****

**CE/CZ 4031:**

**Database System Principles**

***Project 1: Querying Databases Efficiently***

|  |  |
| --- | --- |
| THA TOE OO @JODI | U1621448G |
| LIM ZHI EN | U1622411D |
| ANDY NG CHIN KUAN | U1621910C |
| MUHAMMAD AMIRUL AFIQ | U1602684A |

**Contents**

[Introduction 3](#_Toc21034834)

[Schema Design and Data Acquisition 4](#_Toc21034835)

[Data Acquisition 5](#_Toc21034836)

[Queries and Optimizing Queries 6](#_Toc21034837)

[Effect of database size on the query time 6](#_Toc21034838)

[Building an Index and Studying the Effect of the Index 7](#_Toc21034839)

[Advanced Part: Study the Effect of Cache 8](#_Toc21034840)

[Appendix A : Table Schema 9](#_Toc21034841)

[Appendix B : Queries and Results 10](#_Toc21034842)

[Query 1 10](#_Toc21034843)

[Query 2 10](#_Toc21034844)

[Query 3 10](#_Toc21034845)

[Query 4 11](#_Toc21034846)

[Query 5 11](#_Toc21034847)

[Query 6 12](#_Toc21034848)

[Query 7a 12](#_Toc21034849)

[Query 7b 12](#_Toc21034850)

[Query 8 13](#_Toc21034851)

# Introduction

In this assignment, our team have decided to use PostgreSQL as the database of our choice. The database for this project was created on a Windows laptop as shown below.

TODO: Insert Laptop config image

# Schema Design and Data Acquisition

During our discussion, we have decided to design our schema based on the recommended sets and relationship given by the lab manual. Three tables are created as shown:

1. Author
2. Authored
3. Publications

The ER Diagram for the schema is shown below.

TODO : on 4/10 ER Diagram

The commands used for creating the tables can be found in [Appendix A](#_Annex_A).

# Data Acquisition

In this section of the project, our team have used JAVA for the SAX Parser and to write the data into a CSV file. We have decided to follow the NULL approach to deal with the isa relationship in our relation schema.

During our design of the SAX Parser, we encountered various special cases that we had to deal with. One of the issues that we encountered was the selection of the right “CSV Separator” to distinguish the different columns while writing into a CSV file. In our experiment, we have found that common symbols like , or | is used within the file in the text and will disrupt the column segregation. As a result, we have decided to use the backtick symbol or ` as it is a unique symbol not found within the dblp file.

Another issue faced with the use of the backslash symbol and null values within the text of some titles. We have created a simple “hack” to deal with these cases and can be found in our "escapeSpecialCharacters” function.

Another special case that we found was some title uses tags such as "tt", "sub", "i", "sup" to represent inline teletype text, subscript, italics and superscript. In our experiment we realized that the titles will get shortened and ignored by the parser. To avoid such cases from happening, we detect those tags in the title and append any text found within them to the title string.

TODO : Not sure what he means by : “You need to submit: 1.3) all the codes for this step (as an appendix to your report).” Does he want the parser code ? so long sia, the appendix will be very long lol.

# Queries and Optimizing Queries

Code snippets, screen captures and analysis of the results for all the queries and can be found in [Appendix B](#_Appendix_B) of the report.

For our last query, we decided to find all the authors with the second highest number of publications.

TODO : Do we need to optimize our queries ?

## Effect of database size on the query time

| **Queries** | **before** | **after 1/2** | **after 1/4** |
| --- | --- | --- | --- |
| 1 | 6.763s |  |  |
| 2 | 12.764s |  |  |
| 3 | 2.656s |  |  |
| 4 | 66s |  |  |
| 5 | 17.955s |  |  |
| 6 | 2.527s |  |  |
| 7a | 32.664s |  |  |
| 7b | 19.295s |  |  |
| 8 | 178s |  |  |

TODO: 1. Complete the ½ and ¼ of the database

2. Insert figure of database size on query time

# Building an Index and Studying the Effect of the Index

During our experimentation with indexes, through the Explain Analyze keywords in PostgreSQL we found out that even with the creation of indexes, some of the queries were not making use of the indexes. This is because PostgreSQL decides to perform a sequential scan on any query that will hit a significant portion of a table. This is because, it is much faster to do so than to request for the index and then using the index for the query. To alleviate this, we decided to turn off sequential scan using the following command:

|  |
| --- |
| set enable\_seqscan = off; |

This forces PostgreSQL to use the indexes that we have created and thus shows the effect of index on the queries. On occasions where PostgreSQL uses the indexes naturally, we have kept the sequential scan settings on.

|  |  |  |  |
| --- | --- | --- | --- |
| **Queries** | **before** | **after index** | **seq scan** |
| 1 | 6.763s | 6.577s | off |
| 2 | 12.764s | 11.414s | off |
| 3 | 2.656s | 16.353 | off |
| 4 | 66s | 65s | on |
| 5 | 17.955s | 16.515s | on |
| 6 | 2.527s | 1.812s | off |
| 7a | 32.664s | 147s | off |
| 7b | 19.295s | 15.139s | on |
| 8 | 178s | 175s | off |

TODO: 1. Insert Your CREATE INDEX statements;

2. report of the analysis on whether some of your indexes help some of your queries.

# Advanced Part: Study the Effect of Cache

|  |  |  |  |
| --- | --- | --- | --- |
| **Queries** | **64MB** | **128MB** | **256MB** |
| 1 | 6.649 | 6.763s |  |
| 2 | 12.146 | 12.764s | 12.249 |
| 3 | 2.854 | 2.656s |  |
| 4 | 65 | 66s |  |
| 5 | 18.742 | 17.955s |  |
| 6 | 2.571 | 2.527s |  |
| 7a | 32.834 | 32.664s |  |
| 7b | 19.453 | 19.295s |  |
| 8 | 178 | 178s |  |

TODO: 1. Add explanation and finish the table

2. the analysis on the effect of cache size on the performance of some of your queries.

# Appendix A : Table Schema

|  |
| --- |
| CREATE TABLE Publications( |
| pubKey TEXT primary key, |
| mdate TEXT, |
| title TEXT, |
| category TEXT, |
| journal TEXT, |
| bookTitle TEXT, |
| publishedYear TEXT, |
| publishedMonth TEXT, |
| crossRef TEXT |
| ); |
|  |
| Drop Table IF EXISTS Author; |
|  |
| Create TABLE Author ( |
| name TEXT primary key |
| ); |
|  |
| DROP TABLE IF EXISTS authored; |
| Create TABLE authored ( |
| author\_name TEXT, |
| publication\_key TEXT |
| ); |
|  |
| copy author(name) |
| from '..\author.csv' |
| csv header; |
|  |
| copy authored |
| from '..\authored.csv' |
| delimiter '`'; |
| delete from authored where author\_name = 'author\_name'; |
|  |
| copy publications |
| from '..\publication.csv' |
| delimiter '`'; |
| delete from publications where pubKey = 'pubKey'; |
|  |
| ALTER TABLE authored |
| ADD CONSTRAINT verify\_authored FOREIGN KEY (author\_name) REFERENCES author (name) |
|  |
| ALTER TABLE authored |
| ADD CONSTRAINT verify\_authored FOREIGN KEY (author\_name) REFERENCES author (name) |

# Appendix B : Queries and Results

## Query 1

|  |
| --- |
| select category, count(\*) |
| from publications |
| where publishedyear between '2000' and '2018' |
| group by category |

TODO: Screen captures of the results of these SQL queries

## Query 2

|  |
| --- |
| select distinct booktitle |
| from ( |
| select booktitle, publishedyear, count(\*) as conf\_count |
| from publications |
| where category = 'inproceedings' |
| group by booktitle, publishedyear |
| ) conf |
| where conf\_count > 500 |

TODO: Screen captures of the results of these SQL queries

## Query 3

|  |
| --- |
| SELECT concat(CAST(T.YearDivision \* 10 AS nchar(4)), N' - ', CAST(T.YearDivision \* 10 + 9 AS nchar(4))) AS YearRange, SUM(T.TotalCount) |
| FROM |
| ( |
| SELECT cast (publishedyear as int) / 10 AS YearDivision, COUNT(\*) AS TotalCount |
| FROM publications |
| where publishedyear <> 'null' |
| GROUP BY publishedyear |
| ) T |
| GROUP BY YearDivision |
|  |

TODO: Screen captures of the results of these SQL queries

## Query 4

|  |
| --- |
| create view co\_count as( |
| select T3.author, T3.cnt from ( |
| select author, count(\*) as cnt from ( |
| select A1.author\_name as author, A2.author\_name as co\_author from authored A1 join ( |
| select \* from publications where (category = 'inproceedings' |
| and crossRef in (select pubkey from publications where category = 'proceedings' |
| and lower(title) like '%data%')) |
| or (category = 'article' and lower(journal) like '%data%') |
| )T1 on A1.publication\_key = T1.pubkey join authored A2 on A1.publication\_key = A2.publication\_key |
| and A1.author\_name <> A2.author\_name) T2 group by author) T3); |
|  |
| select author, cnt |
| from co\_count |
| where cnt = (select max(cnt) from co\_count); |

TODO: Screen captures of the results of these SQL queries

## Query 5

|  |
| --- |
| select \* |
| from ( |
| select author\_name, count(\*) as pub\_count |
| from authored join ( |
| select \* from publications where |
| (category = 'inproceedings' |
| and crossRef in (select pubkey from publications where category = 'proceedings' |
| and lower(title) like '%data%')) |
| or (category = 'article' and lower(journal) like '%data%') |
| ) T1 |
| on T1.pubkey = authored.publication\_key |
| group by author\_name |
| )T |
| order by pub\_count DESC |
| LIMIT 10; |
|  |

TODO: Screen captures of the results of these SQL queries

## Query 6

|  |
| --- |
| select title from publications |
| where category = 'proceedings' |
| and lower(title) like '%data%' |
| and pubkey in ( |
| select crossref from( |
| select crossref, count(\*) as cnt from publications where category = 'inproceedings' |
| group by crossref) T1 where T1.cnt > 100); |

TODO: Screen captures of the results of these SQL queries

## Query 7a

|  |
| --- |
| select author.name, authored.author\_name |
| from ( |
| author join authored on author.name = authored.author\_name |
| join publications on authored.publication\_key = publications.pubKey |
| ) |
| where publications.publishedYear Between '1990' and '2019' |
| and substring(author.name, length(author.name)-strpos(reverse(author.name),' ')+2, length(author.name)) like 'H%' |
| group by authored.author\_name, author.name |
| having count(distinct publications.publishedYear) = 30; |

TODO: Screen captures of the results of these SQL queries

## Query 7b

|  |
| --- |
|  |
| select author.name, authored.author\_name, count(\*) |
| from author join authored | |
| on author.name = authored.author\_name | |
| where author.name in ( | |
| select distinct authored.author\_name | |
| from authored join publications on authored.publication\_key = publications.pubKey | |
| where publications.publishedyear = (select min(publishedyear) from publications) | |
| ) | |
| group by author.name, authored.author\_name | |
|  |  | |

TODO: Screen captures of the results of these SQL queries

## Query 8

|  |
| --- |
| create view inpro\_count as( |
| select author\_name, publishedyear, count(\*) as cnt from |
| authored join publications on authored.publication\_key = publications.pubKey |
| where publications.category = 'inproceedings' |
| group by author\_name, publishedyear |
| ); |
|  |
| select author\_name, cnt |
| from inpro\_count |
| where cnt = ( |
| select cnt |
| from inpro\_count |
| order by cnt desc limit 1 offset 1 |
| ) |

TODO: Screen captures of the results of these SQL queries