
time=3.202..2076721.536 rows=66833928 loops=1) -- -> Nested loop inner join
(cost=4025762.89 rows=2495465) (actual time=1.858..1089860.555 rows=2348337 loops=1) --
-> Nested loop inner join (cost=1345659.42 rows=2495465) (actual time=1.451..378221.979
rows=2348337 loops=1) -- -> Filter: (o.DeliveryManId is not null) (cost=255535.23
rows=2495465) (actual time=1.359..10810.925 rows=2348337 loops=1) -- -> Table scan on o
(cost=255535.23 rows=2495465) (actual time=1.356..8885.824 rows=2500000 loops=1) -- ->
Single-row index lookup on d using PRIMARY (Id=o.DeliveryManId) (cost=0.34 rows=1)
(actual time=0.156..0.156 rows=1 loops=2348337) -- -> Single-row index lookup on c using
PRIMARY (Id=o.CustomerId) (cost=0.97 rows=1) (actual time=0.302..0.302 rows=1
loops=2348337) -- -> Index lookup on oc using CustomerId (CustomerId=o.CustomerId)
(cost=1.00 rows=4) (actual time=0.387..0.415 rows=28 loops=2348337) --
```

Query Tree After Index

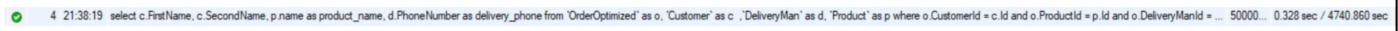


MongoDB Query Tree



Q4

On DB with shown statistics in page 3



```
4  21:38:19  select c.FirstName, c.SecondName, p.name as product_name, d.PhoneNumber as delivery_phone from `OrderOptimized` as o, `Customer` as c ,`DeliveryMan` as d, `Product` as p where o.CustomerId = c.Id and o.ProductId = p.Id and o.DeliveryManId = ...  50000...  0.328 sec / 4740.860 sec
```

Time		Cost [1 per 8KB]	
Before Any optimization	After Index	Before Any optimization	After Index
1.3 hour	26 Sec	12312434.72	6887614.81

Query Tree

```
-- EXPLAIN ANALYZE AFTER INDEX
-- "-> Nested loop inner join  (cost=6887614.81 rows=493177) (actual
time=186.090..29004.538 rows=5 loops=1)
--     -> Nested loop inner join  (cost=1637873.51 rows=4931772) (actual
time=2.771..12376.398 rows=5000000 loops=1)
--         -> Table scan on c  (cost=113401.86 rows=1022722) (actual time=1.579..1747.596
rows=1033612 loops=1)
--         -> Filter: (o.DeliveryManId is not null)  (cost=1.01 rows=5) (actual time=0.006..0.010
rows=5 loops=1033612)
--             -> Index lookup on o using ordering_index_FK (CustomerId=c.Id)  (cost=1.01
rows=5) (actual time=0.006..0.009 rows=5 loops=1033612)
--     -> Filter: (d.City = c.City)  (cost=0.96 rows=0) (actual time=0.003..0.003 rows=0
loops=5000000)
--         -> Single-row index lookup on d using PRIMARY (Id=o.DeliveryManId)  (cost=0.96
rows=1) (actual time=0.003..0.003 rows=1 loops=5000000)
```

```
EXPLAIN ANALYZE BEFORE INDEX
-- -> Nested loop inner join  (cost=12312434.72 rows=4931772) (actual
time=5.732..2741420.522 rows=5000000 loops=1)
--     -> Nested loop inner join  (cost=6887614.81 rows=4931772) (actual
time=4.944..567011.570 rows=5000000 loops=1)
--         -> Nested loop inner join  (cost=1637873.51 rows=4931772) (actual
time=4.391..68941.898 rows=5000000 loops=1)
--             -> Table scan on c  (cost=113401.86 rows=1022722) (actual time=3.797..25439.810
rows=1033612 loops=1)
--             -> Filter: ((o.DeliveryManId is not null) and (o.ProductId is not null))  (cost=1.01
rows=5) (actual time=0.021..0.040 rows=5 loops=1033612)
--                 -> Index lookup on o using ordering_index_FK (CustomerId=c.Id)  (cost=1.01
rows=5) (actual time=0.019..0.033 rows=5 loops=1033612)
--         -> Single-row index lookup on d using PRIMARY (Id=o.DeliveryManId)  (cost=0.96
rows=1) (actual time=0.099..0.099 rows=1 loops=5000000)
--     -> Single-row index lookup on p using PRIMARY (ID=o.ProductId)  (cost=1.00 rows=1)
(actual time=0.434..0.434 rows=1 loops=5000000)
```

Queries

Query	Time			Cost (For 1M database)
	10K	30K/100K	1M	
Q1 MySQL Before Optimization	52 Sec	Out of memory	Out of memory	ran out of memory
Q1 MySQL After schema Optimization & Index	0.047 Sec	27 Sec	30 min	7062568.70
Q1 MongoDB	19 min	66 min / fast 18 Sec	-------	
Q2 MySQL Before Optimization	0.047 Sec	5 Sec	31 min	5014060.45
Q2 MySQL After Optimization & Index	0.031 Sec	22 Sec	20 Sec	3363868.91
Q2 MongoDB	19 min	55 min / fast 19 Sec	-----	---------
Q3 MySQL Before Optimization	0.234Sec	25 Sec	1.5 hour	7430656.76
Q3 MySQL After Optimization & Index	0.031 Sec	22 Sec	45 min	2664830.74
Q3 MongoDB	59 min	/ fast 1.6 min	----	---------
Q4 MySQL Before Optimization	0.016 Sec	1 Sec	1.3 hour	12312434.72
Q4 MySQL After Optimization & Index	0.015 Sec	0.5 sec	26 Sec	6887614.81
Q5 MySQL Before Optimization & Index	-----	-----	30 Sec	1224916.41
Q5 MySQL After	-----	-----	30 Sec	1026290.39

Optimization & Index				
Q5 MongoDB	1 min/ 1.19min	3.10min / fast 4.29min	-----	-----------

Running Queries on 10K-data, 100K-data

Query 1:

```
select c.FirstName, c.SecondName, p.name as product_name
from `OrderComponents` as oc, `Order` as o, `Product` as p,
`Customer` as c
where oc.ProductId = p.Id  and oc.OrderId = o.Id ;
```

on DB with 10K data for each table:

Time			Cost	
Before Any optimization	After Schema Optimization	After Index	Before Optimization	After Optimization
52 Sec	0.063 Sec	0.047 Sec	9762442.15	23291.72

on DB with 100K data for each table:

Time			Cost	
Before Any optimization	After Schema Optimization	After Index	Before Optimization	After Optimization
Out of Memory	50 Sec	27 Sec	Out of memory	8159.82

Query Tree Before Optimization 10K

```
-- EXPLAIN -- "-> Inner hash join (no condition) (cost=9762442.15
rows=97450000) (actual time=171.051..15274.991 rows=100000000
loops=1) -- -> Table scan on c (cost=0.23 rows=9745) (actual
time=1.344..72.263 rows=10000 loops=1) -- -> Hash -- -> Nested loop
inner join (cost=16080.50 rows=10000) (actual time=17.153..154.877
rows=10000 loops=1) -- -> Nested loop inner join (cost=11980.50
rows=10000) (actual time=17.131..122.053 rows=10000 loops=1) -- ->
Index scan on oc using CustomerId (cost=1032.23 rows=10000) (actual
time=16.123..28.960 rows=10000 loops=1) -- -> Single-row index lookup
on p using PRIMARY (ID=oc.ProductId) (cost=0.99 rows=1) (actual
time=0.009..0.009 rows=1 loops=10000) -- -> Single-row index lookup
on o using PRIMARY (Id=oc.OrderId) (cost=0.31 rows=1) (actual
time=0.003..0.003 rows=1 loops=10000) -- "
```

Query Tree After Optimization 10K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=23291.72 rows=10162)
(actual time=3.030..190.507 rows=9000 loops=1) -- -> Nested loop
inner join (cost=12171.70 rows=10162) (actual time=2.319..122.603
rows=9000 loops=1) -- -> Filter: (o.ProductId is not null)
(cost=1046.42 rows=10162) (actual time=1.138..14.032 rows=9000
loops=1) -- -> Index scan on o using ordering_index_FK (cost=1046.42
rows=10162) (actual time=1.136..12.221 rows=10000 loops=1) -- ->
Single-row index lookup on p using PRIMARY (ID=o.ProductId)
(cost=0.99 rows=1) (actual time=0.012..0.012 rows=1 loops=9000) -- ->
Single-row index lookup on c using PRIMARY (Id=o.CustomerId)
(cost=0.99 rows=1) (actual time=0.007..0.007 rows=1 loops=9000) -- "
```

Query Tree Before Optimization 100K

Out of memory

Query Tree After Optimization 100K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=8159.82 rows=10162)
(actual time=0.105..45.538 rows=9000 loops=1) -- -> Nested loop inner
join (cost=4603.12 rows=10162) (actual time=0.089..28.015 rows=9000
loops=1) -- -> Filter: (o.ProductId is not null) (cost=1046.42
rows=10162) (actual time=0.066..5.249 rows=9000 loops=1) -- -> Index
scan on o using ordering_index_FK (cost=1046.42 rows=10162) (actual
time=0.064..4.087 rows=10000 loops=1) -- -> Single-row index lookup
on p using PRIMARY (ID=o.ProductId) (cost=0.25 rows=1) (actual
time=0.002..0.002 rows=1 loops=9000) -- -> Single-row index lookup on
c using PRIMARY (Id=o.CustomerId) (cost=0.25 rows=1) (actual
time=0.002..0.002 rows=1 loops=9000) -- "
```

Query 2

```
select c.FirstName, c.SecondName
from `Order` as o, `Customer` as c , `DeliveryMan` as d
where o.CustomerId = c.Id and o.DeliveryManId = d.Id;
```

on DB with 10K data for each table:

Time			Cost	
Before Any optimization	After Schema Optimization	After Index	Before Optimization	After Optimization
0.047 Sec	0.062 Sec	0.031 Sec	14715.21	8780.84

on DB with 100K data for each table:

Time			Cost	
Before Any optimization	After Schema Optimization	After Index	Before Optimization	After Optimization
5 Sec	60 Sec	22 Sec	230599.67	8159.82

Query Tree Before Optimization 10K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=14715.21 rows=19364)
(actual time=17.450..159.382 rows=9000 loops=1) -- -> Nested loop
inner join (cost=7937.66 rows=19364) (actual time=17.307..117.596
rows=9000 loops=1) -- -> Index scan on d using PhoneNumber
(cost=1160.10 rows=10033) (actual time=16.515..29.713 rows=10000
loops=1) -- -> Index lookup on o using DeliveryManId
(DeliveryManId=d.Id) (cost=0.48 rows=2) (actual time=0.007..0.008
rows=1 loops=10000) -- -> Single-row index lookup on c using PRIMARY
(Id=o.CustomerId) (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1
loops=9000) -- "
```

Query Tree After Optimization 10K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=8780.84 rows=10162)
(actual time=17.334..197.739 rows=9000 loops=1) -- -> Nested loop
inner join (cost=4603.12 rows=10162) (actual time=17.292..103.758
rows=9000 loops=1) -- -> Filter: (o.DeliveryManId is not null)
(cost=1046.42 rows=10162) (actual time=16.360..30.309 rows=9000
loops=1) -- -> Index scan on o using ordering_index_FK (cost=1046.42
rows=10162) (actual time=16.357..28.192 rows=10000 loops=1) -- ->
Single-row index lookup on c using PRIMARY (Id=o.CustomerId)
(cost=0.25 rows=1) (actual time=0.008..0.008 rows=1 loops=9000) -- ->
Single-row index lookup on d using PRIMARY (Id=o.DeliveryManId)
(cost=0.31 rows=1) (actual time=0.010..0.010 rows=1 loops=9000) -- "
```

Query Tree Before Optimization 100K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=230599.67 rows=100296)
(actual time=3.241..19917.694 rows=98219 loops=1) -- -> Nested loop
inner join (cost=120329.87 rows=100296) (actual time=1.874..10127.772
rows=98219 loops=1) -- -> Filter: (o.DeliveryManId is not null)
(cost=10317.70 rows=100296) (actual time=1.102..312.713 rows=98219
loops=1) -- -> Table scan on o (cost=10317.70 rows=100296) (actual
time=1.100..257.562 rows=100000 loops=1) -- -> Single-row index
lookup on d using PRIMARY (Id=o.DeliveryManId) (cost=1.00 rows=1)
(actual time=0.099..0.099 rows=1 loops=98219) -- -> Single-row index
lookup on c using PRIMARY (Id=o.CustomerId) (cost=1.00 rows=1)
(actual time=0.099..0.099 rows=1 loops=98219) -- "
```

Query Tree After Optimization 100K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=8159.82 rows=10162)
(actual time=0.160..57.312 rows=9000 loops=1) -- -> Nested loop inner
join (cost=4603.12 rows=10162) (actual time=0.151..26.667 rows=9000
loops=1) -- -> Filter: (o.DeliveryManId is not null) (cost=1046.42
rows=10162) (actual time=0.128..6.184 rows=9000 loops=1) -- -> Index
scan on o using ordering_index_FK (cost=1046.42 rows=10162) (actual
time=0.126..4.807 rows=10000 loops=1) -- -> Single-row index lookup
on c using PRIMARY (Id=o.CustomerId) (cost=0.25 rows=1) (actual
time=0.002..0.002 rows=1 loops=9000) -- -> Single-row index lookup on
d using PRIMARY (Id=o.DeliveryManId) (cost=0.25 rows=1) (actual
time=0.003..0.003 rows=1 loops=9000) -- "
```

Query 3:

```
select c.FirstName, c.SecondName
from `OrderComponents` as oc,`Order` as o, `Customer` as c ,
`DeliveryMan` as d
where o.CustomerId = c.Id and oc.CustomerId = c.Id and
o.DeliveryManId = d.Id;
```

on DB with 10K data for each table:

Time			Cost	
Before Any optimization	After Schema Optimization	After Index	Before Optimization	After Optimization
0.234 Sec	0.062 Sec	0.031 Sec	20029.40	12171.70

on DB with 100K data for each table:

Time			Cost	
Before Any optimization	After Schema Optimization	After Index	Before Optimization	After Optimization
25 Sec	60 Sec	22 Sec	324921.68	4603.12

Query Tree Before Optimization 10K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=20029.40 rows=18046)
(actual time=1.381..236.002 rows=74225 loops=1) -- -> Nested loop
inner join (cost=15654.04 rows=10268) (actual time=0.867..127.366
rows=9000 loops=1) -- -> Nested loop inner join (cost=4627.10
rows=10268) (actual time=0.062..46.111 rows=9000 loops=1) -- ->
Filter: (o.DeliveryManId is not null) (cost=1033.30 rows=10268)
(actual time=0.044..8.894 rows=9000 loops=1) -- -> Table scan on o
(cost=1033.30 rows=10268) (actual time=0.043..6.961 rows=10000
loops=1) -- -> Single-row index lookup on c using PRIMARY
(Id=o.CustomerId) (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1
loops=9000) -- -> Single-row index lookup on d using PRIMARY
(Id=o.DeliveryManId) (cost=0.97 rows=1) (actual time=0.009..0.009
rows=1 loops=9000) -- -> Index lookup on oc using CustomerId
(CustomerId=o.CustomerId) (cost=0.25 rows=2) (actual
time=0.005..0.011 rows=8 loops=9000) -- "
```

Query Tree Before Optimization 100K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=324921.68 rows=587118)
(actual time=2.357..33067.437 rows=571388 loops=1) -- -> Nested loop
inner join (cost=165454.76 rows=100296) (actual time=1.419..21911.565
rows=98219 loops=1) -- -> Nested loop inner join (cost=64224.99
rows=100296) (actual time=1.114..11369.463 rows=98219 loops=1) -- ->
Filter: (o.DeliveryManId is not null) (cost=10315.89 rows=100296)
(actual time=1.034..441.785 rows=98219 loops=1) -- -> Table scan on o
(cost=10315.89 rows=100296) (actual time=1.031..367.168 rows=100000
loops=1) -- -> Single-row index lookup on d using PRIMARY
(Id=o.DeliveryManId) (cost=0.44 rows=1) (actual time=0.110..0.111
rows=1 loops=98219) -- -> Single-row index lookup on c using PRIMARY
(Id=o.CustomerId) (cost=0.91 rows=1) (actual time=0.106..0.106 rows=1
loops=98219) -- -> Index lookup on oc using CustomerId
(CustomerId=o.CustomerId) (cost=1.00 rows=6) (actual
time=0.107..0.111 rows=6 loops=98219) -- "
```

Query Tree After Optimization 10K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=12171.70 rows=10162)
(actual time=17.558..77.238 rows=9000 loops=1) -- -> Filter:
(o.ProductId is not null) (cost=1046.42 rows=10162) (actual
time=15.993..25.491 rows=9000 loops=1) -- -> Index scan on o using
ProductId (cost=1046.42 rows=10162) (actual time=15.717..24.309
rows=10000 loops=1) -- -> Single-row index lookup on p using PRIMARY
(ID=o.ProductId) (cost=0.99 rows=1) (actual time=0.005..0.005 rows=1
loops=9000) -- "
```

Query Tree After Optimization 100K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=4603.12 rows=10162)
(actual time=16.548..42.153 rows=9000 loops=1) -- -> Filter:
(o.ProductId is not null) (cost=1046.42 rows=10162) (actual
time=16.525..25.498 rows=9000 loops=1) -- -> Index scan on o using
ProductId (cost=1046.42 rows=10162) (actual time=16.090..24.580
rows=10000 loops=1) -- -> Single-row index lookup on p using PRIMARY
(ID=o.ProductId) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1
loops=9000) -- "
```

Query 4:

```
select p.name as product_name
from `OrderComponents` as oc, `Order` as o, `Product` as p
where oc.ProductId = p.Id  and oc.OrderId = o.Id ;
```

on DB with 10K data for each table:

Time			Cost	
Before Any optimization	After Schema Optimization	After Index	Before Optimization	After Optimization
0.016 Sec	0.047 Sec	0.015 Sec	14601.19	21008.42

on DB with 100K data for each table:

Time			Cost	
Before Any optimization	After Schema Optimization	After Index	Before Optimization	After Optimization
1 Sec	0.484 Sec	0.5 Sec	232341.72	11716.53

Query Tree Before Optimization 10K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=14601.19 rows=10000)
(actual time=16.330..125.917 rows=10000 loops=1) -- -> Nested loop
inner join (cost=11101.19 rows=10000) (actual time=16.297..101.253
rows=10000 loops=1) -- -> Index scan on oc using CustomerId
(cost=1032.23 rows=10000) (actual time=0.049..5.435 rows=10000
loops=1) -- -> Single-row index lookup on p using PRIMARY
(ID=oc.ProductId) (cost=0.91 rows=1) (actual time=0.009..0.009 rows=1
loops=10000) -- -> Single-row index lookup on o using PRIMARY
(Id=oc.OrderId) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1
loops=10000) -- "
```

Query Tree After Optimization 10K

```
-- EXPLAIN -- "-> Nested loop inner join (cost=21008.42 rows=10162)
(actual time=16.299..254.044 rows=8102 loops=1) -- -> Nested loop
inner join (cost=11185.15 rows=10162) (actual time=16.288..133.187
rows=8102 loops=1) -- -> Nested loop inner join (cost=4603.12
rows=10162) (actual time=15.524..64.558 rows=8102 loops=1) -- ->
Filter: ((o.ProductId is not null) and (o.DeliveryManId is not null))
(cost=1046.42 rows=10162) (actual time=15.504..29.994 rows=8102
loops=1) -- -> Index scan on o using ordering_index_FK (cost=1046.42
rows=10162) (actual time=15.500..27.465 rows=10000 loops=1) -- ->
Single-row index lookup on p using PRIMARY (ID=o.ProductId)
(cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=8102) -- ->
Single-row index lookup on c using PRIMARY (Id=o.CustomerId)
(cost=0.55 rows=1) (actual time=0.008..0.008 rows=1 loops=8102) -- ->
Single-row index lookup on d using PRIMARY (Id=o.DeliveryManId)
(cost=0.87 rows=1) (actual time=0.014..0.015 rows=1 loops=8102) -- "
```

Query Tree Before Optimization 100K

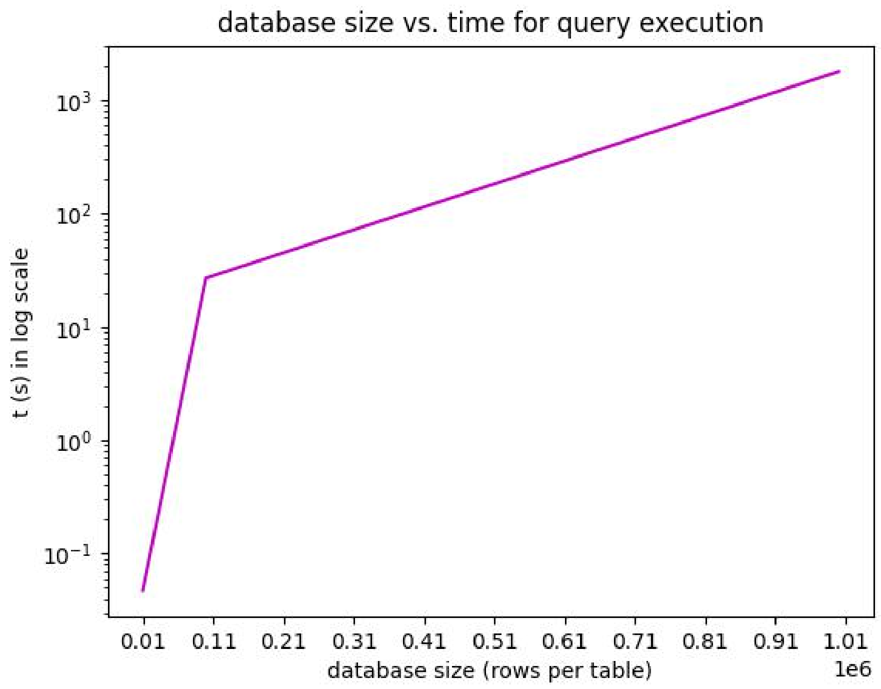
```
-- EXPLAIN -- "-> Nested loop inner join (cost=232341.72 rows=101301)
(actual time=17.281..623.780 rows=100000 loops=1) -- -> Nested loop
inner join (cost=121860.32 rows=101301) (actual time=16.943..502.427
rows=100000 loops=1) -- -> Index scan on oc using CustomerId
(cost=10482.23 rows=101301) (actual time=0.091..100.211 rows=100000
loops=1) -- -> Single-row index lookup on p using PRIMARY
(ID=oc.ProductId) (cost=1.00 rows=1) (actual time=0.004..0.004 rows=1
loops=100000) -- -> Single-row index lookup on o using PRIMARY
(Id=oc.OrderId) (cost=0.99 rows=1) (actual time=0.001..0.001 rows=1
loops=100000) -- "
```

Query Tree After Optimization 100K

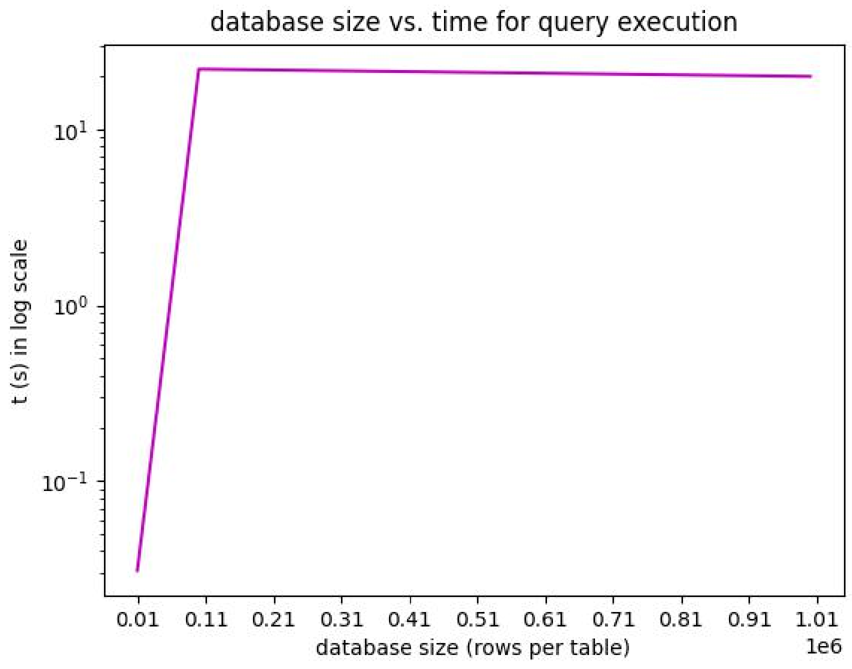
```
-- EXPLAIN -- "-> Nested loop inner join (cost=11716.53 rows=10162)
(actual time=0.146..142.434 rows=8102 loops=1) -- -> Nested loop
inner join (cost=8159.82 rows=10162) (actual time=0.137..92.927
rows=8102 loops=1) -- -> Nested loop inner join (cost=4603.12
rows=10162) (actual time=0.123..60.765 rows=8102 loops=1) -- ->
Filter: ((o.ProductId is not null) and (o.DeliveryManId is not null))
(cost=1046.42 rows=10162) (actual time=0.094..14.228 rows=8102
loops=1) -- -> Index scan on o using ordering_index_FK (cost=1046.42
rows=10162) (actual time=0.092..10.427 rows=10000 loops=1) -- ->
Single-row index lookup on p using PRIMARY (ID=o.ProductId)
(cost=0.25 rows=1) (actual time=0.005..0.005 rows=1 loops=8102) -- ->
Single-row index lookup on c using PRIMARY (Id=o.CustomerId)
(cost=0.25 rows=1) (actual time=0.003..0.004 rows=1 loops=8102) -- ->
Single-row index lookup on d using PRIMARY (Id=o.DeliveryManId)
(cost=0.25 rows=1) (actual time=0.006..0.006 rows=1 loops=8102) --
```

Different Database sizes

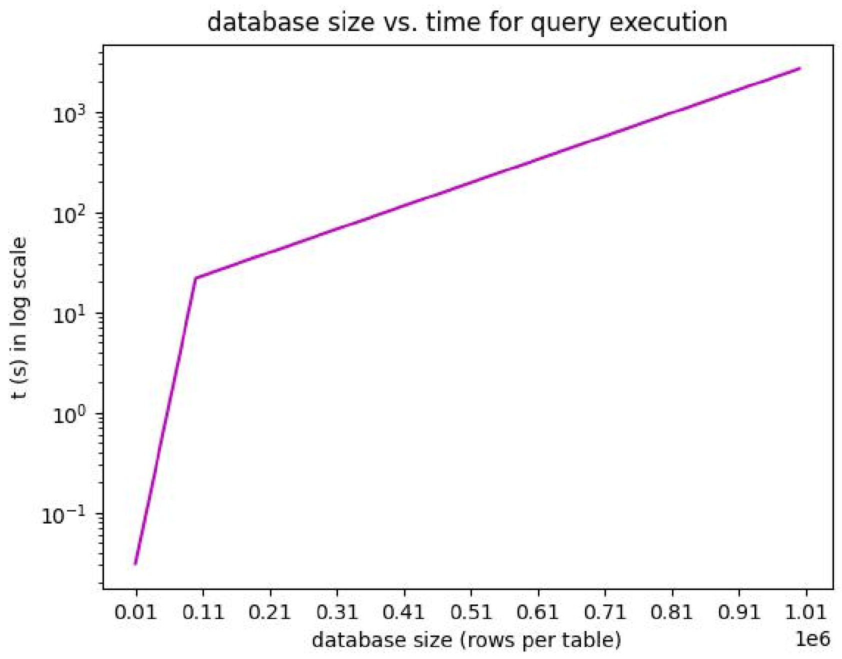
Q1: Y-axis [database size] values are [10K, 100K, 1M]



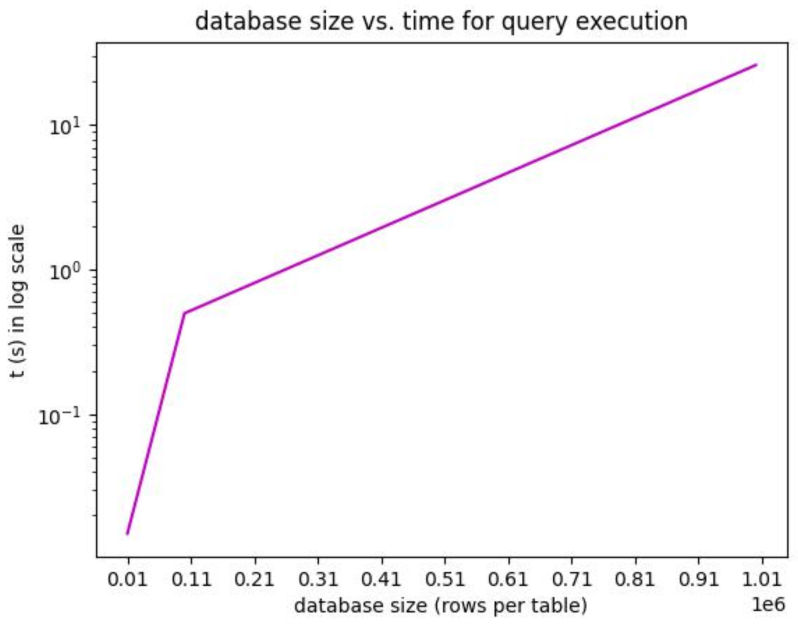
Q2: Y-axis [database size] values are [10K, 100K, 1M]



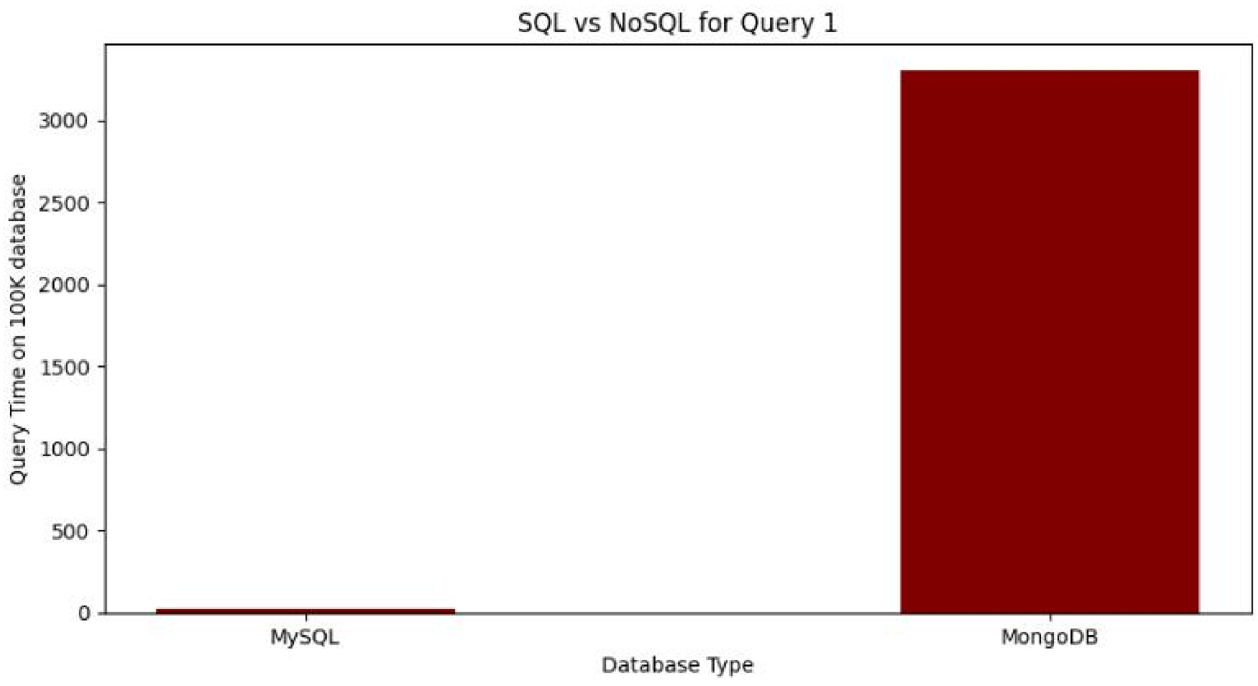
Q3: Y-axis [database size] values are [10K, 100K, 1M]

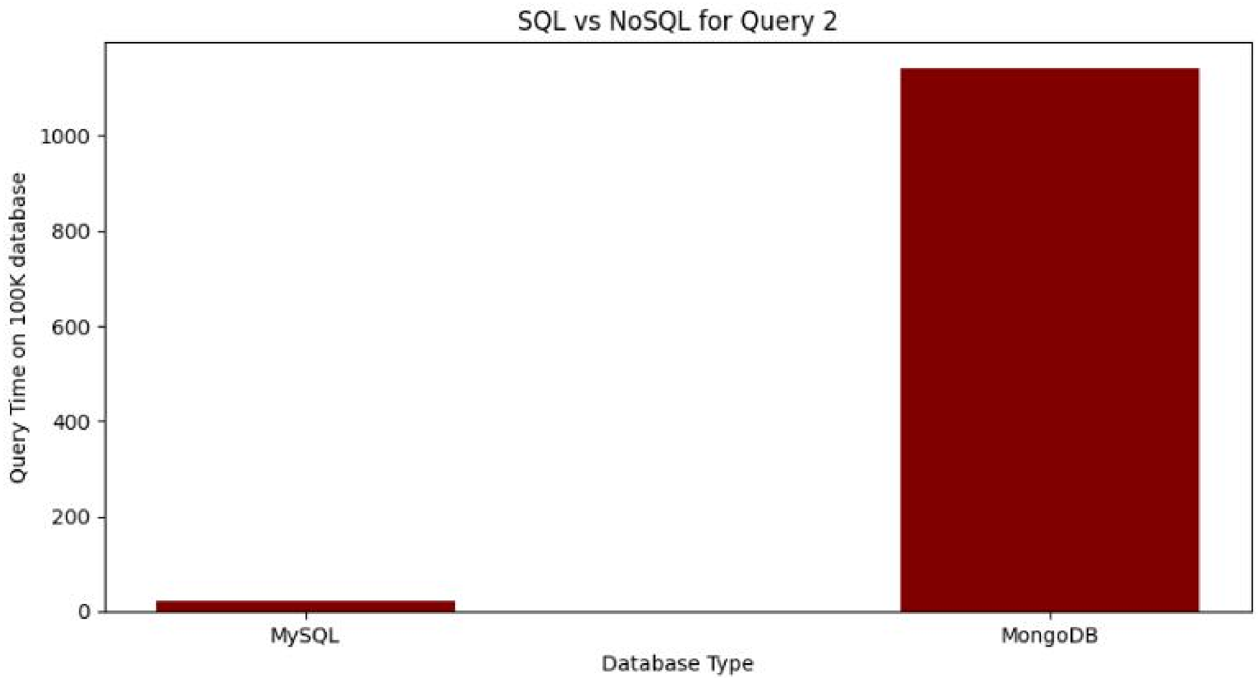


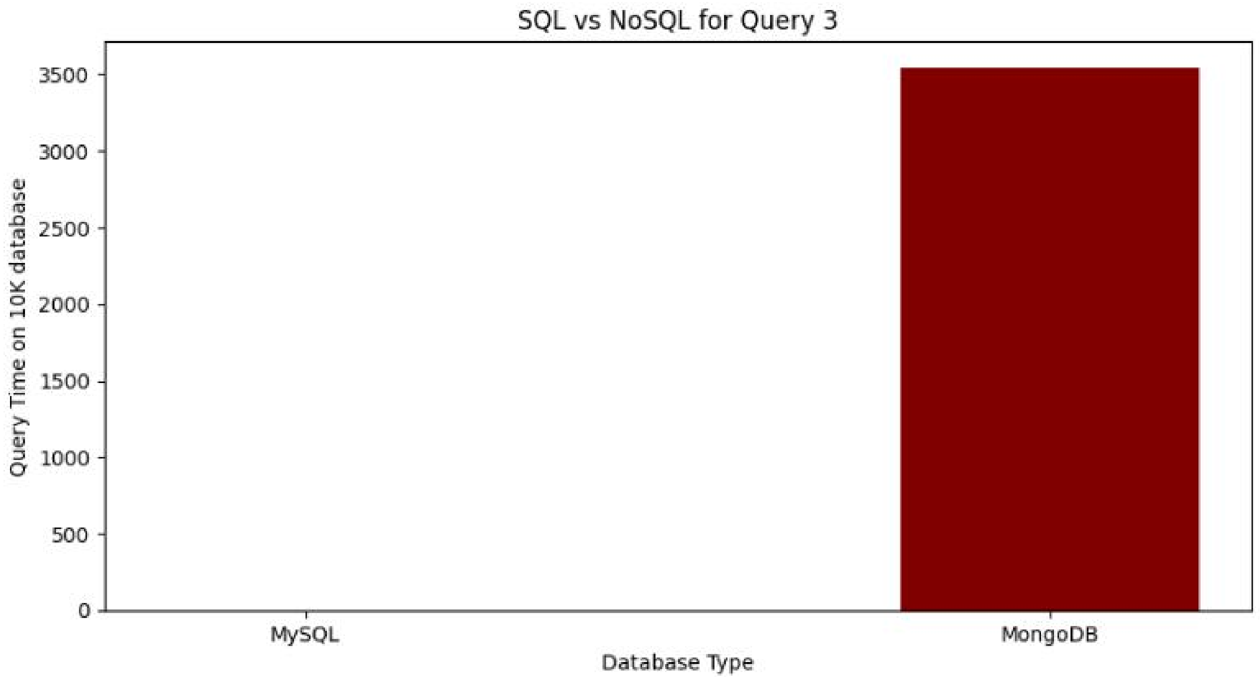
Q4: Y-axis [database size] values are [10K, 100K, 1M]



SQL vs NOSQL:


SQL vs NoSQL for Query 1
Query Time on 100K database
3000
2500
2000
1500
1000
500
0
MySQL
MongoDB
Database Type


SQL vs NoSQL for Query 2
Query Time on 100K database
1000
800
600
400
200
0
MySQL
MongoDB
Database Type


SQL vs NoSQL for Query 3
Query Time on 10K database
3500
3000
2500
2000
1500
1000
500
0
MySQL
MongoDB
Database Type