Automatic Creation of Indices

CS 631 Project

Jitesh Gawas - 22m0748

Meet Doshi - 22m0742

Gaurang Athavale - 22m0774

Under Guidance of Prof. S. Sudarshan

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

26th November, 2022

Contents

1 Introduction 1

2 Index Creation Overview 2

2.1 Need for Creation of Index 2

2.2 Managing the Indexes 2

2.3 Objective 2

3 Automatic Creation of Indices in PostgreSQL 3

3.1 Goals of the project 3

3.2 Methodology 3

3.3 Example 4

4 Future Scope 5

5 References 6

1. Introduction

Indexes help to improve the performance of the databases. An index helps in faster retrieving of the tuples than it could do without an index. But indexes should be created sensibly else it can affect the performance of the databases. In this project, we intend to create an automatic index which would be beneficial in improving the performance of the database.

2. Index Creation Overview

2.1 Need for Creation of Index

In normal circumstances, without any index creation, the system has to sequentially scan all the tuples in the relation. If there are a lot of tuples in the relation and only a few tuples are matching the condition of the query, then it would be very inefficient. In such cases, index scan works far better than the sequential scan.

2.2 Managing the Indexes

Indexes can also work well with update and Delete commands. When an index is created, it has to be kept updated with the status of the table. This can result in overhead. Thus indexes which are not used or which affect the performance of the system should be deleted and the ones which benefit the system should be added.

2.3 Objective

We are focusing on creating automatic indexes which would benefit the system. If the tuples are accessed via sequential scan, then all the tuples need to be checked. But if we create an index , then only fewer tuples need to be accessed. We monitor the queries and keep a track of the tables and the attributes the query is working upon and then create an index based on conditions satisfied by the attributes and tables.

3. Automatic Creation of Indices in PostgreSQL

3.1 Goals of the project:

- **a.** Monitor incoming and outgoing queries on the database and keep track of the important ones.
- **b.** Create rules which determine benefit over cost for creating and maintaining an index.
- **c.** Use good statistics from the database to help identify the type of data stored for the index.

3.2 Methodology:

1.Modifications required to the following files:

- **a.** **nodeSeqScan.c** - For inserting the user queries into the AIDX_Queries table (only for select and delete queries).
- **b.** **execMain.c** - For inserting the user queries into the AIDX_Queries table (only for insert queries).
- **c.** **worker_spi.c** - New daemon process which actually created the indices. Fetching tuple by tuple, from AIDX_Queries table.

2. Created a new daemon process (worker_spi.c)

3. For maintenance we are invoking the daemon process which will be timed based and will check the logs to create the indices.

4. For creation of index used the following logic:

- **a.** Created a new table for tracking (AIDX_Queries).
- **b.** Ran the worker_spi.c daemon and postgres.c in parallel.
- **c.** Inserted the user queries into a new table AIDX_Queries.
- **d.** For fetching the type, table name, attributes and qualifiers of the query, we used nodeSeqScan.c
    - i. Atrributes: scanstate->ss.ss_ScanTupleSlot->tts_tupleDescriptor->attrs
    - ii. Qualifiers: scanstate->ss.ps.qual->steps->d.var.attnum
    - iii. Command Type: estate->es_plannedstmt->commandType
    - iv. Table name: scanstate->ss.ss_currentRelation->rd_rel->relname.data
- **e.** Once the new_query (insert query) is ready using the above information, it is passed on to SPI_execute() for processing, and the query is now inserted to the AIDX_Queries table.
- **f.** Fetched the tuples in worker_spi.c
- **g.** Created index for each tuple fetched from AIDX_Queries.

3.3 Example:

- User sends a query: SELECT * FROM STUDENT WHERE id = 1
- It is first stored into table AIDX_Queries

type varchar(100)	tablename varchar(100)	attr varchar(100)
SELECT	STUDENT	id
DELETE	STUDENT	*
DELETE	STUDENT	id
INSERT	STUDENT	*

  Other examples:
  Row 2: DELETE FROM STUDENT;
  Row 3: DELETE FROM STUDENT WHERE id = 1;
  Row 4: INSERT INTO STUDENT VALUES(1);

- This table is then fetched and all the tuples of the table are retrieved and indices are created using the following query:
  - **CREATE INDEX IF NOT EXISTS AIDX_Queries_index on AIDX_Queries(type);**

4. Future Scope

a. Scaling it over joins and complex queries involving subqueries.

b. Deciding whether creation of the index would benefit the queries involving select, delete and insert statements individually.

c. This decision would be based on few factors:
    i. Type of the query
    ii. Size of the table on which query is executed
    iii. Type of joins
    iv. Type of where condition

d. To check the amount of benefit, some statistics tables can be used, eg, pg_stat_database; pg_stat_all_tables; pg_stat_all_indexes; etc.

e. To extend creation and deletion of indices based on update, insert and delete queries.

References

[1] https://www.postgresql.org/docs/current/spi.html

[2] https://doxygen.postgresql.org/

[3] https://oracle-base.com/articles/19c/automatic-indexing-19c