

DROP TABLE IF EXISTS "gin\_test";

CREATE TABLE "gin\_test"("id" bigserial PRIMARY KEY, "string" text, "gin\_vector" tsvector);

INSERT INTO "gin\_test"("string") SELECT substr(characters, (random()\*length(characters)+1)::integer, 10) FROM (VALUES('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) as symbols(characters), generate\_series(1, 1000000) as q;

UPDATE "gin\_test" set "gin\_vector" = to\_tsvector("string");

SELECT COUNT(\*) FROM "gin\_test" WHERE "id" % 2 = 0;

SELECT COUNT(\*) FROM "gin\_test" WHERE ("gin\_vector" @@ to\_tsquery('bnm'));

SELECT SUM("id") FROM "gin\_test" WHERE ("gin\_vector" @@ to\_tsquery('QWERTYUIOP')) OR ("gin\_vector" @@ to\_tsquery('bnm'));

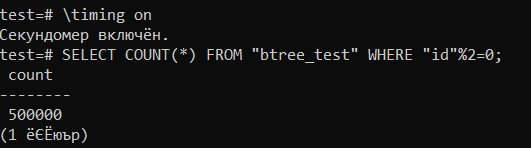
SELECT MIN("id"), MAX("id") FROM "gin\_test" WHERE ("gin\_vector" @@ to\_tsquery('bnm')) GROUP BY "id" % 2;

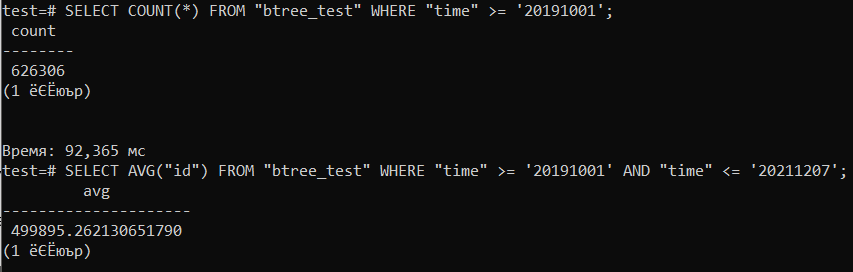
**BTREE**

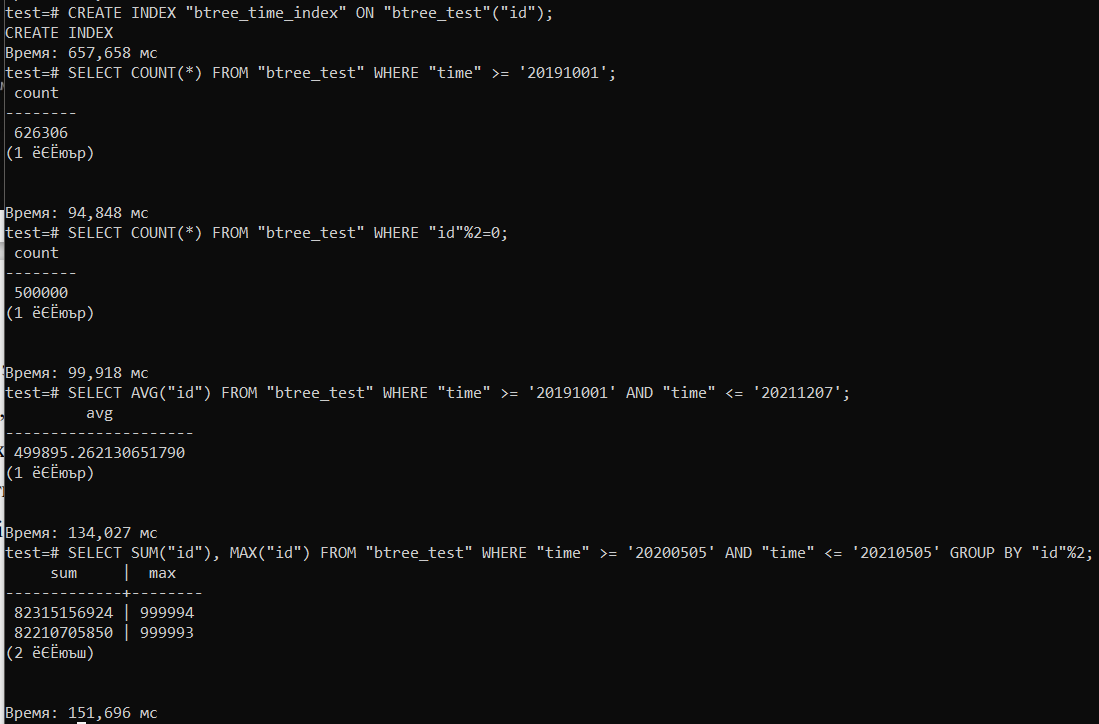
DROP TABLE IF EXISTS "btree\_test";

CREATE TABLE "btree\_test"("id" bigserial PRIMARY KEY, "time" timestamp);

INSERT INTO "btree\_test"("time") SELECT (timestamp '2021-01-01' + random()\*(timestamp '2020-01-01'-timestamp '2022-01-01')) FROM (VALUES('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) as symbols(characters), generate\_series(1, 1000000) as q;







**TRIGGERS**

DROP TABLE IF EXISTS "trigger\_test";

CREATE TABLE "trigger\_test"(

"trigger\_testID" bigserial PRIMARY KEY,

"trigger\_testName" text

);

DROP TABLE IF EXISTS "trigger\_test\_log";

CREATE TABLE "trigger\_test\_log"(

"id" bigserial PRIMARY KEY,

"trigger\_test\_log\_ID" bigint,

"trigger\_test\_log\_name" text

);

INSERT INTO "trigger\_test"("trigger\_testName")

VALUES ('trigger\_test1'), ('trigger\_test2'), ('trigger\_test3'), ('trigger\_test4'), ('trigger\_test5'), ('trigger\_test6'), ('trigger\_test7'), ('trigger\_test8'), ('trigger\_test9'), ('trigger\_test10');

INSERT INTO "trigger\_test\_log"("trigger\_test\_log\_name")

VALUES ('trigger\_test1'), ('trigger\_test2');

CREATE OR REPLACE FUNCTION after\_insert\_update\_func() RETURNS TRIGGER as $trigger$

DECLARE

CURSOR\_LOG CURSOR FOR SELECT \* FROM "trigger\_test\_log";

row\_ "trigger\_test\_log"%ROWTYPE;

BEGIN

IF NEW."trigger\_testID"%2=0 THEN

IF NEW."trigger\_testID"%3=0 THEN

RAISE NOTICE 'trigger\_testID is multiple of 2 and 3';

FOR row\_ IN CURSOR\_LOG LOOP

UPDATE "trigger\_test\_log" SET "trigger\_test\_log\_name"='\_' || row\_."trigger\_test\_log\_name" || '\_log' WHERE "id"=row\_."id";

END LOOP;

RETURN NEW;

ELSE

RAISE NOTICE 'trigger\_testID is even';

INSERT INTO "trigger\_test\_log"("trigger\_test\_log\_ID", "trigger\_test\_log\_name") VALUES (NEW."trigger\_testID", NEW."trigger\_testName");

UPDATE "trigger\_test\_log" SET "trigger\_test\_log\_name" = trim(BOTH '\_log' FROM "trigger\_test\_log\_name");

RETURN NEW;

END IF;

ELSE

RAISE NOTICE 'trigger\_testID is odd';

FOR row\_ IN CURSOR\_LOG LOOP

UPDATE "trigger\_test\_log" SET "trigger\_test\_log\_name" = '\_' || row\_."trigger\_test\_log\_name" || '\_log' WHERE "id" = row\_."id";

END LOOP;

RETURN NEW;

END IF;

END;

$trigger$ LANGUAGE plpgsql;

CREATE TRIGGER "after\_insert\_update\_trigger"

AFTER INSERT OR UPDATE ON "trigger\_test"

FOR EACH ROW

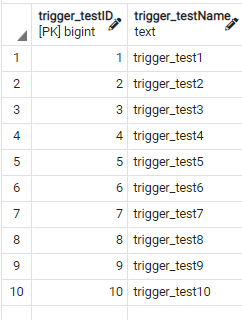
EXECUTE procedure after\_insert\_update\_func();

INSERT INTO "trigger\_test"("trigger\_testName")

VALUES ('trigger\_test11'), ('trigger\_test12');

INSERT INTO "trigger\_test\_log"("trigger\_test\_log\_name")

VALUES ('trigger\_test\_1'), ('trigger\_test2');



**UPDATED CODE**

DROP TABLE IF EXISTS "trigger\_test";

CREATE TABLE "trigger\_test"(

"trigger\_testID" bigserial PRIMARY KEY,

"trigger\_testName" text

);

DROP TABLE IF EXISTS "trigger\_test\_log";

CREATE TABLE "trigger\_test\_log"(

"id" bigserial PRIMARY KEY,

"trigger\_test\_log\_ID" bigint,

"trigger\_test\_log\_name" text

);

INSERT INTO trigger\_test("trigger\_testName") VALUES ('test1'), ('test2'), ('test3'), ('test4'), ('test5'), ('test6');

CREATE OR REPLACE FUNCTION after\_insert\_func() RETURNS TRIGGER AS $trigger$

DECLARE

CURSOR\_LOG CURSOR FOR SELECT \* FROM "trigger\_test\_log";

row\_ "trigger\_test\_log"%ROWTYPE;

BEGIN

IF NEW."trigger\_testID"%2=0 THEN

IF NEW."trigger\_testID"%3=0 THEN

RAISE NOTICE 'trigger\_testID is multiple of 2 and 3';

FOR row\_ IN CURSOR\_LOG LOOP

-- UPDATE "trigger\_test\_log" SET "trigger\_test\_log\_name"='\_' || row\_."trigger\_test\_log\_name" || '\_log' WHERE "id"=row\_."id";

INSERT INTO "trigger\_test\_log"("trigger\_test\_log\_ID", "trigger\_test\_log\_name") VALUES (NEW."trigger\_testID", NEW."trigger\_testName");

END LOOP;

RETURN NEW;

ELSE

RAISE NOTICE 'trigger\_testID is even';

INSERT INTO "trigger\_test\_log"("trigger\_test\_log\_ID", "trigger\_test\_log\_name") VALUES (NEW."trigger\_testID", NEW."trigger\_testName");

UPDATE "trigger\_test\_log" SET "trigger\_test\_log\_name" = trim(BOTH '\_log' FROM "trigger\_test\_log\_name");

RETURN NEW;

END IF;

ELSE

RAISE NOTICE 'trigger\_testID is odd';

FOR row\_ IN CURSOR\_LOG LOOP

UPDATE "trigger\_test\_log" SET "trigger\_test\_log\_name" = '\_' || row\_."trigger\_test\_log\_name" || '\_log' WHERE "id" = row\_."id";

END LOOP;

RETURN NEW;

END IF;

END;

$trigger$ LANGUAGE plpgsql;

CREATE TRIGGER after\_insert\_test

AFTER INSERT OR UPDATE ON "trigger\_test"

FOR EACH ROW EXECUTE PROCEDURE after\_insert\_func();

-- INSERT INTO trigger\_test("trigger\_testName") VALUES ('test1'), ('test2'), ('test3'), ('test4'), ('test5'), ('test6');

UPDATE "trigger\_test" SET "trigger\_testName" = "trigger\_testName" || '\_2' WHERE "trigger\_testID"%2=0

**TRANSACTIONS**

DROP TABLE IF EXISTS "transactions";

CREATE TABLE "transactions"(

"id" bigserial PRIMARY KEY,

"numeric" bigint,

"text" text

);

INSERT INTO "transactions"("numeric", "text") VALUES (222, 'string1'), (223, 'string2'), (224, 'string3');

