Query1

BEFORE:

latency average = 0.585 ms

tps = 1709.458839 (including connections establishing)

tps = 1709.505765 (excluding connections establishing)

--CREATE INDEX Q1 ON "Track" using btree("TrackId") where "Composer" is null;

Values foe Q1 was: 100000, 97800000, 97800000

AFTER:

latency average = 0.518 ms

tps = 1929.682907 (including connections establishing)

tps = 1929.771498 (excluding connections establishing)

Planning:

Buffers: shared hit=106

Planning Time: 0.189 ms

Execution Time: 0.255 ms

Justification:

Using B+Trees, on the "TrackId" column with a filter condition on the "Composer" column being null, the query benefits from reduced disk I/O, faster lookup times, improved data locality, and enhanced index scans. These optimizations contribute to the observed improvements in latency and TPS metrics after creating the index.

Query2

BEFORE:

latency average = 0.986 ms

tps = 1013.849146 (including connections establishing)

tps = 1013.864680 (excluding connections establishing)

I used partial , hash , btree and brin index but the values are 0,0,0

After:

latency average = 0.985 ms

tps = 1015.636108 (including connections establishing)

tps = 1015.652628 (excluding connections establishing)

Planning:

Buffers: shared hit=106

Planning Time: 0.260 ms

Execution Time: 0.351 ms

Justification: using different types of indexes (partial, hash, B-tree, and BRIN) did not have a significant effect on the latency and TPS metrics for the given query. The values before and after creating the indexes are similar, with few changes observed.

Query3

BEFORE:

latency average = 0.541 ms

tps = 1848.127258 (including connections establishing) tps = 1848.179906 (excluding connections establishing)

--CREATE INDEX Q3 ON "Track" using btree("TrackId") where "Composer" = 'AC/DC';

-- CREATE INDEX Q33 ON "PlaylistTrack" USING btree ("TrackId");

Values for Q3 are: 100000, 800000,0

Valuse for Q33 are : 1000000, 1800000,0

Also, if we use table: Playlist

Column:PlaylistId

Using btree index

The output will be : 1600000, 1600000, 1600000

After for Q33:

latency average = 0.180 ms

tps = 5556.784661 (including connections establishing) tps = 5557.244615 (excluding connections establishing)

AFTER:

latency average = 0.532 ms

tps = 1880.203990 (including connections establishing)

tps = 1880.265169 (excluding connections establishing)

Planning:

Buffers: shared hit=198 Planning Time: 0.444 ms Execution Time: 0.999 ms

Justification:

improvements in latency and TPS observed after creating the B-tree indexes on the "TrackId" column with different filter conditions and on the "TrackId" column in the "PlaylistTrack" table can be attributed to these optimizations. The reduced disk I/O, faster lookup times, and improved index join and usage contribute to the overall improved performance of the query.

Query4

BEFORE:

latency average = 0.585 ms

tps = 1710.672839 (including connections establishing)

tps = 1710.723439 (excluding connections establishing)

--CREATE INDEX Q4 ON "Track" using btree("TrackId") where "Composer" = 'AC/DC';

--CREATE INDEX Q44 ON "PlaylistTrack" using btree ("TrackId", "PlaylistId");

Values for Q4 are: 100000, 800000,0

Values for Q44 are: 1000000, 1800000,0

AFTER:  
latency average = 0.179 ms

tps = 5595.155774 (including connections establishing)

tps = 5595.584834 (excluding connections establishing)

Also, if we use table: Playlist

Column: PlaylistId

Using b+tree index

The output will be: 1600000, 1600000, 1600000

AFTER:  
latency average = 0.512 ms

tps = 1952.225642 (including connections establishing)

tps = 1952.288991 (excluding connections establishing)

Planning:

Buffers: shared hit=202 Planning Time: 0.425 ms Execution Time: 0.997 ms

Justification: improvements in latency and TPS observed after creating the B-tree indexes on the "TrackId" column with different filter conditions and on the "TrackId" column in the "PlaylistTrack" table can be attributed to these optimizations. The reduced disk I/O, faster lookup times, and improved index join and usage pay to the overall improved performance of the query.

Query 5

BEFORE:

latency average = 0.416 ms

tps = 2401.133518 (including connections establishing)

tps = 2401.221727 (excluding connections establishing)

--CREATE INDEX Q5 ON "Track" using btree("TrackId") Where "UnitPrice" !=0.99 ;

The values are: 100000, 21300000, 21300000

AFTER:

latency average = 0.139 ms

tps = 7212.372337 (including connections establishing)

tps = 7213.153127 (excluding connections establishing)

Planning:

Buffers: shared hit=112

Planning Time: 0.218 ms

Execution Time: 0.380 ms

Justification: after applying hash function no crucial difference had happened in the latency time but the latency and TPS observed after creating the B-tree index on the "TrackId" column with the filter condition "UnitPrice" != 0.99 can be attributed to these optimizations. The reduced data access time, improved selectivity, and efficient index usage collectively contribute to the improved performance of the query.

Query 6

BEFORE:

latency average = 1.395 ms

tps = 716.633105 (including connections establishing)

tps = 716.641825 (excluding connections establishing)

--CREATE UNIQUE INDEX Q6 ON "Track"("TrackId") where "UnitPrice" =0.99;

Using hash , btree , brin and unique indices the values are 0,0,0

AFTER:

latency average = 1.365 ms

tps = 732.502058 (including connections establishing)

tps = 732.510630 (excluding connections establishing)

Planning:

Buffers: shared hit=115

Planning Time: 0.228 ms

Execution Time: 0.500 ms

Justification: after trying most of the indexes like btree and hash index, unique indices on the "TrackId" column with the given filter condition did not provide noticeable improvements in latency and TPS metrics. Further analysis and assessment of the specific query, data characteristics, and workload patterns may be necessary to determine more effective indexing strategies for optimizing performance.

Query 7

Before:

latency average = 4.491 ms

tps = 222.654145 (including connections establishing)

tps = 222.654898 (excluding connections establishing)

-- CREATE INDEX Q7 ON "Track" using btree("AlbumId");

-- CREATE INDEX Q77 ON "Album" using btree("ArtistId");

--CREATE INDEX Q777 ON "Album" using btree("AlbumId");

Values for Q7 are: 400000, 400000,0

Values for Q77 are: 200000, 200000, 200000

Values for Q777 are: 400000, 400000, 400000

After:

latency average = 4.576 ms

tps = 218.549451 (including connections establishing)

tps = 218.550219 (excluding connections establishing)

Planning:

Buffers: shared hit=265

Planning Time: 0.673 ms

Execution Time: 6.029 ms

Justification:

Using B+tree indexes on the "AlbumId" column in the "Track" table and the "AlbumId" column in the "Album" table, as well as an index on the "ArtistId" column in the "Album" table, did not result in significant improvements in latency and TPS metrics for the given query. The values before and after creating the indexes are similar, with minimal changes observed.

Query 8

BEFORE:

latency average = 0.370 ms

tps = 2700.813784 (including connections establishing)

tps = 2700.925244 (excluding connections establishing)

--CREATE INDEX Q8 ON "Album" using btree("ArtistId") ;

The values are: 200000, 200000, 200000

AFTER:

latency average = 0.381 ms

tps = 2627.220276 (including connections establishing)

tps = 2627.378844 (excluding connections establishing)

Planning:

Buffers: shared hit=267

Planning Time: 0.711 ms

Execution Time: 0.067 ms

Justification: B-tree index on the "ArtistId" column in the "Album" table has resulted in a slight increase in latency and a decrease in TPS metrics for the given query, like:

\*Index Overhead: Creating an index incurs additional overhead during data modifications, such as insert, update, and delete operations. If the workload involves frequent modifications on the "ArtistId" column, the overhead of maintaining the index may impact the overall performance of the query.

\*Index Utilization: The specific query being executed may not effectively utilize the newly created index. The query optimizer determines the most efficient access path for a given query based on factors such as selectivity, available indexes, and data distribution.

\* The data distribution in the "ArtistId" column and the query pattern can influence the effectiveness of the index.

Query 9

BEFORE:

latency average = 0.871 ms

tps = 1147.704757 (including connections establishing)

tps = 1147.733053 (excluding connections establishing)

--CREATE INDEX Q9 ON "Track" using btree ("AlbumId") ;

The values are: 600000, 5200000, 4800000

AFTER:

latency average = 0.793 ms

tps = 1261.315494 (including connections establishing)

tps = 1261.339899 (excluding connections establishing)

Planning:

Buffers: shared hit=267

Planning Time: 0.639 ms

Execution Time: 1.350 ms

Justification:

The decision to use a B-tree index on the "AlbumId" column in the given query can be justified because Btrees are well-suited for equality and range queries. The index efficiently handles conditions involving the "AlbumId" column, improves data access, and reduces the number of scanned rows. Additionally, Btrees are effective for various data distributions, making them a suitable choice for optimizing query performance. Overall, the Btree index has contributed to the observed improvements in latency and TPS.

Query 10

BEFORE:

latency average = 4.485 ms

tps = 222.989785 (including connections establishing)

tps = 222.990857 (excluding connections establishing)

--CREATE INDEX Q10 ON "Album" using btree("AlbumId") ;

The values are: 400000, 400000, 400000

AFTER:

latency average = 4.417 ms

tps = 226.412452 (including connections establishing)

tps = 226.413208 (excluding connections establishing)

Planning:

Buffers: shared hit=274

Planning Time: 0.594 ms

Execution Time: 6.058 ms

Justification:

Btree index on the "AlbumId" column in the given query can be justified because it improves the performance of equality joins and benefits from the selectivity and data distribution of the column. The Btree index allows for efficient lookup and matching of rows based on the "AlbumId" value, resulting in faster query execution and improved latency and TPS.

Query 11

BEFORE:

latency average = 0.781 ms

tps = 1280.267050 (including connections establishing)

tps = 1280.297318 (excluding connections establishing)

--CREATE INDEX Q11 ON "PlaylistTrack" using btree("TrackId") ;  
The values are: 5200000, 14900000, 14500000

AFTER:

latency average = 0.503 ms

tps = 1986.585147 (including connections establishing)

tps = 1986.645175 (excluding connections establishing)

Planning:

Buffers: shared hit=271

Planning Time: 0.613 ms

Execution Time: 1.348 ms

Justification:

When using a hash index there was not a significant impact on tps and latency, but Btree index on the "TrackId" column in a query can improve lookup, join operations, data distribution, and query optimization. It reduces time taken to locate and retrieve relevant rows, improves query performance, and narrows search space for relevant rows. The effectiveness of an index depends on the data distribution of the indexed column, and the query optimizer may choose to use the index based on selectivity and data distribution. This results in faster query execution and improved performance.

Query 12

BEFORE:

latency average = 0.154 ms

tps = 6492.332259 (including connections establishing)

tps = 6492.950091 (excluding connections establishing)

CREATE INDEX Q12 ON "Customer" using btree("CustomerId") ;

The values are: 200000, 200000, 200000

AFTER:

latency average = 0.139 ms

tps = 7187.981638 (including connections establishing)

tps = 7188.662320 (excluding connections establishing)

Planning:

Buffers: shared hit=156

Planning Time: 0.309 ms

Execution Time: 0.096 ms

Query 13

BEFORE:

latency average = 0.060 ms

tps = 16644.325632 (including connections establishing)

tps = 16648.513261 (excluding connections establishing)

Using hash , btree , unique , partial and brin index the values was 0,0,0

AFTER:

latency average = 0.070 ms

tps = 14320.359331 (including connections establishing)

tps = 14323.540095 (excluding connections establishing)

Planning:

Buffers: shared hit=96

Planning Time: 0.182 ms

Execution Time: 0.034 ms

Query 14

BEFORE:

latency average = 0.287 ms

tps = 3487.319408 (including connections establishing)

tps = 3487.490841 (excluding connections establishing)

-- CREATE INDEX Q14 ON "Customer" using btree("CustomerId");

The values are: 200000, 200000,0

AFTER:

latency average = 0.350 ms

tps = 2854.345761 (including connections establishing)

tps = 2854.589208 (excluding connections establishing)

Planning:

Buffers: shared hit=213

Planning Time: 0.380 ms

Execution Time: 0.341 ms

Query 15

BEFORE:

latency average = 0.206 ms

tps = 4844.940797 (including connections establishing)

tps = 4845.481863 (excluding connections establishing)

TABLE:Invoice

COLUMN:BillingCity

using btree , brin and hash index the output was 0,0,0

Planning:

Buffers: shared hit=94

Planning Time: 0.213 ms

Execution Time: 0.202 ms

Query 16

BEFORE:

latency average = 0.225 ms

tps = 4449.836717 (including connections establishing)

tps = 4450.554994 (excluding connections establishing)

--CREATE INDEX Q16 ON "Invoice" using btree("Total");

The values are: 100000, 6200000, 6200000

AFTER:

latency average = 0.129 ms

tps = 7723.360811 (including connections establishing)

tps = 7724.829701 (excluding connections establishing)

Planning:

Buffers: shared hit=110

Planning Time: 0.200 ms

Execution Time: 0.259 ms

Query 17

BEFORE:

latency average = 0.961 ms

tps = 1040.092081 (including connections establishing)

tps = 1040.123360 (excluding connections establishing)

-- CREATE INDEX Q17 ON "Invoice" using btree("InvoiceId");

Values of Q17 are: 200000, 200000, 200000

AFTER:

atency average = 0.963 ms

tps = 1038.689629 (including connections establishing)

tps = 1038.708581 (excluding connections establishing)

Planning:

Buffers: shared hit=173

Planning Time: 0.357 ms

Execution Time: 1.003 ms

Query 18

BEFORE:

latency average = 0.233 ms

tps = 4288.198059 (including connections establishing)

tps = 4288.514690 (excluding connections establishing)

--CREATE INDEX Q18 ON "Customer" using btree("CustomerId");

The values are: 200000, 200000,0

AFTER:

latency average = 0.250 ms

tps = 4000.508693 (including connections establishing)

tps = 4000.837941 (excluding connections establishing)

Planning:

Buffers: shared hit=181

Planning Time: 0.364 ms

Execution Time: 0.354 ms