Data Flow In Postgres

Common Steps:

1. Debugger run
2. Go to main
3. Error and memory manager init
4. Line 206 Postgres main
5. Postgres.c void PostgresMain line 3813
6. ReadCommand(&input_message) line 4287
7. Opens console psql

Query wise flow
List all databases
Select datname from pg_database;
1. exec_simple_query(query_string) line 4347
2. parsetree_list= pg_parse_query(query_string) line 1035 postgres.c
3. Run through raw parse trees and process each one of them
4. Querytree_list = pg_analyze_and_rewrite(parsetree,query_string,....) line 1154
5. Plantree_list = pg_plan_queries(querytree_list,query_string,..) line 1157
6. PortalRun(portal,....) line 1238 go into for executing parse tree
7. Goes to PortalRunSelect pquery.c line 864
8. ExecutorRun() line 921 pquery.c
9. Goes to execMain.c line 291
10. standard_ExecutorRun(queryDesc,...) line 294 execMain.c
11. ExecutePlan() line 350 execMain.c
12. Goes to line 1s603 in execMain.c
13. Slot = ExecProcNode(planState); line 1632 execMain.c {Runs until proper number of tuples are received}
14. Goes to line 245 executor.h
15. return node->ExecProcNode(node)
16. Goes to NodeSeqScan.c line 110 {ExecSeqScan function}
17. Return ExecScan(...)
18. Goes to line 169 in execScan.c
19. Fetches data from the node
20. If no qual or projection then returns the raw tuple from the flow below
21. Return ExecScanFetch(node,...) line 182 execScan.c
22. Goes to line 39 execScan.c ExecScanFetch
23. Return (*accessMtd) (node) line 133 execScan.c
    a. /*
    b. * Run the node-type-specific access method function to get the next tuple
    c. */
24. Goes to line 60 nodeSeqScan.c for SeqNext function
25. if(table_scan_getnextslot(...)) //get the next tuple from the table

26. Goes to line 905 in tableam.h table_scan_getnextslot
    a. /*
    b. * Return next tuple from `scan`, store in slot.
    c. */
27. return sscan->rs_rd->rd_tableam->scan_getnextslot(sscan, direction, slot);
28. Goes to heapam.c line 1330 to heap_getnextslot()
29. After exiting goes to ExecutePlan() in execMain.c
    a. /*
    b. * Loop until we've processed the proper number of tuples from the plan.
    c. */

To Do:

Schema:

1. AIDX_QUERIES MEET
    a. TYPE
    b. TABLE NAME
    c. ATTRIBUTE
    d. CONDITION
    e. AGGREGATE
    f. JOINS
2. AIDX_BLACKLISTED_QUERIES MEET
3. AIDX_CANDIDATE_QUERIES MEET
4. AIDX_CANDIDATES
5. AIDX_BLACKLISTED
6. AIDX_CREATED

PROPORTIONALITY to BENEFIT

1. SELECT DIRECTLY
2. INSERT DELETE UPDATE INVERSELY
3. TABLE SIZE DIRECTLY SELECT
4. TABLE SIZE INVERSELY INSERT DEL UPDATE
5. SUBSET HASHMAP FOR ATTRIBUTES (STORE BENEFITS FOR EACH SUBSET COLUMNS)
6. CONDITIONS
    a. NON OVERLAPPING BETWEEN READ AND WRITES THEN DEFINITELY CREATE INDEX WITH SOME BOUNDARY CONDITION EG: 0-250 WRITES 450-550 READS 750-1000 WRITES ON TABLE SIZE 1000 THEN CREATE INDEX ON 350-650 WITH 100 BOUNDARY ADJUSTMENT.
    b. HANDLE OVERLAP CASE AFTERWARDS
7. AGGREGATE USUALLY HAS NO BENEFIT DUE TO SEQ SCAN

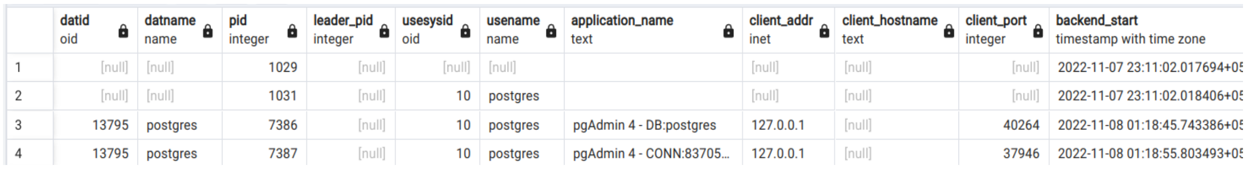
8. JOINS
    a. EQUI JOIN DEFINITELY
    b. INNER JOIN/ NATURAL JOIN TBD
    c. OUTER JOIN NOT DECIDED

REMOVE STALE INDEXES

Queries:

To check running queries:

Select * from pg_stat_activity;



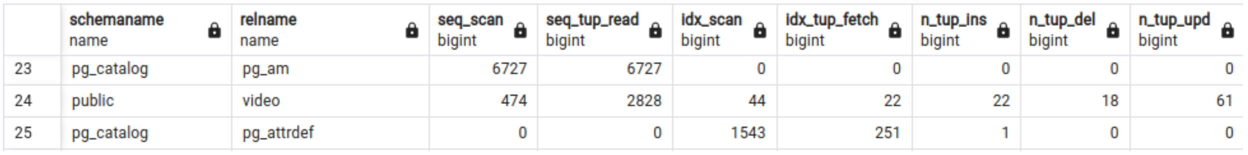
	datid	datname	pid	leader_pid	usesysid	usename	application_name	client_addr	client_hostname	client_port	backend_start
	oid	name	integer	integer	oid	name	text	inet	text	integer	timestamp with time zone
1	[null]	[null]	1029	[null]	[null]	[null]		[null]	[null]	[null]	2022-11-07 23:11:02.017694+05
2	[null]	[null]	1031	[null]	10	postgres		[null]	[null]	[null]	2022-11-07 23:11:02.018406+05
3	13795	postgres	7386	[null]	10	postgres	pgAdmin 4 - DB:postgres	127.0.0.1	[null]	40264	2022-11-08 01:18:45.743386+05
4	13795	postgres	7387	[null]	10	postgres	pgAdmin 4 - CONN:83705...	127.0.0.1	[null]	37946	2022-11-08 01:18:55.803493+05

To check database statistics:

Select * from pg_stat_database;

Check the table statistics:

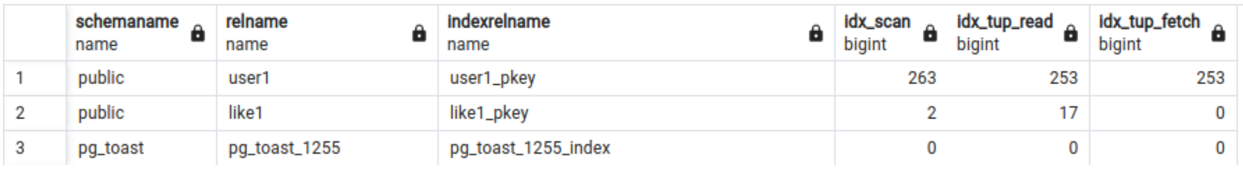
Select * from pg_stat_all_tables;



	schemaname	relname	seq_scan	seq_tup_read	idx_scan	idx_tup_fetch	n_tup_ins	n_tup_del	n_tup_upd
	name	name	bigint	bigint	bigint	bigint	bigint	bigint	bigint
23	pg_catalog	pg_am	6727	6727	0	0	0	0	0
24	public	video	474	2828	44	22	22	18	61
25	pg_catalog	pg_attrdef	0	0	1543	251	1	0	0

Check the index statistics:

Select * from pg_stat_all_indexes;



	schemaname	relname	indexrelname	idx_scan	idx_tup_read	idx_tup_fetch
	name	name	name	bigint	bigint	bigint
1	public	user1	user1_pkey	263	253	253
2	public	like1	like1_pkey	2	17	0
3	pg_toast	pg_toast_1255	pg_toast_1255_index	0	0	0

Check the user function statistics:

Select * from pg_stat_user_functions;