SQL Optimization Project

In this project you'll set up tables in Postgres, import a dataset of computer science papers, write queries to answer a series of questions about the dataset, examine the performance of these queries with and without indexing, then write a report of your findings.

Part One: Project Set Up

1. Postgres Installation

Before starting work on your computer, you'll need to install PostgresSQL. If you do not already have Postgres set up on your machine, please follow this tutorial from DataCamp to install Postgres.

https://www.datacamp.com/tutorial/installing-postgresql-windows-macosx

2. Table creation

Now that you have Postgres running, you'll need to create the following tables in your database. Run these SQL statements to create the Articles, Authors, Books, Inproceedings, and Processing tables.

2.1 Articles Table

```
CREATE TABLE public.articles
(
"articleID" UUID NOT NULL DEFAULT gen_random_uuid (),
title text,
author text,
year text,
journal text,
pages text,
CONSTRAINT articles_pkey PRIMARY KEY ("articleID")
)
```

2.2 Authors Table

```
CREATE TABLE public.authors
(
"authorID" UUID NOT NULL DEFAULT gen_random_uuid (),
"authorName" text COLLATE pg_catalog."default",
CONSTRAINT authors_pkey PRIMARY KEY ("authorID")
)
```

2.3 Books Table

```
CREATE TABLE public.books
(
"bookID" UUID NOT NULL DEFAULT gen_random_uuid (),
title text  COLLATE pg_catalog."default",
author text COLLATE pg_catalog."default",
publisher text COLLATE pg_catalog."default",
isbn text,
year text,
pages text,
CONSTRAINT books_pkey PRIMARY KEY ("bookID")
)
```

2.4 Inproceedings Table

```
CREATE TABLE public.inproceedings
```

```
(
"inproceedingsID" UUID NOT NULL DEFAULT gen_random_uuid (),
title text COLLATE pg_catalog."default",
author text COLLATE pg_catalog."default",
year text,
pages text,
booktitle text COLLATE pg_catalog."default",
CONSTRAINT inproceedings_pkey PRIMARY KEY ("inproceedingsID")
)
```

2.5 Proceedings Table

```
CREATE TABLE public.proceedings
(
"proceedingsID" UUID NOT NULL DEFAULT gen_random_uuid (),
title text,
editor text,
year text,
booktitle text,
series text,
publisher text,
CONSTRAINT proceedings_pkey PRIMARY KEY ("proceedingsID")
)
```

2.6 Publications Table

```
CREATE TABLE public.publications
(
"pubID" UUID NOT NULL DEFAULT gen_random_uuid (),
title text,
year text,
pages text,
CONSTRAINT publications_pkey PRIMARY KEY ("pubID")
)
```

3. Data Acquisition

Now that you have the Tables set up, you need to download the data and import it into Postgres. Follow the steps below to download the data, extract it into a CSV, and load it into the Postgres tables you just created.

3.1 Set Client encoding

Execute the following in the PSQL shell `SET CLIENT_ENCODING TO 'utf8';`

3.2 Download, extract & parse the data

Execute this Python script - this downloads, extracts and parses the dataset into CSVs (which are stored into the `dataset` folder).

3.3 Load data to Postgres

Use Pgadmin's import function to copy and insert values from the .csv files into the
Tables you created in the last steps. Follow along with this tutorial if you have not used the import function before.

4. Write queries to answer the following questions

Now that your database is set up, you'll do the bulk of the work in this project. You'll need to write SQL Queries to answer the following questions. Once you finish, please produce a report comparing query performance with and without proper indices - detailed more in Step 5.

HINT: we use the EXPLAIN function of PostgreSQL to view the query plan that was generated for the queries and how index was used in each plan as compared to the queries without index

BONUS: Study the effect of cache in query performance

4.1 - Highly Literate July Conferences.

Write a SQL Query to find all the **conferences** held in **2018** that have published at least 200 papers in a single decade.

Please note, conferences may be annual conferences, such as KDD. Each year a different number of conferences are held. You should list conferences multiple times if they appear in multiple years.

4.2 - Accomplished Authors

Write a SQL Query to find all the **authors** who published **at least 10 PVLDB** papers and at least **10 SIGMOD** papers. You may need to do some legwork here to see how the DBLP spells the names of various conferences and journals.

4.3 - Conference Publications by Decade

Write a SQL Query to find the total number of **conference publications** for each decade, starting from 1970 and ending in 2019. For instance, to find the total papers from the 1970s you would sum the totals from 1970, 1971,1972…1978, up to 1979. Please do this for the decades 1970, 1980, 1990, 2000, and 2010.

Hint: You may want to create a temporary table with all the distinct years.

4.4 Highly Published Data Authors

Write a SQL Query to find the **top 10 authors** publishing in **journals and conferences whose titles contain the word data.** These will likely be some of the people at the cutting edge of data science and data analytics. As a fun exercise, find a paper one of them wrote that interests you and read it!

4.5 Highly Published June Conferences

Write a SQL query to find the names of **all conferences**, happening in June, where the **proceedings** contain **more than 100 publications**. Proceedings and inproceedings are classified under conferences - according to the dblp website, so make sure you use both tables and use the proper attribute **year**.

5. Write the Report

Write a report to answer the following questions for each problem.

1) How do you improve query performance from your initial query?
2) Where did you create new indexes?
3) What was the impact of new index creation in terms of query cost and performance?

4) BONUS: What was the effect of cache on each query's performance?