**Try different kind of indexes (B-tree, Hash, GIN, GIST) for your fields. Analyze performance for each of the indexes (use ANALYZE and EXPLAIN). Check the size of the index. Try to set index before inserting test data and after. What was the time?**

|  |  |
| --- | --- |
|  | **B-tree** |
| a. Find user by name (exact match)  - 100K of users  EXPLAIN ANALYZE  SELECT \* FROM epam\_training.students WHERE first\_name='FN1'; | 1. Before adding data   EXPLAIN ANALYZE SELECT \* FROM epam\_training.students WHERE first\_name='FN1';    After adding data, before adding index    After adding index: CREATE INDEX ix\_first\_name ON epam\_training.students USING btree (first\_name);    ***Execution time lower after adding the index***  drop INDEX epam\_training.ix\_first\_name ; |
| b. Find user by surname (partial match)  - 100K of users  EXPLAIN ANALYZE SELECT \* FROM epam\_training.students WHERE last\_name like '%Last%'; | Before adding index    After adding index  CREATE INDEX ix\_last\_name ON epam\_training.students USING btree (last\_name);    ***No performance improvement***  drop INDEX epam\_training.ix\_last\_name ; |
| c. Find user by phone number (partial match)  - 100K of users  EXPLAIN ANALYZE SELECT \* FROM epam\_training.students WHERE phone\_number like '%123%'; | Before adding index    After adding index  CREATE INDEX ix\_phone ON epam\_training.students USING btree (phone\_number);    ***No performance improvement***  drop INDEX epam\_training.ix\_phone ; |
| d. Find user with marks by user surname (partial match)  - 100K of users  - 1K of subjects  - 1 million of marks  explain analyse SELECT \* FROM epam\_training.students where id in (select distinct student\_id from epam\_training.exam\_results where mark > 0) and last\_name like '%0%' | Before adding index    After adding indexes  CREATE INDEX ix\_last\_name ON epam\_training.students USING btree (last\_name);  CREATE INDEX ix\_mark ON epam\_training.exam\_results USING btree (mark);    ***No performance improvement***  drop INDEX epam\_training.ix\_last\_name ;  drop INDEX epam\_training.ix\_mark; |

|  |  |
| --- | --- |
|  | **Hash** |
| a. Find user by name (exact match)  - 100K of users | After adding index: CREATE INDEX ix\_first\_name ON epam\_training.students USING hash  (first\_name);    ***Execution time lower after adding the index, faster than B-tree***  drop INDEX epam\_training.ix\_first\_name ; |
| b. Find user by surname (partial match)  - 100K of users | CREATE INDEX ix\_last\_name ON epam\_training.students USING hash (last\_name);    ***No performance improvement***  drop INDEX epam\_training.ix\_last\_name ; |
| c. Find user by phone number (partial match)  - 100K of users | CREATE INDEX ix\_phone ON epam\_training.students USING hash (phone\_number);    ***No performance improvement***  drop INDEX epam\_training.ix\_phone ; |
| d. Find user with marks by user surname (partial match)  - 100K of users  - 1K of subjects  - 1 million of marks | CREATE INDEX ix\_last\_name ON epam\_training.students USING hash (last\_name);  CREATE INDEX ix\_mark ON epam\_training.exam\_results USING hash (mark);    ***No performance improvement***  drop INDEX epam\_training.ix\_last\_name ;  drop INDEX epam\_training.ix\_mark; |

|  |  |
| --- | --- |
|  | **GIN** |
| a. Find user by name (exact match)  - 100K of users | Added extensions:  CREATE EXTENSION pg\_trgm;  CREATE EXTENSION btree\_gin;  CREATE INDEX ix\_first\_name ON epam\_training.students USING gin(first\_name);    ***Execution time lower after adding the index***  drop INDEX epam\_training.ix\_first\_name ; |
| b. Find user by surname (partial match)  - 100K of users | CREATE INDEX ix\_last\_name ON epam\_training.students USING gin (last\_name);    ***No performance improvement***  drop INDEX epam\_training.ix\_last\_name ; |
| c. Find user by phone number (partial match)  - 100K of users | CREATE INDEX ix\_phone ON epam\_training.students USING gin (phone\_number);    ***No Performance improvement***  drop INDEX epam\_training.ix\_phone ; |
| d. Find user with marks by user surname (partial match)  - 100K of users  - 1K of subjects  - 1 million of marks | CREATE INDEX ix\_last\_name ON epam\_training.students USING gin (last\_name);  CREATE INDEX ix\_mark ON epam\_training.exam\_results USING gin (mark);    ***No performance improvement***  drop INDEX epam\_training.ix\_last\_name ;  drop INDEX epam\_training.ix\_mark; |

|  |  |
| --- | --- |
|  | **GIST** |
| a. Find user by name (exact match)  - 100K of users | CREATE EXTENSION btree\_gist;  CREATE INDEX ix\_first\_name ON epam\_training.students USING gist (first\_name);    ***Execution time lower after adding the index***  drop INDEX epam\_training.ix\_first\_name ; |
| b. Find user by surname (partial match)  - 100K of users | CREATE INDEX ix\_last\_name ON epam\_training.students USING gist (last\_name);    ***No performance improvement***  drop INDEX epam\_training.ix\_last\_name ; |
| c. Find user by phone number (partial match)  - 100K of users | CREATE INDEX ix\_phone ON epam\_training.students USING gist (phone\_number);    ***Performance improvement 1 ms***  drop INDEX epam\_training.ix\_phone ; |
| d. Find user with marks by user surname (partial match)  - 100K of users  - 1K of subjects  - 1 million of marks | CREATE INDEX ix\_last\_name ON epam\_training.students USING gist (last\_name);  CREATE INDEX ix\_mark ON epam\_training.exam\_results USING gist (mark);    ***No performance improvement, 2x slower than without indexes***  drop INDEX epam\_training.ix\_last\_name ;  drop INDEX epam\_training.ix\_mark; |

**Index size comparison:**

CREATE INDEX ix\_first\_name\_btree ON epam\_training.students USING btree (first\_name);

CREATE INDEX ix\_first\_name\_hash ON epam\_training.students USING hash (first\_name);

CREATE INDEX ix\_first\_name\_gin ON epam\_training.students USING gin (first\_name);

CREATE INDEX ix\_first\_name\_gist ON epam\_training.students USING gist (first\_name);

SELECT

t.schemaname,

t.tablename,

c.reltuples::bigint AS num\_rows,

pg\_size\_pretty(pg\_relation\_size(c.oid)) AS table\_size,

psai.indexrelname AS index\_name,

pg\_size\_pretty(pg\_relation\_size(i.indexrelid)) AS index\_size

FROM

pg\_tables t

LEFT JOIN pg\_class c ON t.tablename = c.relname

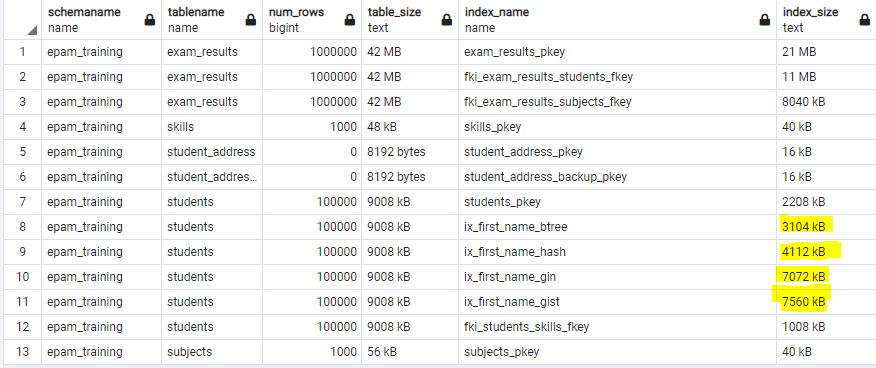
LEFT JOIN pg\_index i ON c.oid = i.indrelid

LEFT JOIN pg\_stat\_all\_indexes psai ON i.indexrelid = psai.indexrelid

WHERE

t.schemaname NOT IN ('pg\_catalog', 'information\_schema')

ORDER BY 1, 2;



Indexes added the performance improvement for exact match search, the bast result is for Hash index, however its size is bigger than B-tree index size.

Partial match search is slightly improved by gist index, however its size is the biggest.

**5. Add trigger that will update column updated\_datetime to current date in case of updating any of student. (0.3 point)**

CREATE FUNCTION epam\_training.update\_time() RETURNS trigger AS $update\_time$

BEGIN

IF OLD.\* IS DISTINCT FROM NEW.\* THEN

NEW.updated\_date := current\_timestamp;

END IF;

RETURN NEW;

END;

$update\_time$ LANGUAGE plpgsql;

CREATE TRIGGER update\_time BEFORE INSERT OR UPDATE ON epam\_training.students

FOR EACH ROW EXECUTE PROCEDURE epam\_training.update\_time();

**6. Add validation on DB level that will check username on special characters (reject student name with next characters '@', '#', '$'). (0.3 point)**

CREATE FUNCTION epam\_training.check\_first\_name() RETURNS trigger AS $check\_first\_name$

BEGIN

IF position('@' in NEW.first\_name)>0 or position('#' in NEW.first\_name)>0 or position('$' in NEW.first\_name)>0 THEN

RAISE EXCEPTION 'Student name cannot contain @,#,$ symbols';

END IF;

RETURN NEW;

END;

$check\_first\_name$ LANGUAGE plpgsql;

CREATE TRIGGER check\_first\_name BEFORE INSERT OR UPDATE ON epam\_training.students

FOR EACH ROW EXECUTE PROCEDURE epam\_training.check\_first\_name();

**7. Create snapshot that will contain next data: student name, student surname, subject name, mark (snapshot means that in case of changing some data in source table – your snapshot should not change). (0.3 point)**

CREATE VIEW epam\_training.marks AS

SELECT s.first\_name,s.last\_name, sub.name, er.mark

FROM epam\_training.students s

left join epam\_training.exam\_results er on s.id=er.student\_id left join epam\_training.subjects sub on sub.id = er.subject\_id;

From pgsql bin folder:

pg\_dump epam\_training -t epam\_training.marks > outfile

**8. Create function that will return average mark for input user. (0.3 point)**

CREATE FUNCTION epam\_training.give\_average(uname TEXT)

RETURNS NUMERIC AS $$

DECLARE average NUMERIC;

BEGIN

SELECT AVG (er.mark) INTO average

FROM epam\_training.students st left join epam\_training.exam\_results er on st.id = er.student\_id where concat (st.first\_name, ' ', st.last\_name) = uname;

RETURN average;

END;

$$ LANGUAGE plpgsql

Check:

SELECT epam\_training.give\_average(

'FirstName2 LastName0'

)

**9. Create function that will return avarage mark for input subject name. (0.3 point).**

CREATE FUNCTION epam\_training.give\_average\_subj(subj TEXT)

RETURNS NUMERIC AS $$

DECLARE average NUMERIC;

BEGIN

SELECT AVG (er.mark) INTO average

FROM epam\_training.subjects sub left join epam\_training.exam\_results er on sub.id = er.subject\_id where sub.name = subj;

RETURN average;

END;

$$ LANGUAGE plpgsql

Check:

SELECT epam\_training.give\_average\_subj(

'Subject2'

)

**10. Create function that will return student at "red zone" (red zone means at least 2 marks <=3). (0.3 point)**

select student\_id from epam\_training.exam\_results where mark <= 3 group by student\_id having count(mark) > 2 order by student\_id;

**11. Extra point (1 point). Show in tests (java application) transaction isolation phenomena. Describe what kind of phenomena is it and how did you achieve it.**

**-**

**12. Extra point 2 (1 point). Implement immutable data trigger. Create new table student\_address. Add several rows with test data and do not give acces to update any information inside it. Hint: you can create trigger that will reject any update operation for target table, but save new row with updated (merged with original) data into separate table.**

CREATE OR REPLACE FUNCTION epam\_training.immutable\_table() RETURNS trigger AS $immutable\_table$

BEGIN

IF NEW.id IS NOT NULL THEN

CREATE TABLE IF NOT EXISTS epam\_training.student\_address\_backup (

id integer NOT NULL ,

address character varying COLLATE pg\_catalog."default",

student\_id integer,

CONSTRAINT student\_address\_backup\_pkey PRIMARY KEY (id),

CONSTRAINT students\_addresses\_backup\_fkey FOREIGN KEY (student\_id)

REFERENCES epam\_training.students (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

);

INSERT INTO epam\_training.student\_address\_backup(id, address, student\_id)

SELECT id, address, student\_id FROM epam\_training.student\_address

ON CONFLICT ON CONSTRAINT student\_address\_backup\_pkey DO NOTHING;

INSERT INTO epam\_training.student\_address\_backup(

id, address, student\_id)

VALUES (NEW.id, NEW.address, NEW.student\_id) ON CONFLICT ON CONSTRAINT student\_address\_backup\_pkey DO NOTHING;

END IF;

RETURN NULL;

END;

$immutable\_table$ LANGUAGE plpgsql;

CREATE TRIGGER immutable\_table BEFORE INSERT OR UPDATE ON epam\_training.student\_address

FOR EACH ROW EXECUTE PROCEDURE epam\_training.immutable\_table();