# Индексы и их использование

Задание выполнено в Greenplum single-node, развернутым в Docker на таблицах созданных по инструкции из ДЗ2 - там таблицы были партиционированы по датам.

1. **Query 1: Retrieve Customer Orders with Order and Customer Details**

**explain** **analyze**

**SELECT** o\_orderkey

,o\_custkey

,o\_orderstatus

,o\_totalprice

,o\_orderdate

,o\_orderpriority

,c\_name **AS** customer\_name

,c\_address **AS** customer\_address

, c\_phone **AS** customer\_phone

**FROM** public.customer c

**JOIN** public.orders o **ON** c.c\_custkey = o.o\_custkey

Gather Motion 4:1 (slice2; segments: 4) (cost=0.00..865.22 rows=1 width=101) (actual time=107.444..176.133 rows=300000 loops=1)

-> Hash Join (cost=0.00..865.22 rows=1 width=101) (actual time=107.104..138.538 rows=76456 loops=1)

Hash Cond: (customer.c\_custkey = orders.o\_custkey)

Extra Text: (seg1) Hash chain length 15.3 avg, 48 max, using 5010 of 262144 buckets.

-> Seq Scan on customer (cost=0.00..431.71 rows=7500 width=65) (actual time=0.101..1.436 rows=7530 loops=1)

-> Hash (cost=431.00..431.00 rows=1 width=40) (actual time=106.300..106.300 rows=76456 loops=1)

-> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..431.00 rows=1 width=40) (actual time=0.115..80.945 rows=76456 loops=1)

Hash Key: orders.o\_custkey

-> Sequence (cost=0.00..431.00 rows=1 width=40) (actual time=0.472..50.579 rows=75163 loops=1)

-> Partition Selector for orders (dynamic scan id: 1) (cost=10.00..100.00 rows=25 width=4) (never executed)

Partitions selected: 87 (out of 87)

-> Dynamic Seq Scan on orders (dynamic scan id: 1) (cost=0.00..431.00 rows=1 width=40) (actual time=0.461..44.673 rows=75163 loops=1)

Partitions scanned: Avg 87.0 (out of 87) x 4 workers. Max 87 parts (seg0).

Planning time: 7.397 ms

(slice0) Executor memory: 592K bytes.

(slice1) Executor memory: 57415K bytes avg x 4 workers, 57430K bytes max (seg1).

(slice2) Executor memory: 18952K bytes avg x 4 workers, 18952K bytes max (seg0). Work\_mem: 5955K bytes max.

Memory used: 128000kB

Optimizer: Pivotal Optimizer (GPORCA)

Execution time: 187.968 ms

**2. Query 2: Retrieve Detailed Order Information with Line Items**

**explain** **analyze**

**SELECT** o\_orderkey

,o\_custkey

,o\_orderstatus

,o\_totalprice

,o\_orderdate

,o\_orderpriority

,c\_name **AS** customer\_name

,c\_address **AS** customer\_address

, c\_phone **AS** customer\_phone

,l\_partkey

,l\_suppkey

,l\_quantity

,l\_extendedprice

,l\_linestatus

,l\_shipdate

**FROM** public.customer c

**JOIN** public.orders o **ON** c.c\_custkey = o.o\_custkey

**join** public.lineitem li **on** li.l\_orderkey = o.o\_orderkey

Gather Motion 4:1 (slice4; segments: 4) (cost=0.00..1296.22 rows=1 width=130) (actual time=584.038..1030.090 rows=1140352 loops=1)

-> Hash Join (cost=0.00..1296.22 rows=1 width=130) (actual time=583.611..776.055 rows=286297 loops=1)

Hash Cond: (orders.o\_orderkey = lineitem.l\_orderkey)

Extra Text: (seg3) Hash chain length 5.2 avg, 33 max, using 55486 of 131072 buckets.

-> Redistribute Motion 4:4 (slice2; segments: 4) (cost=0.00..865.22 rows=1 width=101) (actual time=0.008..41.305 rows=75163 loops=1)

Hash Key: orders.o\_orderkey

-> Hash Join (cost=0.00..865.22 rows=1 width=101) (actual time=318.293..354.657 rows=76456 loops=1)

Hash Cond: (customer.c\_custkey = orders.o\_custkey)

Extra Text: (seg1) Hash chain length 15.4 avg, 49 max, using 4958 of 131072 buckets.

-> Seq Scan on customer (cost=0.00..431.71 rows=7500 width=65) (actual time=0.216..1.893 rows=7530 loops=1)

-> Hash (cost=431.00..431.00 rows=1 width=40) (actual time=317.362..317.362 rows=76456 loops=1)

-> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..431.00 rows=1 width=40) (actual time=0.047..274.593 rows=76456 loops=1)

Hash Key: orders.o\_custkey

-> Sequence (cost=0.00..431.00 rows=1 width=40) (actual time=6.382..223.236 rows=75163 loops=1)

-> Partition Selector for orders (dynamic scan id: 1) (cost=10.00..100.00 rows=25 width=4) (never executed)

Partitions selected: 87 (out of 87)

-> Dynamic Seq Scan on orders (dynamic scan id: 1) (cost=0.00..431.00 rows=1 width=40) (actual time=6.364..215.012 rows=75163 loops=1)

Partitions scanned: Avg 87.0 (out of 87) x 4 workers. Max 87 parts (seg0).

-> Hash (cost=431.00..431.00 rows=1 width=37) (actual time=583.019..583.019 rows=286297 loops=1)

-> Redistribute Motion 4:4 (slice3; segments: 4) (cost=0.00..431.00 rows=1 width=37) (actual time=2.711..482.553 rows=286297 loops=1)

Hash Key: lineitem.l\_orderkey

-> Sequence (cost=0.00..431.00 rows=1 width=37) (actual time=1.850..309.815 rows=286000 loops=1)

-> Partition Selector for lineitem (dynamic scan id: 2) (cost=10.00..100.00 rows=25 width=4) (never executed)

Partitions selected: 81 (out of 81)

-> Dynamic Seq Scan on lineitem (dynamic scan id: 2) (cost=0.00..431.00 rows=1 width=37) (actual time=1.830..288.803 rows=286000 loops=1)

Partitions scanned: Avg 81.0 (out of 81) x 4 workers. Max 81 parts (seg0).

Planning time: 19.156 ms

(slice0) Executor memory: 707K bytes.

(slice1) Executor memory: 57419K bytes avg x 4 workers, 57434K bytes max (seg1).

(slice2) Executor memory: 17928K bytes avg x 4 workers, 17928K bytes max (seg0). Work\_mem: 5955K bytes max.

(slice3) Executor memory: 69297K bytes avg x 4 workers, 69327K bytes max (seg3).

(slice4) Executor memory: 42120K bytes avg x 4 workers, 42168K bytes max (seg1). Work\_mem: 19729K bytes max.

Memory used: 128000kB

Optimizer: Pivotal Optimizer (GPORCA)

Execution time: 1102.636 ms

**3.Query 3: Retrieve Supplier and Part Information for Each Supplier-Part Relationship**

**explain** **analyze**

**select** s\_suppkey

, s\_name

, s\_address

, p\_partkey

, p\_name

, p\_mfgr

, p\_brand

, p\_type

, p\_size

, p\_retailprice

, ps\_availqty

**from** public.supplier s

**join** public.partsupp ps **on** ps.ps\_suppkey = s.s\_suppkey

**join** public.part p **on** p.p\_partkey = ps.ps\_partkey

**Gather Motion 4:1 (slice3; segments: 4) (cost=0.00..1457.64 rows=157002 width=165) (actual time=24.067..83.168 rows=160000 loops=1)**

**-> Hash Join (cost=0.00..1370.98 rows=39251 width=165) (actual time=23.590..42.121 rows=40220 loops=1)**

**Hash Cond: (partsupp.ps\_suppkey = supplier.s\_suppkey)**

**Extra Text: (seg0) Hash chain length 1.0 avg, 2 max, using 1988 of 131072 buckets.**

**-> Hash Join (cost=0.00..903.41 rows=39251 width=113) (actual time=19.070..28.676 rows=40220 loops=1)**

**Hash Cond: (part.p\_partkey = partsupp.ps\_partkey)**

**Extra Text: (seg0) Hash chain length 4.1 avg, 8 max, using 9858 of 262144 buckets.**

**-> Seq Scan on part (cost=0.00..431.79 rows=10000 width=105) (actual time=0.226..1.926 rows=10055 loops=1)**

**-> Hash (cost=436.87..436.87 rows=40000 width=12) (actual time=18.518..18.518 rows=40220 loops=1)**

**-> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..436.87 rows=40000 width=12) (actual time=0.032..11.400 rows=40220 loops=1)**

**Hash Key: partsupp.ps\_partkey**

**-> Seq Scan on partsupp (cost=0.00..434.48 rows=40000 width=12) (actual time=0.121..3.849 rows=40196 loops=1)**

**-> Hash (cost=432.64..432.64 rows=2000 width=56) (actual time=1.180..1.180 rows=2000 loops=1)**

**-> Broadcast Motion 4:4 (slice2; segments: 4) (cost=0.00..432.64 rows=2000 width=56) (actual time=0.014..0.824 rows=2000 loops=1)**

**-> Seq Scan on supplier (cost=0.00..431.04 rows=500 width=56) (actual time=0.044..0.085 rows=517 loops=1)**

**Planning time: 14.509 ms**

**(slice0) Executor memory: 1576K bytes.**

**(slice1) Executor memory: 412K bytes avg x 4 workers, 412K bytes max (seg0).**

**(slice2) Executor memory: 396K bytes avg x 4 workers, 396K bytes max (seg0).**

**(slice3) Executor memory: 6600K bytes avg x 4 workers, 6600K bytes max (seg0). Work\_mem: 1257K bytes max.**

**Memory used: 128000kB**

**Optimizer: Pivotal Optimizer (GPORCA)**

**Execution time: 95.444 ms**

**4. Query 4: Retrieve Comprehensive Customer Order and Line Item Details**

**explain** **analyze**

**SELECT** o\_orderkey

,o\_custkey

,o\_orderstatus

,o\_totalprice

,o\_orderdate

,o\_orderpriority

,c\_name **AS** customer\_name

,c\_address **AS** customer\_address

, c\_phone **AS** customer\_phone

,l\_partkey

,l\_suppkey

,l\_quantity

,l\_extendedprice

,l\_linestatus

,l\_shipdate

, s.s\_name **as** supp\_name

**FROM** public.customer c

**JOIN** public.orders o **ON** c.c\_custkey = o.o\_custkey

**join** public.lineitem li **on** li.l\_orderkey = o.o\_orderkey

**join** public.supplier s **on** li.l\_suppkey = s.s\_suppkey

**where**

c.c\_custkey = 19

**Hash Join (cost=0.00..1726.23 rows=500 width=156) (actual time=1379.716..1549.019 rows=52 loops=1)**

**Hash Cond: (orders.o\_orderkey = lineitem.l\_orderkey)**

**Extra Text: Initial batch 0:**

**Wrote 70198K bytes to inner workfile.**

**Wrote 2K bytes to outer workfile.**

**Overflow batches 1..3:**

**Read 88799K bytes from inner workfile: 29600K avg x 3 nonempty batches, 42088K max.**

**Wrote 18601K bytes to inner workfile.**

**Read 2K bytes from outer workfile: 1K avg x 3 nonempty batches, 1K max.**

**Hash chain length 6.5 avg, 41 max, using 175262 of 262144 buckets.**

**-> Gather Motion 4:1 (slice2; segments: 4) (cost=0.00..862.96 rows=1 width=101) (actual time=0.004..0.006 rows=14 loops=1)**

**-> Hash Join (cost=0.00..862.96 rows=1 width=101) (actual time=206.047..207.090 rows=14 loops=1)**

**Hash Cond: (customer.c\_custkey = orders.o\_custkey)**

**Extra Text: (seg0) Hash chain length 14.0 avg, 14 max, using 1 of 131072 buckets.**

**-> Seq Scan on customer (cost=0.00..431.96 rows=1 width=65) (actual time=1.265..2.209 rows=1 loops=1)**

**Filter: (c\_custkey = 19)**

**-> Hash (cost=431.00..431.00 rows=1 width=40) (actual time=181.383..181.383 rows=14 loops=1)**

**-> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..431.00 rows=1 width=40) (actual time=136.220..181.361 rows=14 loops=1)**

**Hash Key: orders.o\_custkey**

**-> Sequence (cost=0.00..431.00 rows=1 width=40) (actual time=7.450..154.298 rows=6 loops=1)**

**-> Partition Selector for orders (dynamic scan id: 1) (cost=10.00..100.00 rows=25 width=4) (never executed)**

**Partitions selected: 87 (out of 87)**

**-> Dynamic Seq Scan on orders (dynamic scan id: 1) (cost=0.00..431.00 rows=1 width=40) (actual time=7.438..154.283 rows=6 loops=1)**

**Filter: (o\_custkey = 19)**

**Partitions scanned: Avg 87.0 (out of 87) x 4 workers. Max 87 parts (seg0).**

**-> Hash (cost=862.17..862.17 rows=1 width=63) (actual time=1379.653..1379.653 rows=1140352 loops=1)**

**Buckets: 65536 Batches: 4 (originally 1) Memory Usage: 42451kB**

**-> Gather Motion 4:1 (slice4; segments: 4) (cost=0.00..862.17 rows=1 width=63) (actual time=729.921..1018.494 rows=1140352 loops=1)**

**-> Hash Join (cost=0.00..862.17 rows=1 width=63) (actual time=731.245..838.461 rows=294517 loops=1)**

**Hash Cond: (supplier.s\_suppkey = lineitem.l\_suppkey)**

**Extra Text: (seg0) Hash chain length 569.7 avg, 650 max, using 517 of 131072 buckets.**

**-> Seq Scan on supplier (cost=0.00..431.04 rows=500 width=30) (actual time=0.167..0.510 rows=517 loops=1)**

**-> Hash (cost=431.00..431.00 rows=1 width=37) (actual time=730.328..730.328 rows=294517 loops=1)**

**-> Redistribute Motion 4:4 (slice3; segments: 4) (cost=0.00..431.00 rows=1 width=37) (actual time=0.035..643.079 rows=294517 loops=1)**

**Hash Key: lineitem.l\_suppkey**

**-> Sequence (cost=0.00..431.00 rows=1 width=37) (actual time=6.643..328.316 rows=286000 loops=1)**

**-> Partition Selector for lineitem (dynamic scan id: 2) (cost=10.00..100.00 rows=25 width=4) (never executed)**

**Partitions selected: 81 (out of 81)**

**-> Dynamic Seq Scan on lineitem (dynamic scan id: 2) (cost=0.00..431.00 rows=1 width=37) (actual time=6.633..264.788 rows=286000 loops=1)**

**Partitions scanned: Avg 81.0 (out of 81) x 4 workers. Max 81 parts (seg0).**

**Planning time: 64.248 ms**

**\* (slice0) Executor memory: 67165K bytes. Work\_mem: 42451K bytes max, 106908K bytes wanted.**

**(slice1) Executor memory: 61499K bytes avg x 4 workers, 61514K bytes max (seg1).**

**(slice2) Executor memory: 1514K bytes avg x 4 workers, 1544K bytes max (seg0). Work\_mem: 2K bytes max.**

**(slice3) Executor memory: 69297K bytes avg x 4 workers, 69327K bytes max (seg3).**

**(slice4) Executor memory: 42312K bytes avg x 4 workers, 42312K bytes max (seg0). Work\_mem: 20304K bytes max.**

**Memory used: 128000kB**

**Memory wanted: 322224kB**

**Optimizer: Pivotal Optimizer (GPORCA)**

**Execution time: 1550.297 ms**

**5. Query 5: Retrieve All Parts Supplied by a Specific Supplier with Supplier Details**

**with** cte **as**

(**select** 1004 **as** *vsuppkey*)

**select** s\_suppkey

, s\_name

, s\_address

, p\_partkey

, p\_name

, p\_mfgr

, p\_brand

, p\_type

, p\_size

, p\_retailprice

, ps\_availqty

**from** public.supplier *s*

**join** public.partsupp *ps* **on** *ps*.ps\_suppkey = *s*.s\_suppkey

**join** public.part *p* **on** *p*.p\_partkey = *ps*.ps\_partkey

**join** cte *c* **on** *c*.vsuppkey = *s*.s\_suppkey

**Gather Motion 4:1 (slice4; segments: 4) (cost=0.00..1303.08 rows=99 width=165) (actual time=16.721..20.381 rows=80 loops=1)**

**-> Hash Join (cost=0.00..1303.02 rows=25 width=165) (actual time=14.145..16.942 rows=21 loops=1)**

**Hash Cond: ((partsupp.ps\_suppkey = supplier.s\_suppkey) AND (partsupp.ps\_suppkey = "outer".vsuppkey))**

**Extra Text: (seg1) Hash chain length 1.0 avg, 1 max, using 1 of 65536 buckets.**

**-> Hash Join (cost=0.00..871.93 rows=25 width=113) (actual time=8.148..10.897 rows=21 loops=1)**

**Hash Cond: (part.p\_partkey = partsupp.ps\_partkey)**

**Extra Text: (seg1) Hash chain length 1.0 avg, 1 max, using 21 of 131072 buckets.**

**-> Seq Scan on part (cost=0.00..431.79 rows=10000 width=105) (actual time=3.577..5.223 rows=10055 loops=1)**

**-> Hash (cost=435.79..435.79 rows=25 width=12) (actual time=2.559..2.559 rows=21 loops=1)**

**-> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..435.79 rows=25 width=12) (actual time=0.005..2.551 rows=21 loops=1)**

**Hash Key: partsupp.ps\_partkey**

**-> Seq Scan on partsupp (cost=0.00..435.79 rows=25 width=12) (actual time=4.045..7.363 rows=23 loops=1)**

**Filter: (ps\_suppkey = 1004)**

**-> Hash (cost=431.06..431.06 rows=1 width=60) (actual time=3.885..3.885 rows=1 loops=1)**

**-> Broadcast Motion 1:4 (slice3) (cost=0.00..431.06 rows=4 width=60) (actual time=3.882..3.883 rows=1 loops=1)**

**-> Hash Join (cost=0.00..431.06 rows=1 width=60) (actual time=2.734..2.976 rows=1 loops=1)**

**Hash Cond: (supplier.s\_suppkey = "outer".vsuppkey)**

**Extra Text: Hash chain length 1.0 avg, 1 max, using 1 of 262144 buckets.**

**-> Gather Motion 4:1 (slice2; segments: 4) (cost=0.00..431.06 rows=1 width=56) (actual time=0.005..0.014 rows=1 loops=1)**

**-> Seq Scan on supplier (cost=0.00..431.06 rows=1 width=56) (actual time=1.949..1.965 rows=1 loops=1)**

**Filter: (s\_suppkey = 1004)**

**-> Hash (cost=0.00..0.00 rows=1 width=4) (actual time=0.006..0.006 rows=1 loops=1)**

**-> Result (cost=0.00..0.00 rows=1 width=4) (actual time=0.002..0.002 rows=1 loops=1)**

**-> Result (cost=0.00..0.00 rows=1 width=1) (actual time=0.001..0.001 rows=1 loops=1)**

**Planning time: 60.894 ms**

**(slice0) Executor memory: 1560K bytes.**

**(slice1) Executor memory: 396K bytes avg x 4 workers, 396K bytes max (seg0).**

**(slice2) Executor memory: 396K bytes avg x 4 workers, 396K bytes max (seg0).**

**(slice3) Executor memory: 3640K bytes (entry db). Work\_mem: 1K bytes max.**

**(slice4) Executor memory: 2424K bytes avg x 4 workers, 2424K bytes max (seg0). Work\_mem: 1K bytes max.**

**Memory used: 128000kB**

**Optimizer: Pivotal Optimizer (GPORCA)**

**Execution time: 53.526 ms**

# Используя материалы вебинара, оптимизируйте планы запросов. Примените vacuum, analyze, партиционирования и построение индексов.

**Созданы таблицы без партиционирования по датам с ключами дистрибуции, которые участвуют в джойнах. Выполнен запрос из п.2. Запрос выполнился оптимальнее, т.к. Первоначальные партиции по датам ему только мешали.**

**CREATE** **TABLE** lineitem2 (

L\_ORDERKEY **BIGINT**,

L\_PARTKEY **INT**,

L\_SUPPKEY **INT**,

L\_LINENUMBER **INTEGER**,

L\_QUANTITY **DECIMAL**(15, 2),

L\_EXTENDEDPRICE **DECIMAL**(15, 2),

L\_DISCOUNT **DECIMAL**(15, 2),

L\_TAX **DECIMAL**(15, 2),

L\_RETURNFLAG **CHAR**(1),

L\_LINESTATUS **CHAR**(1),

L\_SHIPDATE **DATE**,

L\_COMMITDATE **DATE**,

L\_RECEIPTDATE **DATE**,

L\_SHIPINSTRUCT **CHAR**(25),

L\_SHIPMODE **CHAR**(10),

L\_COMMENT **VARCHAR**(44)

) **WITH** (

appendonly = **true**,

orientation = **column**,

compresstype = ZSTD

)

**DISTRIBUTED** **BY** (L\_ORDERKEY)

**insert** **into** lineitem2

**select** \*

**from** lineitem

**CREATE** **TABLE** orders2 (

O\_ORDERKEY **BIGINT**,

O\_CUSTKEY **INT**,

O\_ORDERSTATUS **CHAR**(1),

O\_TOTALPRICE **DECIMAL**(15, 2),

O\_ORDERDATE **DATE**,

O\_ORDERPRIORITY **CHAR**(15),

O\_CLERK **CHAR**(15),

O\_SHIPPRIORITY **INTEGER**,

O\_COMMENT **VARCHAR**(79)

) **WITH** (

appendonly = **true**,

orientation = **column**,

compresstype = ZSTD

)

**DISTRIBUTED** **BY** (O\_ORDERKEY)

**insert** **into** orders2

**select** \*

**from** orders

**explain** **analyze**

**SELECT** o\_orderkey

,o\_custkey

,o\_orderstatus

,o\_totalprice

,o\_orderdate

,o\_orderpriority

,c\_name **AS** customer\_name

,c\_address **AS** customer\_address

, c\_phone **AS** customer\_phone

,l\_partkey

,l\_suppkey

,l\_quantity

,l\_extendedprice

,l\_linestatus

,l\_shipdate

**FROM** public.customer c

**JOIN** public.orders2 o **ON** c.c\_custkey = o.o\_custkey

**join** public.lineitem2 li **on** li.l\_orderkey = o.o\_orderkey

Gather Motion 4:1 (slice3; segments: 4) (cost=0.00..2297.29 rows=1140352 width=130) (actual time=168.673..730.316 rows=1140352 loops=1)

-> Hash Join (cost=0.00..1801.41 rows=285088 width=130) (actual time=168.074..348.755 rows=286297 loops=1)

Hash Cond: (lineitem2.l\_orderkey = orders2.o\_orderkey)

Extra Text: (seg3) Hash chain length 1.7 avg, 8 max, using 44545 of 65536 buckets.

-> Seq Scan on lineitem2 (cost=0.00..452.17 rows=285088 width=35) (actual time=0.720..58.491 rows=286297 loops=1)

-> Hash (cost=962.48..962.48 rows=75000 width=103) (actual time=164.645..164.645 rows=75163 loops=1)

-> Redistribute Motion 4:4 (slice2; segments: 4) (cost=0.00..962.48 rows=75000 width=103) (actual time=9.851..133.437 rows=75163 loops=1)

Hash Key: orders2.o\_orderkey

-> Hash Join (cost=0.00..938.30 rows=75000 width=103) (actual time=8.615..67.613 rows=76456 loops=1)

Hash Cond: (orders2.o\_custkey = customer.c\_custkey)

Extra Text: (seg1) Hash chain length 1.0 avg, 3 max, using 7309 of 131072 buckets.

-> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..451.87 rows=75000 width=42) (actual time=0.020..24.965 rows=76456 loops=1)

Hash Key: orders2.o\_custkey

-> Seq Scan on orders2 (cost=0.00..436.16 rows=75000 width=42) (actual time=0.723..18.067 rows=75163 loops=1)

-> Hash (cost=431.71..431.71 rows=7500 width=65) (actual time=8.010..8.010 rows=7530 loops=1)

-> Seq Scan on customer (cost=0.00..431.71 rows=7500 width=65) (actual time=3.067..4.839 rows=7530 loops=1)

Planning time: 42.352 ms

(slice0) Executor memory: 1960K bytes.

(slice1) Executor memory: 896K bytes avg x 4 workers, 896K bytes max (seg0).

(slice2) Executor memory: 3592K bytes avg x 4 workers, 3592K bytes max (seg0). Work\_mem: 696K bytes max.

(slice3) Executor memory: 26216K bytes avg x 4 workers, 26216K bytes max (seg0). Work\_mem: 10537K bytes max.

Memory used: 128000kB

Optimizer: Pivotal Optimizer (GPORCA)

Execution time: 783.628 ms

**Cоздаем индекс на таблицу с клиентами выполняем запрос выше, из плана запроса видно, что индекс не был использован**

**create** **index** idx\_custkey **on** public.customer(c\_custkey)

**explain** **analyze**

**SELECT** o\_orderkey

,o\_custkey

,o\_orderstatus

,o\_totalprice

,o\_orderdate

,o\_orderpriority

,c\_name **AS** customer\_name

,c\_address **AS** customer\_address

, c\_phone **AS** customer\_phone

,l\_partkey

,l\_suppkey

,l\_quantity

,l\_extendedprice

,l\_linestatus

,l\_shipdate

**FROM** public.customer c

**JOIN** public.orders2 o **ON** c.c\_custkey = o.o\_custkey

**join** public.lineitem2 li **on** li.l\_orderkey = o.o\_orderkey

**where** c\_custkey = 19

**А если подставить выборку по конкретному клиенту, то индекс используется**

**Gather Motion 4:1 (slice2; segments: 4) (cost=0.00..1353.00 rows=78 width=130) (actual time=224.130..246.730 rows=52 loops=1)**

**-> Hash Join (cost=0.00..1352.97 rows=20 width=130) (actual time=24.885..242.111 rows=25 loops=1)**

**Hash Cond: (lineitem2.l\_orderkey = orders2.o\_orderkey)**

**Extra Text: (seg1) Hash chain length 1.0 avg, 1 max, using 6 of 65536 buckets.**

**-> Seq Scan on lineitem2 (cost=0.00..452.17 rows=285088 width=35) (actual time=0.969..109.720 rows=286297 loops=1)**

**-> Hash (cost=826.60..826.60 rows=4 width=103) (actual time=17.369..17.369 rows=6 loops=1)**

**-> Hash Join (cost=0.00..826.60 rows=4 width=103) (actual time=5.259..17.359 rows=6 loops=1)**

**Hash Cond: (orders2.o\_custkey = customer.c\_custkey)**

**Extra Text: (seg1) Hash chain length 1.0 avg, 1 max, using 1 of 131072 buckets.**

**-> Seq Scan on orders2 (cost=0.00..438.62 rows=4 width=42) (actual time=1.619..13.611 rows=6 loops=1)**

**Filter: (o\_custkey = 19)**

**-> Hash (cost=387.97..387.97 rows=1 width=65) (actual time=3.407..3.407 rows=1 loops=1)**

**-> Broadcast Motion 4:4 (slice1; segments: 4) (cost=0.00..387.97 rows=1 width=65) (actual time=1.400..3.404 rows=1 loops=1)**

**-> Bitmap Heap Scan on customer (cost=0.00..387.97 rows=1 width=65) (actual time=0.652..0.671 rows=1 loops=1)**

**Recheck Cond: (c\_custkey = 19)**

**-> Bitmap Index Scan on idx\_custkey (cost=0.00..0.00 rows=0 width=0) (actual time=0.073..0.073 rows=1 loops=1)**

**Index Cond: (c\_custkey = 19)**

**Planning time: 25.269 ms**

**(slice0) Executor memory: 1576K bytes.**

**(slice1) Executor memory: 1084K bytes avg x 4 workers, 1108K bytes max (seg0). Work\_mem: 9K bytes max.**

**(slice2) Executor memory: 3425K bytes avg x 4 workers, 3425K bytes max (seg0). Work\_mem: 1K bytes max.**

**Memory used: 128000kB**

**Optimizer: Pivotal Optimizer (GPORCA)**

**Execution time: 254.914 ms**

**Теперь индекс удаляем и повторяем запрос, запрос без индекса для выборки конкретного клиента выполнился менее оптимально.**

**DROP** **INDEX** public.idx\_custkey;

**Gather Motion 4:1 (slice2; segments: 4) (cost=0.00..1397.00 rows=78 width=130) (actual time=80.329..264.677 rows=52 loops=1)**

**-> Hash Join (cost=0.00..1396.96 rows=20 width=130) (actual time=22.596..263.714 rows=25 loops=1)**

**Hash Cond: (lineitem2.l\_orderkey = orders2.o\_orderkey)**

**Extra Text: (seg1) Hash chain length 1.0 avg, 1 max, using 6 of 65536 buckets.**

**-> Seq Scan on lineitem2 (cost=0.00..452.17 rows=285088 width=35) (actual time=0.851..91.269 rows=286297 loops=1)**

**-> Hash (cost=870.59..870.59 rows=4 width=103) (actual time=15.451..15.451 rows=6 loops=1)**

**-> Hash Join (cost=0.00..870.59 rows=4 width=103) (actual time=7.449..15.442 rows=6 loops=1)**

**Hash Cond: (orders2.o\_custkey = customer.c\_custkey)**

**Extra Text: (seg1) Hash chain length 1.0 avg, 1 max, using 1 of 131072 buckets.**

**-> Seq Scan on orders2 (cost=0.00..438.62 rows=4 width=42) (actual time=2.534..10.431 rows=6 loops=1)**

**Filter: (o\_custkey = 19)**

**-> Hash (cost=431.96..431.96 rows=1 width=65) (actual time=3.878..3.878 rows=1 loops=1)**

**-> Broadcast Motion 4:4 (slice1; segments: 4) (cost=0.00..431.96 rows=1 width=65) (actual time=2.526..3.875 rows=1 loops=1)**

**-> Seq Scan on customer (cost=0.00..431.96 rows=1 width=65) (actual time=0.236..0.798 rows=1 loops=1)**

**Filter: (c\_custkey = 19)**

**Planning time: 33.512 ms**

**(slice0) Executor memory: 1944K bytes.**

**(slice1) Executor memory: 492K bytes avg x 4 workers, 492K bytes max (seg0).**

**(slice2) Executor memory: 3425K bytes avg x 4 workers, 3425K bytes max (seg0). Work\_mem: 1K bytes max.**

**Memory used: 128000kB**

**Optimizer: Pivotal Optimizer (GPORCA)**

**Execution time: 275.336 ms**