AUTHOR’S NAME

Zeeshan Anwar (160931)

Zoya Nadeem (161000)

Materialized Views vs Non-Materialized Views in Contemporary Database Management Systems

Bachelor of Science in Computer Science

Supervisor: Shoaib Malik

Co-Supervisor: Dr Tauseef

Department of Computer Science
Air University, Islamabad

February 2020

Certificate

We accept the work contained in the report titled “MATERIALIZED VIEWS VS NON-MATERIALIZED VIEWS IN CONTEMPORARY DATABASES”, written by Mr. Zeeshan Anwar AND Miss Zoya Nadeem as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: Name of the Supervisor (Title)

____

Internal Examiner: Name of the Internal Examiner (Title)

____

External Examiner: Name of the External Examiner (Title)

____

Project Coordinator: Name of the Project Coordinator (Title)

____

Head of the Department: Name of the HOD (Title)

____

June18th , 2020

Abstract

As the world is getting more and more modernized. Data in the previous form of papers and documents are now being converted into digital form. Day by day the data is increasing in size. As the data size increases the retrieval of some particular is getting more and more complex. People are getting more agitated as it takes a lot of time to retrieve that data. To overcome these problems views were generated in databases. Views are defined as a virtual table. It is basically a result of a stored query on the data.

As the data started growing larger and larger. Suddenly views were no longer adaptable in large data warehouses. To tackle this problem Materialized Views were introduced. Materialized views are also a database object which contains the result of a query. Materialized view better than a normal view as its approach is rather different. The query result is cached as a concrete table that may be updated from the original base tables from time to time.

In this report we will talk about Materialized views and Non-Materialized views in contemporary databases to determine which approach is better for the same dataset. We will present a method in which query response time are shown in different form of visualizations as to get a better understanding.

We will perform multiple experiments in different database management systems to determine which system is more efficient for the same dataset and the same set of queries. We will compare materialized and non-materialized views for the same query and determine which query responses faster than the other.

We will display our results in the form of tables and graph to get a better understanding of the comparison.

Acknowledgments

We would first of all like to thank Almighty Allah for giving us the strength and opportunity to achieve the objective that we had set. We would like to thank our parents without whom it would be impossible to be here. We would like to acknowledge the support that has been given to us by our supervisor Sir Shoaib Malik for his patient and continuous guidance. We would like to show our gratitude to the faculty members of our prestigious Computer Science department for their help when we required it.

Muhammad Zeeshan
Zoya Nadeem
Islamabad, Pakistan

June 2020

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

SQL	Structured Query Language
GUI	Graphical User Interface

Chapter 1

Introduction

1.1 Project Background/Overview:

To understand views, we must first talk about databases. A database is an organized collection of data, generally stored and accessed electronically from a computer system. For the retrieval of the data stored in those databases. Many languages were introduced. The language we focused on is STRUCTRED QUERY LANGUAGE (SQL). As the data grew larger and larger it became more difficult to retrieve that data efficiently and optimally.

A data warehouse is a very large database system that collects, summarizes and stores data from multiple remote and heterogeneous information sources. The decision-making queries are complex in nature and take a large amount of time when they are run against a large data warehouse. To reduce the time and cost of the queries **views** are generated.

Traditionally, views are “virtual”. The database system stores their definition queries but not their contents. Virtual views are often used to control access and provide alternative interfaces to base tables. They also support logical data independence: when the base table schema changes, views can be redefined to use the new schema, so application queries written against these views will continue to function. Over the years, however, the concept and practice of materialized views have steadily gained importance.

In a database, a view is the result set of stored queries on the data. A view does not form part of the physical schema. As a resultant set, it is a virtual table computed dynamically from data in the databases when access to the view is requested.

As the data size increases the simple views generated are not enough. For that purpose, a new method was first introduced by Oracle. That method is known as materializing the views. Materialized views are defined as the database objects that contains the result of the query. The process of setting up a materialized view is often called view materialization. The query result is cached as a concrete table that may be updated from original base tables from time to time. Meaning the materialized views are physically stored in databases.

Materialized views are physical structures in databases and are smaller in size. These views are usually created by database administrator which means the quality of views is dependent on the database administrator’s experience and intuition.

We materialize a view by storing its contents. Once materialized, a view can facilitate queries that use it (or can be rewritten to use it), when the base tables are expensive or even unavailable for access.

1.2 Problem Description:

Queries posed against a large data warehouse, are typically analytic, intricate and recurring in nature. Such queries contain a lot of join operations over large volumes of records. To minimize the cost of these queries views are introduced into the system. Materialized views are an updated version of these database views. The main objective of using views is to minimize the cost of a query.

Since a materialized view is an advance form a view it is extremely fast retrieval of aggregate data, since it is precomputed and stored, at the expense of insert/update/delete. The database will keep the Materialized View in sync with the real data.

1.3 Project Objectives:

Following are the objectives and goals of this project:

- Selecting various database management systems (MySQL, POSTGRES etc.).
- Gathering the dataset from a benchmark (TPC-DS).
- Loading the dataset into the management systems.
- Generating materialized and non-materialized views programmatically using python programming language.
- Calculating the runtime (query response time) using various select statements.
- Displaying the results graphically using bar graphs.

1.4 Project Scope:

This project aims to analyze the runtime of the queries posed against the same dataset. Firstly, we selected 4 database management systems (MySQL, POSTGRES, SQL ANYWHERE, ORACLE SQL). After selecting these database management systems, data is loaded into these individually.

After loading the dataset, the management systems were connected to a programming language(python) via a connection string. The materialized views and non-materialized views were generated programmatically for each of these database management systems.

These views were executed programmatically using select statements and the run time (query response time) was calculated by using the inbuilt functions of the programming language. These results will then be displayed graphically using bar graphs for a better understanding.

1.5 Project Life Cycle:

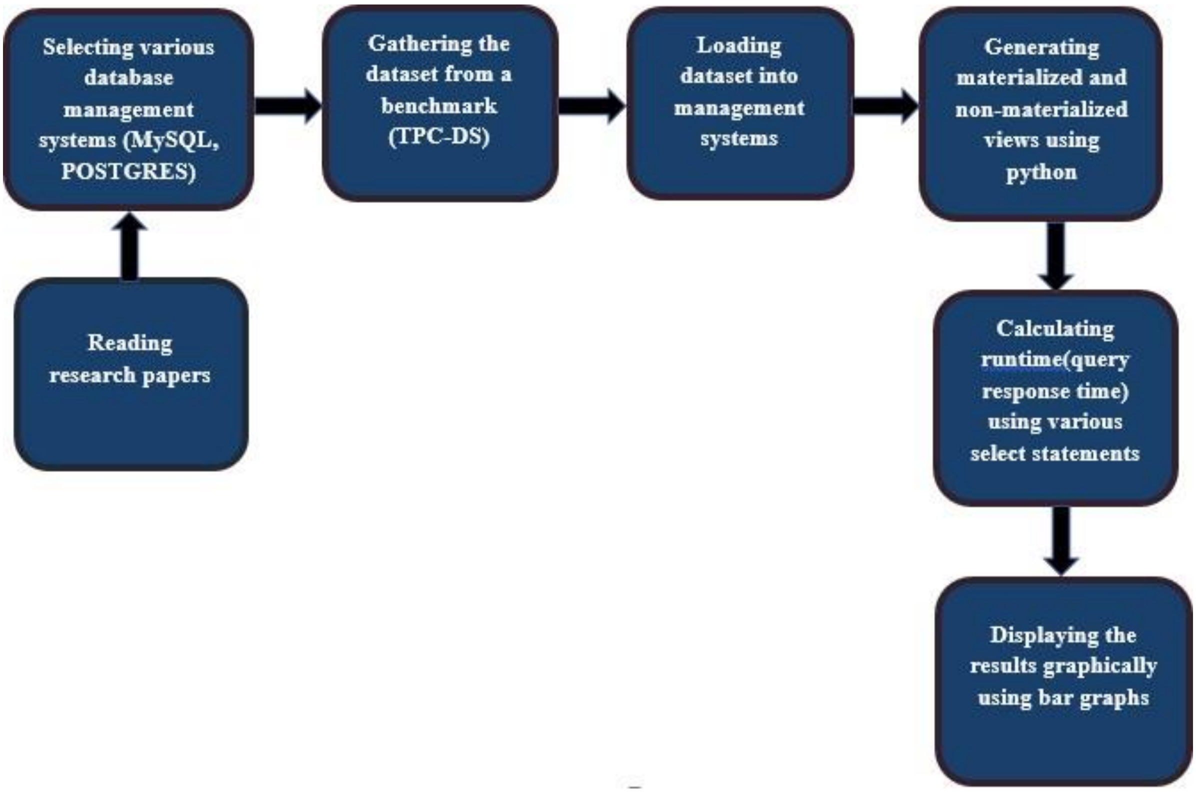


Figure 1.1: Final Year Project Lifecycle

Chapter 2

Literature Review

Our FYP focusses on views in contemporary database management systems. For this purpose, we studied various research papers that are related to materialized and non-materialized views.

2.1 Review of Exiting Research Papers:

2.1.1 Optimizing Queries Using Materialized Views: A Practical, Scalable Solution (Jonathan Goldstein & Larson, 2001)

This paper talks about how materialized views can provide massive improvements in query processing time, especially for aggregated queries over larger tables. This paper performed experiments on TPC-R benchmark. Experimental results based on an implementation in Microsoft SQL Server showed excellent results and very fast. With 1000 views in the system, average optimization time remained as low as 0.15 seconds per query.

2.1.2 Optimizing Queries with Materialized Views (Sura jit Chaudhurix, Ravi Krishnamurthyy & Spyros Potamianosz, 2001)

This paper talks about of maintaining materialized views. Since all views cannot be materialized as it will take enormous space. This paper analyses the optimization problem and provides a comprehensive and efficient solution which reduces the cost of queries.

2.1.3 Materialized Views (Rada Chirkova & Jun Yang, 2012)

Rada chirkova and Jun Yang in their paper "Materialized Views" talk about how In SQL settings the materialized views are considered a mature technology implemented by commercial database systems and tightly integrated into query optimization. They have covered 3 fundamental problems: Maintaining views efficiently when base tables change, using materialize views to improve performance and availability, selecting which views to materialize.

2.2 Survey Table:

Papers	Dataset	System
2.1.1	TPC-R (benchmark)	SQL
2.1.2	Self-made dataset employee relation Emp (name, dno, sal, age) and a department relation Dept (dno, size, loc)	SQL
2.1.3	Self-made dataset Retailer pos (itemID, storeID, date) stores (storeID, city, region) items (itemID, name, cost)	SQL

Table 1.1: Survey

2.3 Conclusion:

To conclude this chapter a lot of work has been done using materialized and non-materialized views. Since materialized views have been introduced a lot of experiments have been done and their accuracy vary based on their systems. We will be performing these experiments on the same system but different database management systems to show which management system stands out. Which management system works better with materialized and non-materialized views.

Chapter 3

Requirement Specifications

Since our FYP is researched base. We had to study numerous research papers and documentations in order to find the best research till the time. Studying these research papers extensively gave us an idea as to how we should approach our problem and how can we use the best tools in order to solve it.

3.1 Existing System:

Out of the research performed by me and my group partner we picked one research paper that benefited us the most. Materialized Views (Rada Chirkova & Jun Yang, 2011) gave us the best idea how we can approach our problem. In this paper they performed their experiments using SQL. The research covered three main factors “maintaining materialized views efficiently when the base table changes.” “using materialized views effectively to improve performance.” “selecting which views to materialize”. Since this research was done on a smaller scale the decision of which views to materialize were made by a database administrator. These experiments were performed in Microsoft SQL server.

3.2 Proposed specifications:

Since in our base paper only used Microsoft SQL Server. We expanded our experiment to more database management systems. In order to discover the best database management system for materialized and non-materialized views. Our research will consist of a vast difference between these management systems and we will be able to distinguish the best system amongst them. Since all of these experiments will be conducted on a single operating system. The results will be unbiased.

3.3 Data Collection:

The dataset that we will be using for our experiment was gathered from tpc.org. The name of the dataset is **TPC-DS.**

3.3.1 TPC-DS:

TPC-DS is a decision support benchmark. It models several generally applicable aspects of a decision support system, including queries and data maintenance. TPC-DS is one of the first benchmarks of the Tpc benchmark which supports the idea of views.

3.3.2 Data Format:

- The data was gathered from https://github.com/databricks/tpcds-kit.
- The data was generated in .dat files.
- The data generated was then converted into csv to import it into the database management systems.
- The scale factor used for this dataset was “SF=1” which is about 1GB of data.

3.3.3 Data Sample:

A data sample of the table item.

	i_item_sk [PK] integer	i_item_id character (16)	i_rec_start_date character varying (255)	i_rec_end_date character varying (255)	i_item_desc character varying (200)	i_current_price numeric (7,2)	i_wholesale_cost numeric (7,2)	i_br inte
1	1	AAAAAAAABAAAA...	27/10/1997	[null]	Powers will not get influences...	27.02	23.23	
2	2	AAAAAAAACAAAA...	27/10/1997	26/10/2000	False opportunities would run ...	1.12	0.38	
3	3	AAAAAAAACAAAA...	27/10/2000	[null]	False opportunities would run ...	7.11	0.38	
4	4	AAAAAAAAEAAAA...	27/10/1997	27/10/1999	Normal systems would join si...	1.35	0.85	
5	5	AAAAAAAAEAAAA...	28/10/1999	26/10/2001	Normal systems would join si...	4.00	1.76	
6	6	AAAAAAAAEAAAA...	27/10/2001	[null]	Normal systems would join si...	0.85	1.76	
7	7	AAAAAAAAHAAAA...	27/10/1997	[null]	Anxious accounts must catch...	9.94	6.75	
8	8	AAAAAAAAIAAAA...	27/10/1997	26/10/2000	F	2.76	0.85	
9	9	AAAAAAAAIAAAA...	27/10/2000	[null]	F	4.46	0.85	
10	10	AAAAAAAAKAAAA...	27/10/1997	27/10/1999	Classical services go trousers...	8.94	4.11	
11	11	AAAAAAAAKAAAA...	28/10/1999	26/10/2001	Correct, fo	54.87	4.11	
12	12	AAAAAAAAKAAAA...	27/10/2001	[null]	Corporate, important facilities...	6.54	4.11	
13	13	AAAAAAAANAAAA...	27/10/1997	[null]	Hard, private departments sp...	8.76	7.62	
14	14	AAAAAAAAOAAAA...	27/10/1997	26/10/2000	Teachers carry by the children...	1.85	0.59	
15	15	AAAAAAAAOAAAA...	27/10/2000	[null]	Teachers carry by the children...	2.57	0.59	
16	16	AAAAAAAAABAAA...	27/10/1997	27/10/1999	Dominant, christian pp. may n...	0.31	0.14	
17	17	AAAAAAAAABAAA...	28/10/1999	26/10/2001	Dominant, christian pp. may n...	6.49	0.14	
18	18	AAAAAAAAABAAA...	27/10/2001	[null]	Twin, particular aspects will a...	0.87	0.48	
19	19	AAAAAAAADBAAA...	27/10/1997	[null]	Political parents know right; p...	10.61	4.77	

3.4 Hypothesis:

The data was obtained in data format. Which we converted into csv files for easy import into the database management system. The data was then loaded into the database management system using INSERT queries. After that the database were connected to python programming language via a connection string. Materialized and non-materialized views were programmatically generated into the database.

Chapter 4

Design

Since our project is researched based. We do not have a specific GUI for it.

4.1 Research Architecture:

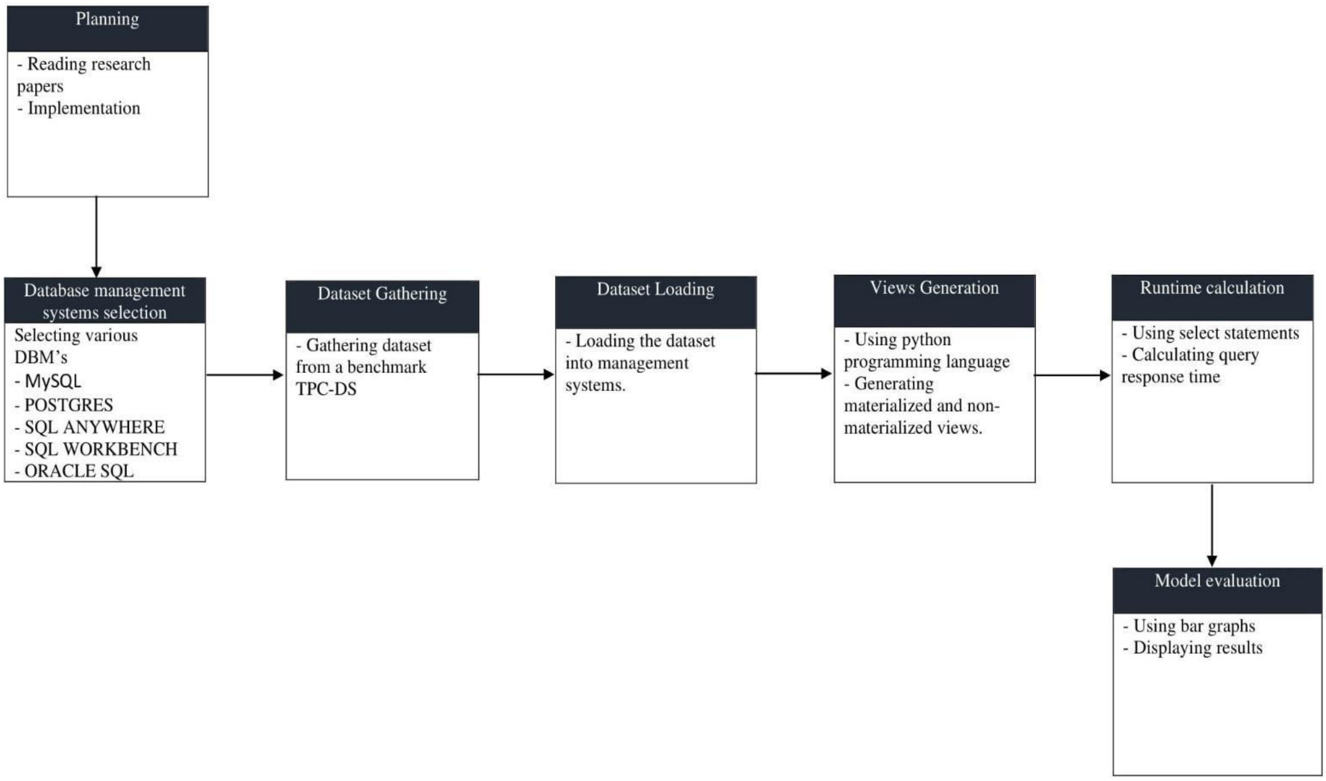


Figure 1.2: Research Architecture

Chapter 5

System Implementation

The following chapter explains the process and development study that we applied in our project. It lists the sorts of visuals that we used to represent our data and our experiments performed on that data.

5.1 System Architecture:

Our system architecture consists of three layers. We used relational database to store our data obtained from the data source. We the experimented on that data using management studios and python programming language. We represented our results in visuals in the form of tables and graphs.

5.1.1 Three-Tier Architecture:

Figure 1.3: Three-Tier Architecture

5.1.2 Language:

The languages we used during our projects are **SQL** and **Python**.

5.1.3 Libraries:

We used the following python libraries for our project

- Psycopg2
- Pyodbc
- cx_Oracle
- mysql. Connector

5.2 Experiments Performed:

We performed our experiments on different database management systems testing their query response time under the same environment. The query response time was then calculated individually of each database management system and compared with each other. The results were then converted into a visual form to get a better understanding.

5.2.1 Data Collection:

We used TPC-DS a decision support benchmark for our dataset. The tools for generating the dataset were downloaded from https://github.com/databricks/tpcds-kit.

The dataset was generated from these tools using command **./dsdgen -scale 1** scale being the size of the data. We used SF=1 to generate 1 GB of data. The data was generated in .dat format. Which was then converted into csv for easy import into the databases by using Microsoft Excel.

The data was the loaded into different database management systems using their respective data importing commands.

5.2.2 Database Management Systems:

We are using variety of database management systems which include

- Microsoft SQL Server Management Studio.
- PostgreSQL.
- SQL Anywhere.
- SQL Workbench.
- Oracle SQL Developer

We had to carefully choose those database management systems which support the generation of materialized views. As our whole work revolves around materialized and non-materialized views.

5.2.3 Experiments Performed on Data:

We performed our experiments on this data with the help of python programming language. Python was connected to these database management systems via a connection string.

Our experiments consisted of generating materialized and non-materialized views. Which were then called in their respective queries. The materialized and non-materialized were generated programmatically into the database.

Every database management system has their own way of generating a materialized view. Following is an example of how a materialized view can be generated in Microsoft SQL Server and PostgreSQL.

Microsoft SQL Server:

```
Create View Myview
WITH SchemaBinding
AS SELECT COL1, SUM(COL2)
From <table-name>
GROUP BY COL1
```

PostgreSQL:

```
Create MATERIALIZED VIEW MyView
AS SELECT COL1,COL2
FROM <table-name>
GROUP BY COL2
```

These views were then called into various SQL statements. These SQL statements were specifically generated for these views.

5.2.4 Calculating Query Response Time:

The query response time for different database management systems was calculated in python using inbuilt libraries. The time was calculated for the SQL statements which were calling a view. Both materialized and non-materialized views were tested with those SQL statements and the response time was noted down. These response times were then converted into tabular and graphical representation.

5.3 Tables:

View Name	View Type	Time in milliseconds
ccv	Materialized view	19
	Non-Materialized view	30
itemv	Materialized view	17
	Non-Materialized view	29
promv	Materialized view	22
	Non-Materialized view	27
storv	Materialized view	22
	Non-Materialized view	34
wrhsv	Materialized view	13
	Non-Materialized view	18
websv	Materialized view	16
	Non-Materialized view	31
webv	Materialized view	22
	Non-Materialized view	28
crv	Materialized view	42
	Non-Materialized view	165
csv	Materialized view	210
	Non-Materialized view	431
srv	Materialized view	58
	Non-Materialized view	175
ssv	Materialized view	210
	Non-Materialized view	633
wrv	Materialized view	34
	Non-Materialized view	191
wsv	Materialized view	110
	Non-Materialized view	370

Table 1.2: Postgres Materialized vs Non-Materialized views

View Name	View Type	Time in milliseconds
ccv	Materialized view	120
	Non-Materialized view	166
itemv	Materialized view	156
	Non-Materialized view	42
promv	Materialized view	370
	Non-Materialized view	468
storv	Materialized view	139
	Non-Materialized view	190
wrhsv	Materialized view	155
	Non-Materialized view	152
websv	Materialized view	312
	Non-Materialized view	596
webv	Materialized view	156
	Non-Materialized view	130
crv	Materialized view	172
	Non-Materialized view	245
csv	Materialized view	460
	Non-Materialized view	932
srv	Materialized view	1317
	Non-Materialized view	3714
ssv	Materialized view	1098
	Non-Materialized view	2999
wrv	Materialized view	5441
	Non-Materialized view	1700
wsv	Materialized view	4309
	Non-Materialized view	2969

Table 1.3: Microsoft SQL Server Materialized vs Non-Materialized views

View Name	View Type	Time in milliseconds
ccv	Materialized view	70
	Non-Materialized view	556
itemv	Materialized view	46
	Non-Materialized view	73
promv	Materialized view	70
	Non-Materialized view	38
storv	Materialized view	38
	Non-Materialized view	75
wrhsv	Materialized view	37
	Non-Materialized view	39
websv	Materialized view	38
	Non-Materialized view	42
webv	Materialized view	30
	Non-Materialized view	34
crv	Materialized view	61
	Non-Materialized view	105
csv	Materialized view	36
	Non-Materialized view	40
srv	Materialized view	40
	Non-Materialized view	87
ssv	Materialized view	95
	Non-Materialized view	105
wrv	Materialized view	70
	Non-Materialized view	91
wsv	Materialized view	110
	Non-Materialized view	150

Table 1.4: SQL Anywhere Materialized vs Non-Materialized views

View Name	View Type	Time in milliseconds
ccv	Materialized view	20
	Non-Materialized view	35
itemv	Materialized view	22
	Non-Materialized view	29
promv	Materialized view	20
	Non-Materialized view	29
storv	Materialized view	25
	Non-Materialized view	39
wrhsv	Materialized view	17
	Non-Materialized view	23
websv	Materialized view	22
	Non-Materialized view	28
webv	Materialized view	31
	Non-Materialized view	39
crv	Materialized view	30
	Non-Materialized view	45
csv	Materialized view	38
	Non-Materialized view	95
srv	Materialized view	58
	Non-Materialized view	140
ssv	Materialized view	190
	Non-Materialized view	473
wrv	Materialized view	28
	Non-Materialized view	139
wsv	Materialized view	100
	Non-Materialized view	247

Table 1.5: Oracle SQL Materialized vs Non-Materialized views

5.4 Graphs:

Following is the graphical representation of the results we achieved during our experiments.

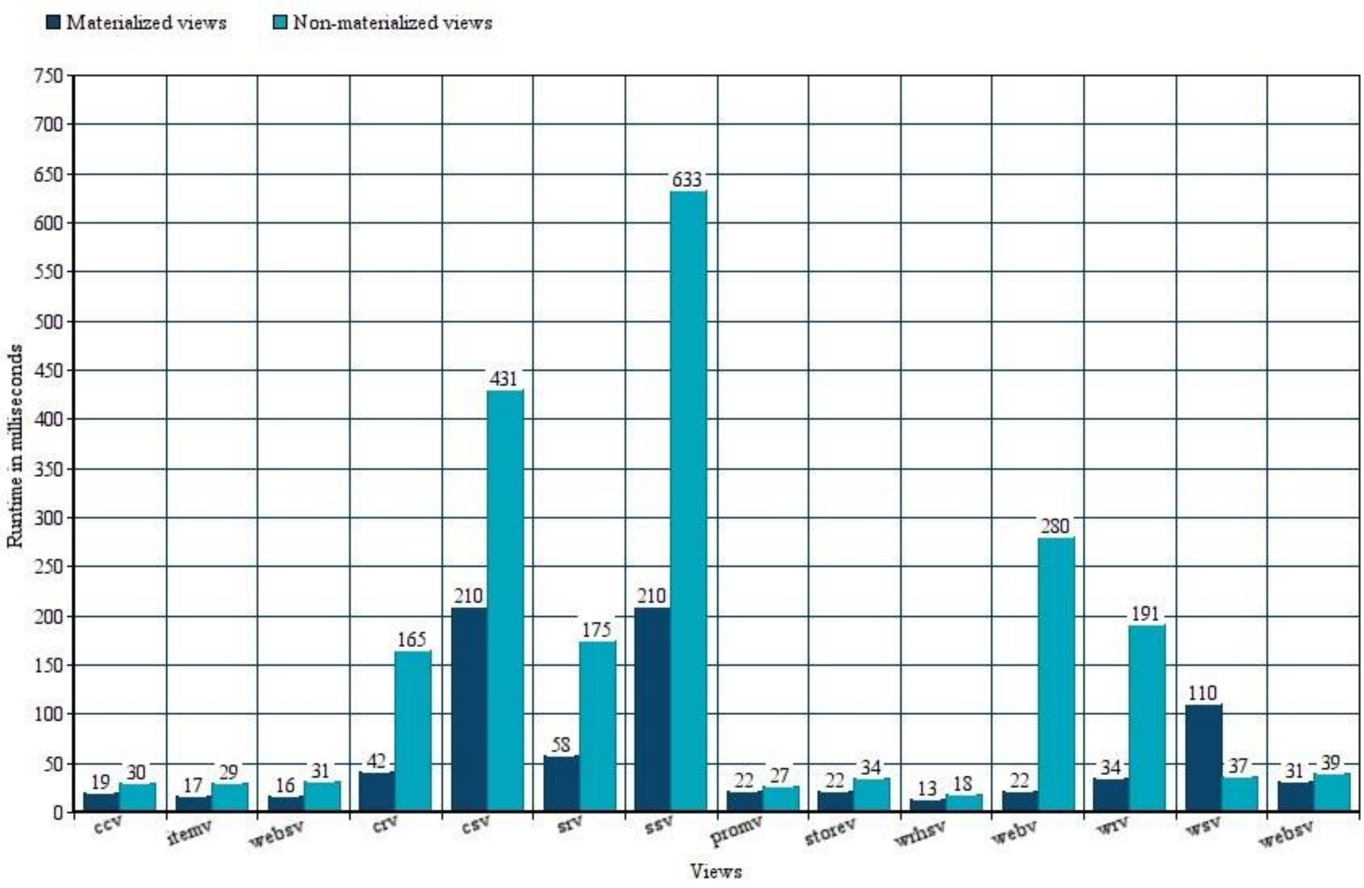


FIGURE 1.4: Query Response Time for POSTGRES Materialized and Non-Materialized Views

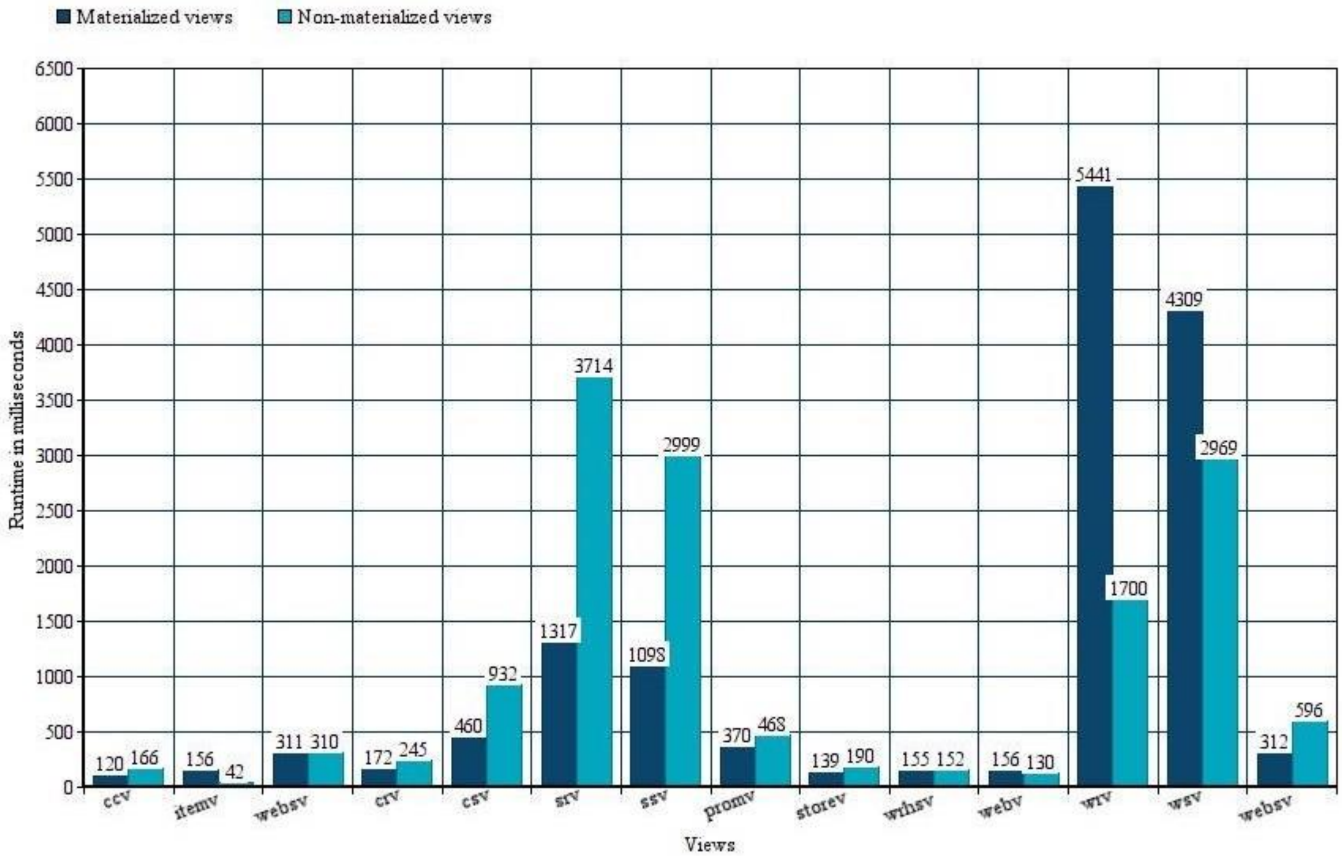


FIGURE 1.5: Query Response Time for SQL SERVER MANAGEMENT STUDIO Materialized and Non-Materialized Views

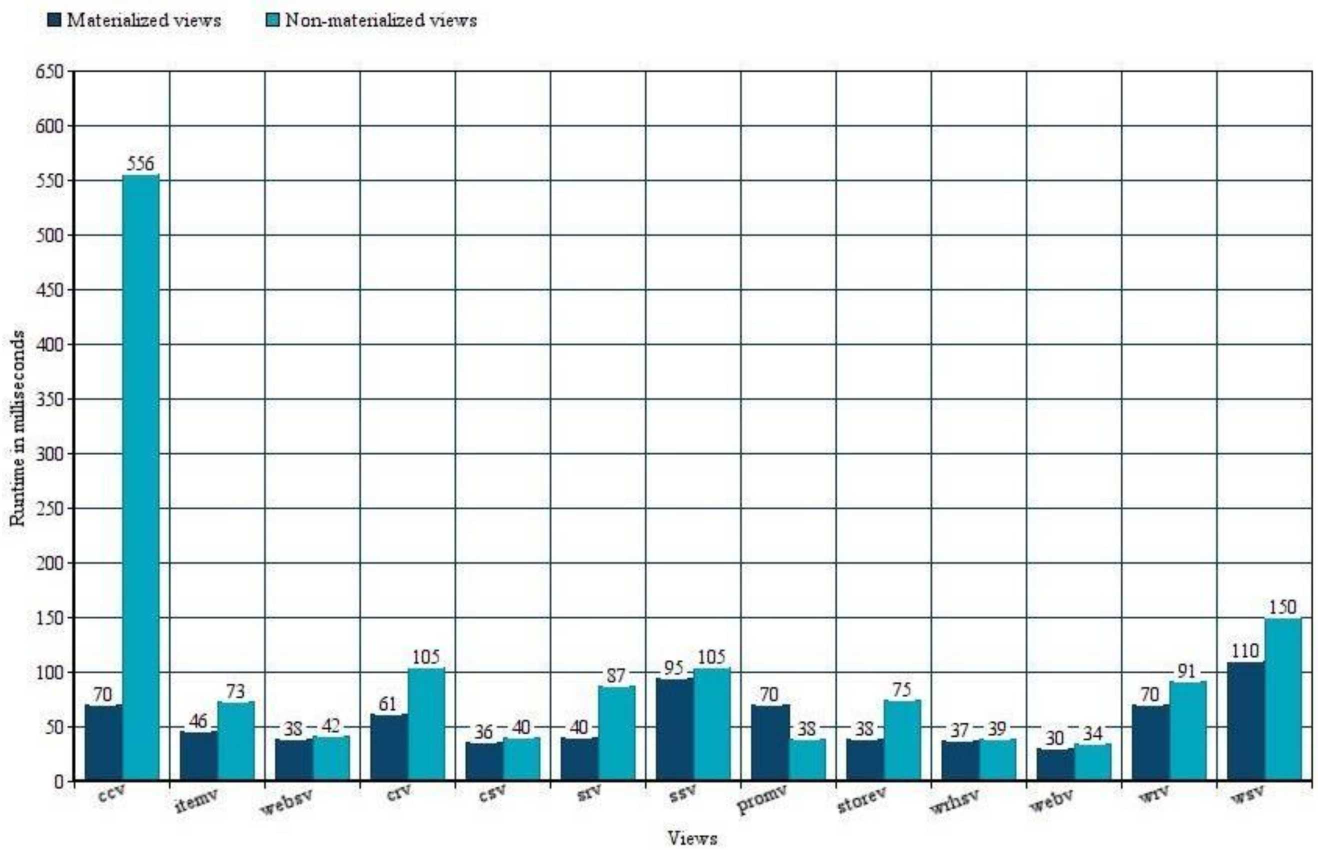


FIGURE 1.6: Query Response Time for SQL ANYWHERE Materialized and Non-Materialized Views

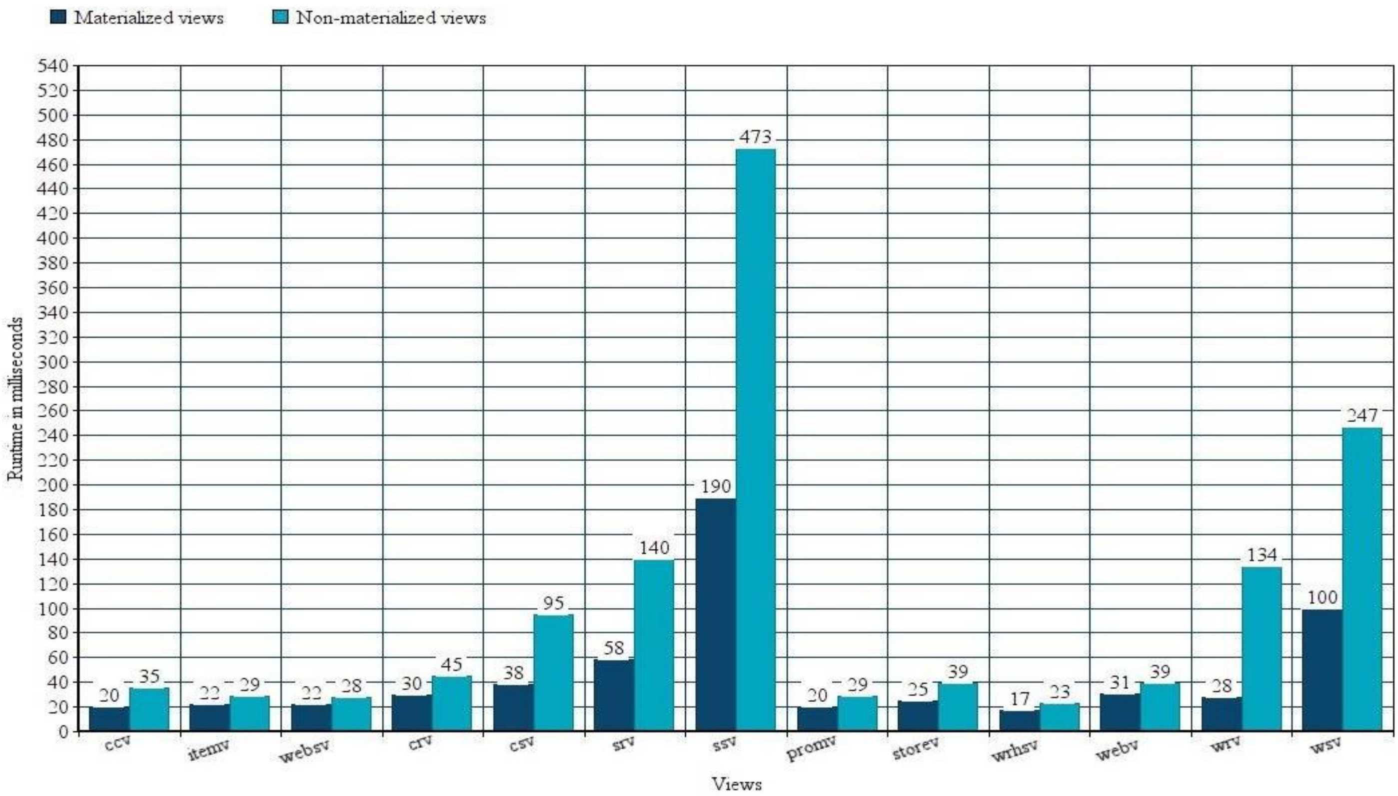


FIGURE 1.7: Query Response Time for ORACLE SQL Materialized and Non-Materialized Views.

5.5 Result Analysis:

According to our research and experiment results we can clearly see that materialized views are far better than non-materialized views. However, in some cases the non-materialized view response time is better than the materialized view.

That is why the database administrator must know which views to materialize and which views to not to. Since materializing all views will take up very much space therefore defeating the purpose of creating them.

We have provided our results of different database management systems. These results vary due to the difference in response times of these management systems. As we can clearly see that some database management systems are way ahead in their response time from others.

Although it also depends on the system on which these management systems are ran. We ran these experiments on a single system providing a fair environment for each database management system.

Chapter 6

Conclusions

This was a brief report detailing our project “Materialized vs Non-Materialized views in contemporary databases”. In which we detailed that we aim to achieve i.e Generation of materialized views. Generation of non-materialized views. Comparing the response time of these views programmatically. We have performed our experiment on different database management systems and compared their response times.

We gathered different database management systems which use “STRUCTURED QUERY LANGUAGE” (SQL) to perform our experiments. We connected our databases to python programming language via a connection string. After connecting with the database management systems, we started our experiments.

We generated materialized and non-materialized views programmatically through python. After generating these views, we ran multiple different queries on these views. We calculated the query response time using python’s inbuilt functions.

We calculated accurate results of every query which used materialized and non-materialized views. The response time was calculated in milliseconds. These results were displayed in the form of tables and graphs. Since we calculated our results on the same system to provide a fair environment for our database management systems, we can say that the accuracy of our results is satisfying.

We achieved our goal of providing results between the response time of materialized and non-materialized views. We can say that which database management system was the best by looking at our results. But since the database management systems depend on the environment, they are used in we are more focused on the response times of materialized and non-materialized views. We cannot say which database management system is the best based on our single system environment.

Since in our experiments the views were generated and selected by the database administrator. For our future work we can focus on generating and selecting these views automatically using artificially using either machine learning or artificial intelligence. As the world is converting its data into digital form the data is going to keep on increasing. We need to focus on how we are going to maintain that data. Since data is growing larger by the day. It will be very optimal if we can automate the maintenance of this data using modern technology.

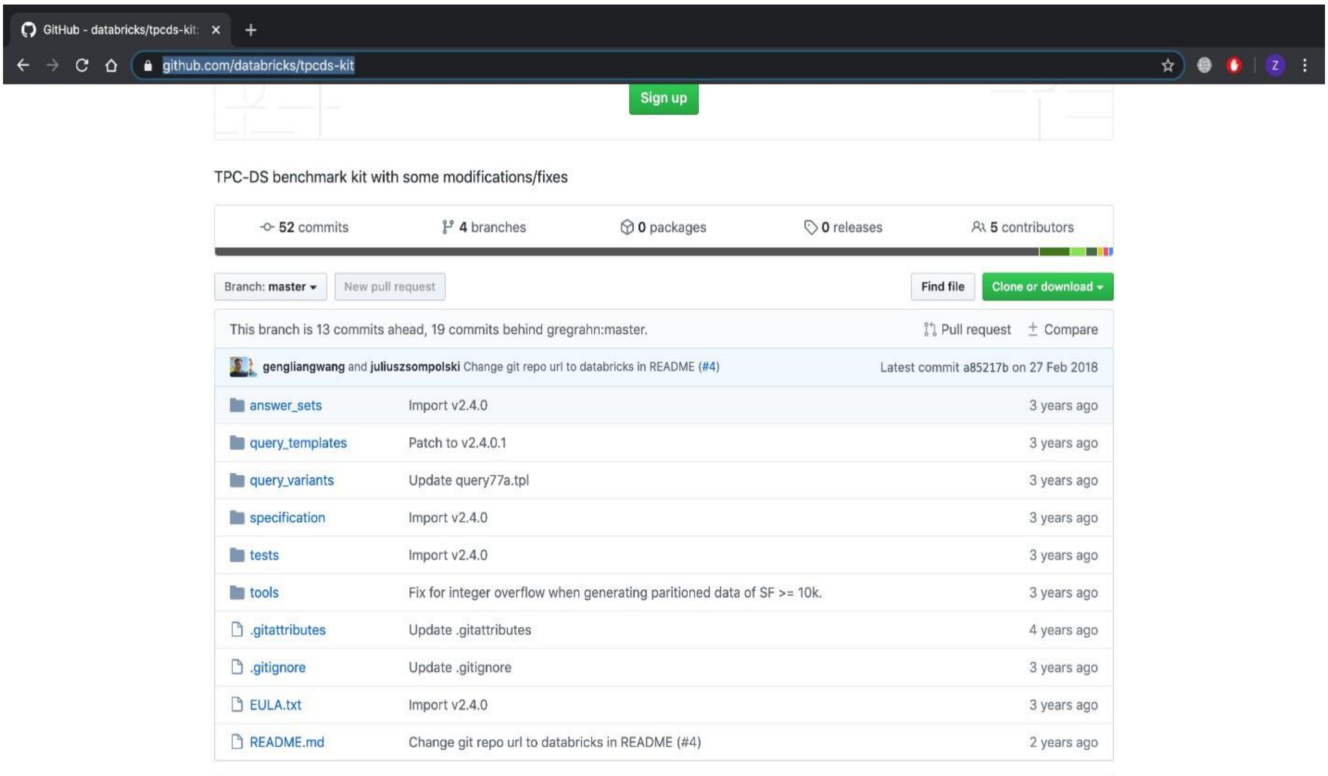
Appendix A

User Manual

Following is a step by step guide on how to perform the above-mentioned experiments.

Step 1:

Download the tpc-ds benchmark from https://github.com/databricks/tpcds-kit



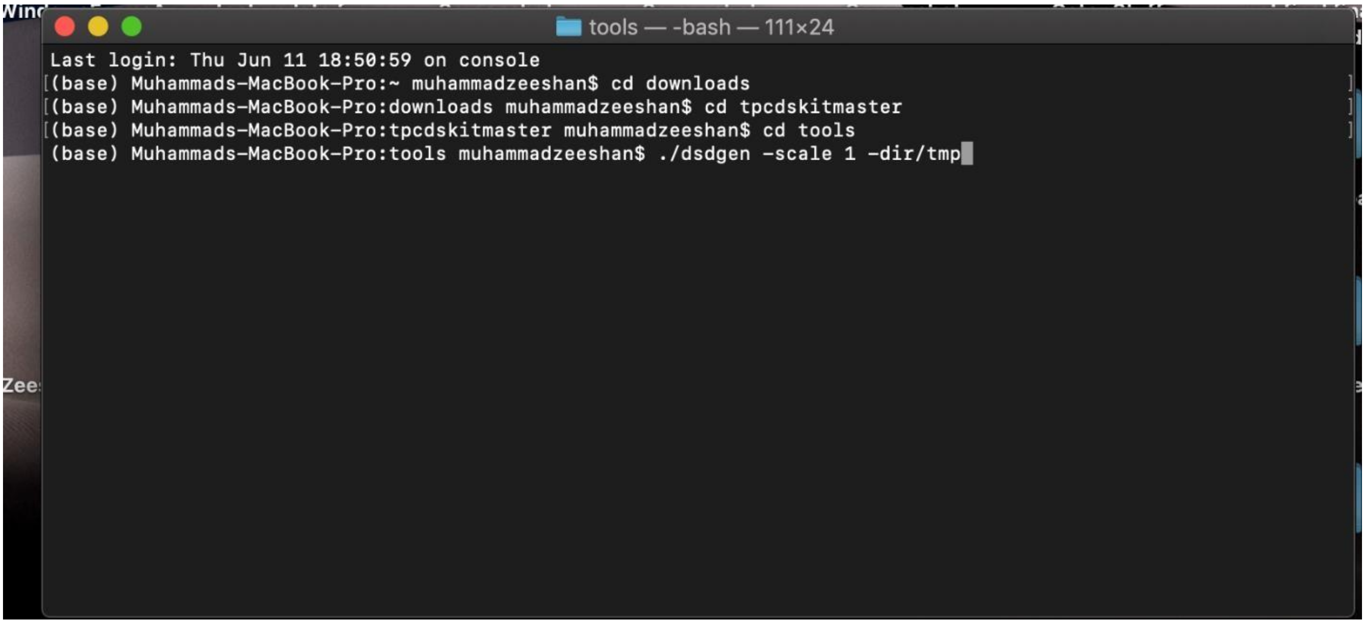
Step 2:

After downloading and extracting the benchmark open the tools folder in the tpc-ds kit folder and run an executable file called MAKEFILE.

join.c	29-Apr-2020 at 6:52 AM	13 KB	C Source
keywords.c	29-Apr-2020 at 6:52 AM	5 KB	C Source
keywords.h	29-Apr-2020 at 6:52 AM	2 KB	C Head...Source
list.c	29-Apr-2020 at 6:52 AM	6 KB	C Source
list.h	29-Apr-2020 at 6:52 AM	3 KB	C Head...Source
load.c	29-Apr-2020 at 6:52 AM	3 KB	C Source
load.h	29-Apr-2020 at 6:52 AM	2 KB	C Head...Source
makefile	29-Apr-2020 at 6:52 AM	37 KB	Unix executable
Makefile.osx	16-May-2020 at 11:55 PM	37 KB	Document
mathops.h	29-Apr-2020 at 6:52 AM	2 KB	C Head...Source
misc.c	29-Apr-2020 at 6:52 AM	4 KB	C Source
misc.h	29-Apr-2020 at 6:52 AM	2 KB	C Head...Source
mkheader.c	29-Apr-2020 at 6:52 AM	6 KB	C Source
mkheader.vcproj	29-Apr-2020 at 6:52 AM	5 KB	Document
names.dst	29-Apr-2020 at 6:52 AM	233 KB	Document
nulls.c	29-Apr-2020 at 6:52 AM	3 KB	C Source

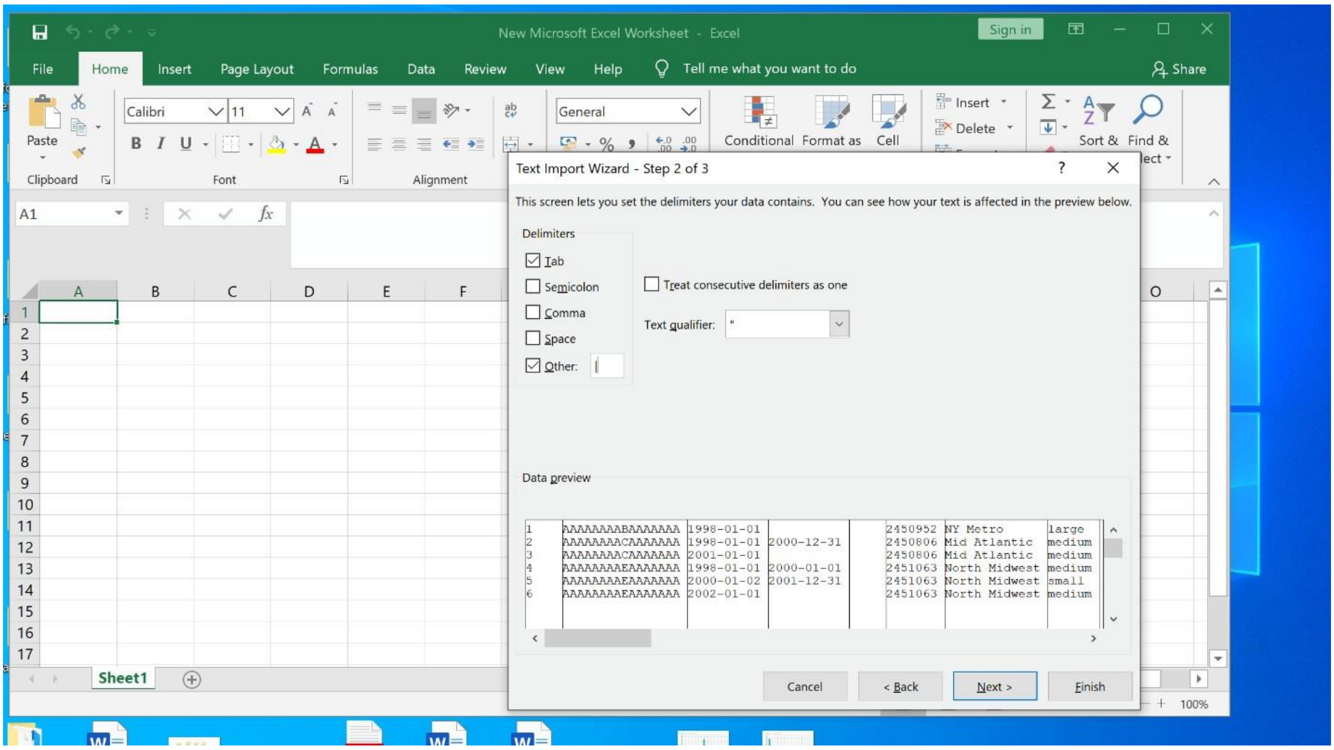
Step 3:

After running the make file. Open up terminal (or command prompt if using windows) go into the tpcds tools directory. In the tools directory run command “ ./dsdgen – scale 100 -dir/tmp” Scale defines the amount of data you need to generate. The official scale factors are 1TB, TB, 10 TB etc. We generated a total of 1gb data so we used SF= 1. -dir/tmp defines which directory to store the generated data in.



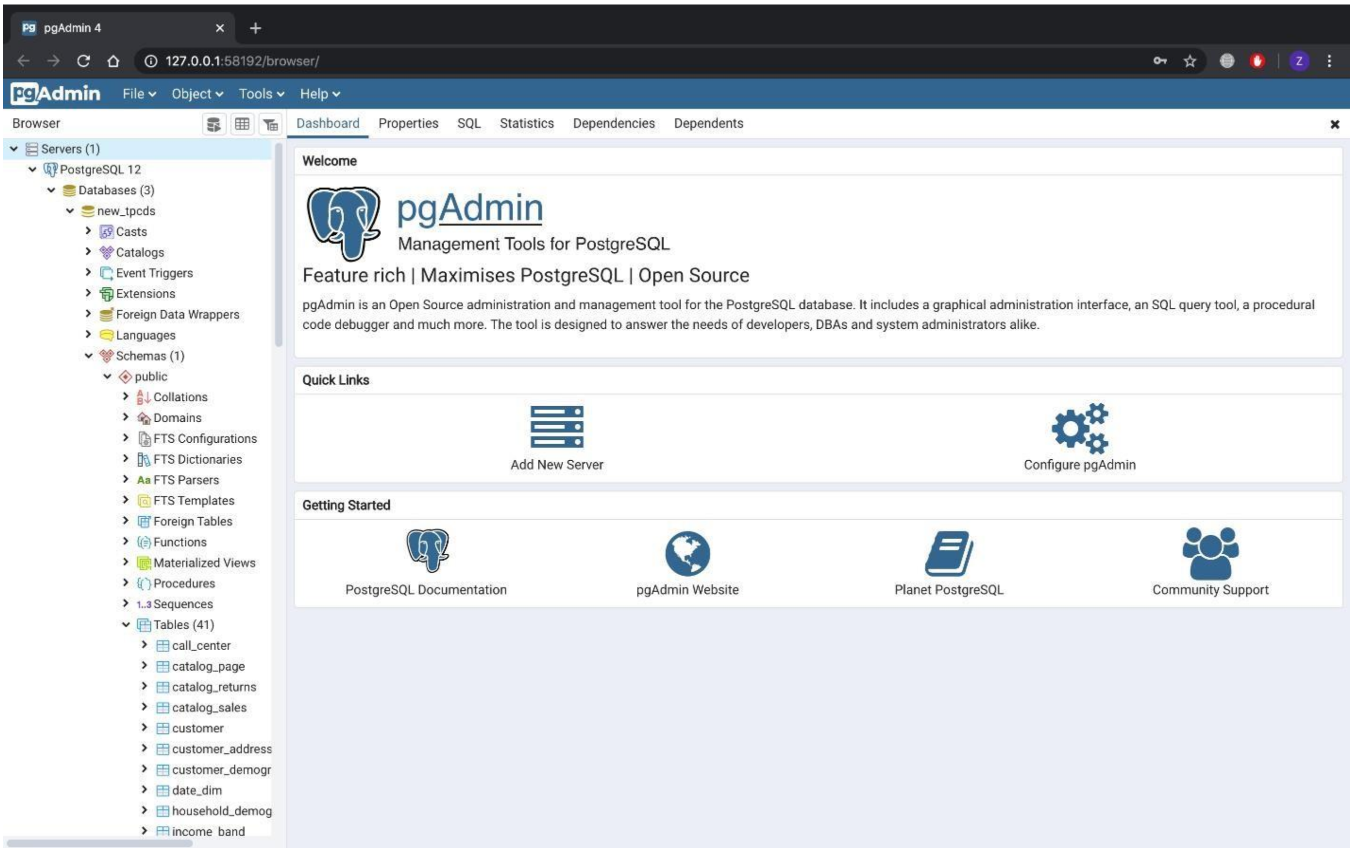
Step 4:

The files generated from the dsdgen are in .dat format which are then converted into csv files using Microsoft Excel. The columns are delimited by "|"



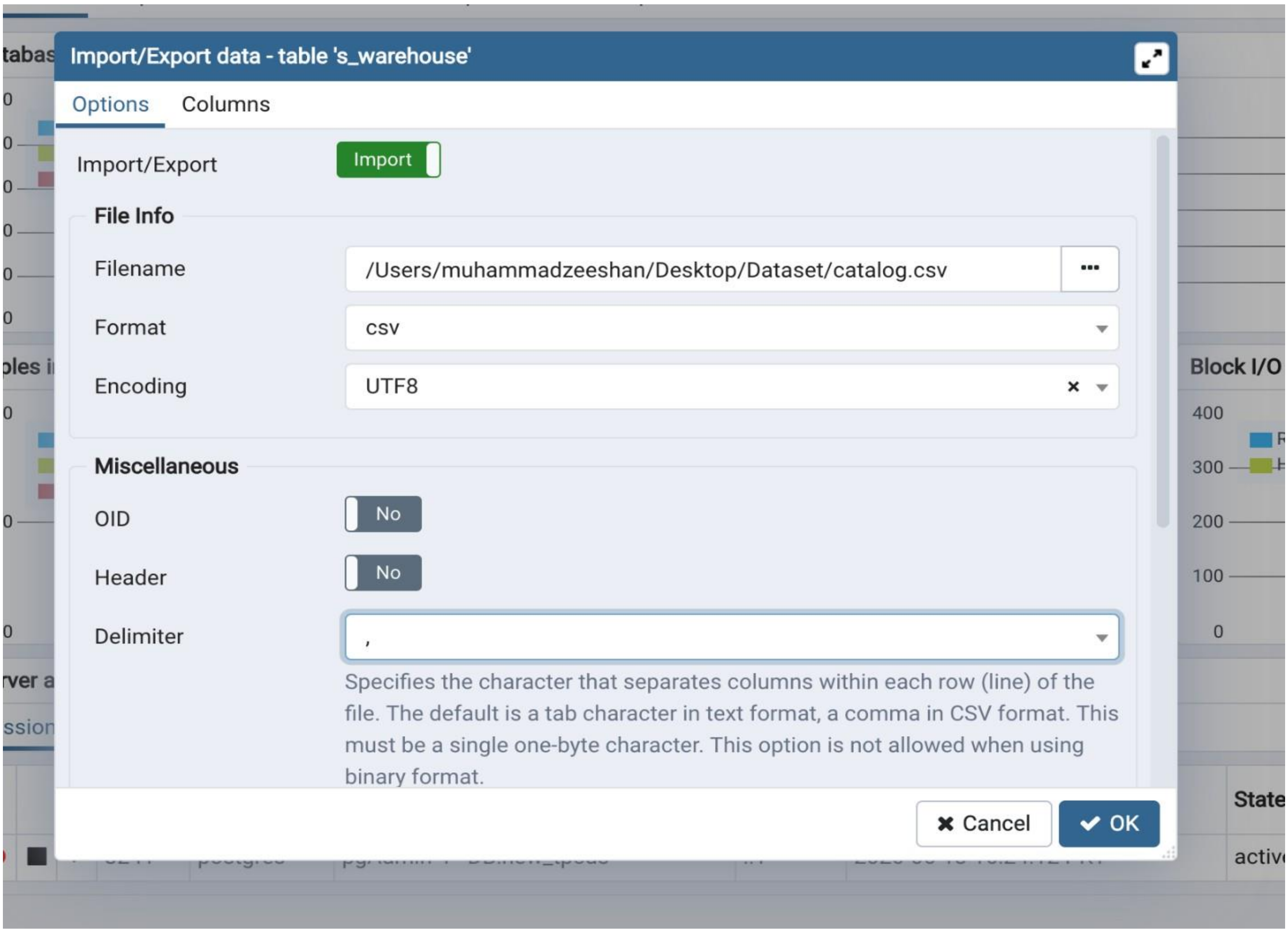
Step 5:

After you have generated the dataset. Download any database management studio which support View Materialization. For this manual purpose I will be using POSTGRES.



Step 6:

Create a database in the database management system and create tales. The create table SQL queries are provided in the tpcds-tools folder named “tpcds source.sql”. After creating the tables import the data into those tables.



Step 7:

Connect the database management system with python programming language using appropriate connection string. For every database management system the connection string varies.

```
Welcome    testingpostgres.py ×
Users > muhammadzeeshan > Desktop > FYP > testingpostgres.py > ...
import psycopg2
from time import time
import datetime
connection = psycopg2.connect(user = "postgres",
                              password = "zeeshan",
                              host = "127.0.0.1",
                              port = "5432",
                              database = "new_tpcds")
```

Step 8:

After connecting with python generate materialized and non-materialized views into the database management system. The view queries can be found in tpc-ds documentation found on www.tpc.org

```
#Creating materialized and non materialized view
mv_storv= """create materialized view storv as
select store_seq.nextVal s_store_sk…
from  s_store_m left outer join date_dim d1 on TO_DATE(stor_closed_date,'YYYY-MM-DD')= d1.d_date
        ,store
where  stor_store_id = s_store_id
   and s_rec_end_date is null;"""

nv_storv = """create table storv as
select store_seq.nextVal s_store_sk…
from  s_store_m left outer join date_dim d1 on TO_DATE(stor_closed_date,'YYYY-MM-DD')= d1.d_date
        ,store
where  stor_store_id = s_store_id
   and s_rec_end_date is null;"""
   cursor.execute(""+mv_storv+"")
   cursor.execute(""+nv_storv+"")
```

Step 9:

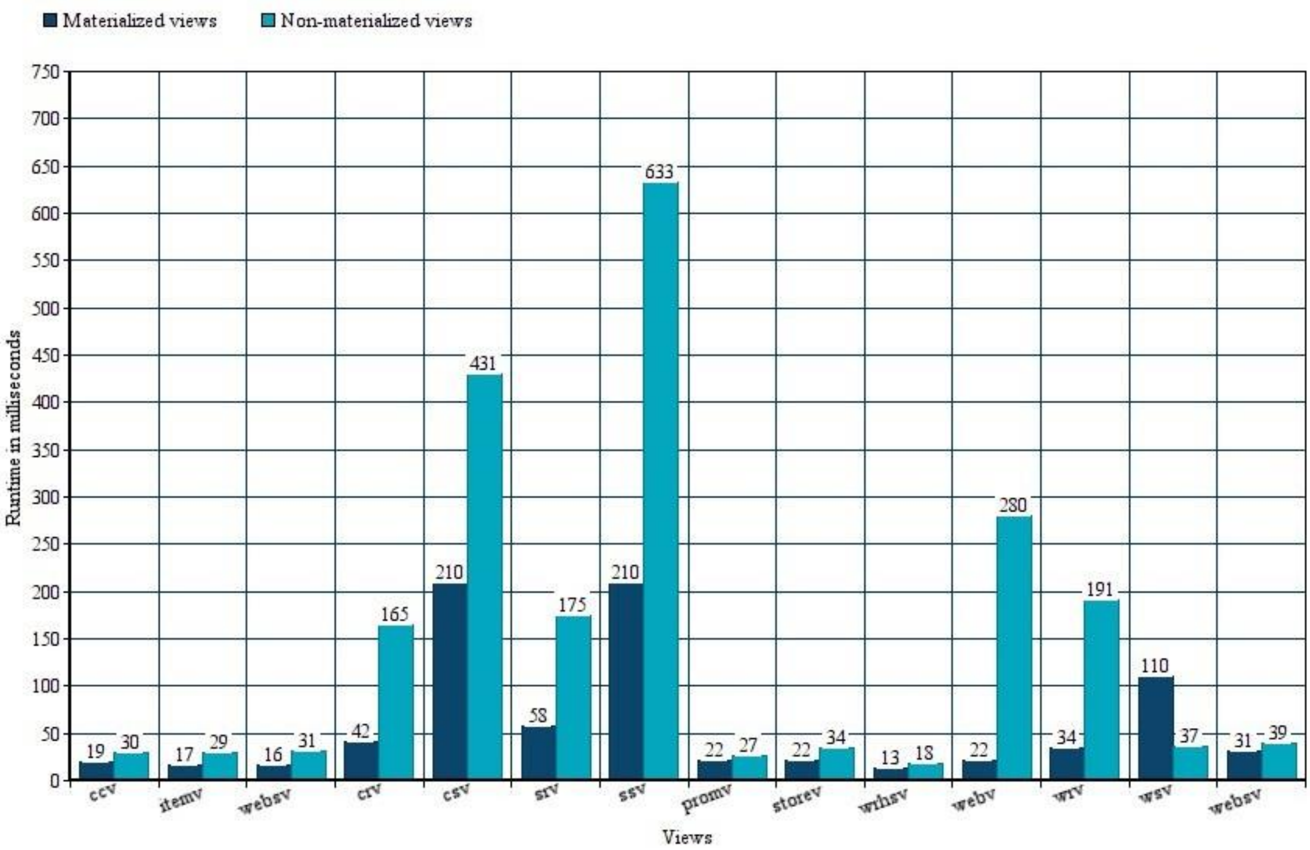
After generating views. Execute the select queries which use these views in them. Those queries can be found in tpcds/tests folder with “sql” extension. Calculate the execution time of these queries using in built functions of Python.

```
cursor = connection.cursor()
a = datetime.datetime.now()
#tic = time()
cursor.execute("insert into store select * from storv;")
#toc = time()

b = datetime.datetime.now()
delta = b-a
print(delta)
#timeinmili = toc-tic
print (int(delta.total_seconds()*10000))
```

Step 10:

Display the time calculated in the form of tables and graphs.



6 References

[1] Jonathan Goldstein & Per-Ake Larson. Optimizing Queries Using Materialized Views: A Practical Scalable Solution,2001. Microsoft Research, One Microsoft Way, Redmond, WA 98052.

[2] Surajid Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos & Kyuseok Shim. Optimizing Queries With Materialized Views, 2001. Microsoft Research, Redmond, WA. Hewlett-Packard Laboratories, Palo Alto, CA. Momferatory 71, Athens, Greece. IBM Almdaen Research Center, San Jose, CA.

[3] Rada Chirkova, Jun Yang. Materialized Views, November 2012. Now Publishers Inc, P.O. Box 1024, Hanover, MA, United States.