

**NANYANG  
TECHNOLOGICAL  
UNIVERSITY**

**Assignment 1 Report**

**CZ4031: Database Design Principles**

**Academic Year 2017/2018**

**Semester 1**

**GROUP ID: 18**

<b>Student Names:</b>	<b>Matric Number:</b>
<b>Huang Jian Wei</b>	<b>U1521567A</b>
<b>Yong Guo Jun</b>	<b>U1440217C</b>
<b>See Xin Yee</b>	<b>U1520918B</b>
<b>Shannon Neo Si Lin</b>	<b>U1521821L</b>

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING  
NANYANG TECHNOLOGICAL UNIVERSITY**

## Table of Contents

1	Introduction.....	3
1.1	Background.....	3
1.2	Assumptions and Constraints.....	3
1.3	ER Diagram .....	4
1.4	SAX PARSER (JAVA).....	5
1.5	Tables Creation .....	9
1.5.1	Articles Table.....	9
1.5.2	Authored Table .....	9
1.5.3	Authors Table.....	9
1.5.4	Books Table .....	10
1.5.5	Incollections Table.....	10
1.5.6	Inproceedings Table.....	10
1.5.7	Proceedings Table.....	11
1.5.8	Publications Table.....	11
2	Queries and Optimizing Queries.....	12
2.1	SQL Queries.....	12
2.1.1	Question 1 .....	12
2.1.2	Question 2 .....	13
2.1.3	Question 3 .....	16
2.1.4	Question 4 .....	19
2.1.5	Question 5 .....	24
2.1.6	Question 6 .....	25
2.1.7	Question 7 .....	29
2.1.8	Question 8 .....	30
2.1.9	Question 9 .....	32
2.1.10	Question 10 .....	34
2.2	Queries Analysis .....	36
3	Build Index and Study the Effect of Index.....	37
3.1	CREATE INDEX Statements .....	37
3.2	In-depth look at how Index are used on each query.....	39
4	Advanced Part: Study the Effect of Cache.....	44

# **1 Introduction**

## **1.1 Background**

For this assignment, we will be using PostgreSQL and PgAdmin as our administration tool. Queries are run on a Home Desktop environment on an Intel® Core™ i5-6400 CPU @ 2.7GHz with 16GB RAM. Data are stored in a local D: drive.

## **1.2 Assumptions and Constraints**

All queries are running on the same machine and factors that may affect run time may be due to background processes.

For our queries, we assume that the following terms are as such:

Conference: refers to publications that contains “conf”, typically inproceedings and proceedings.

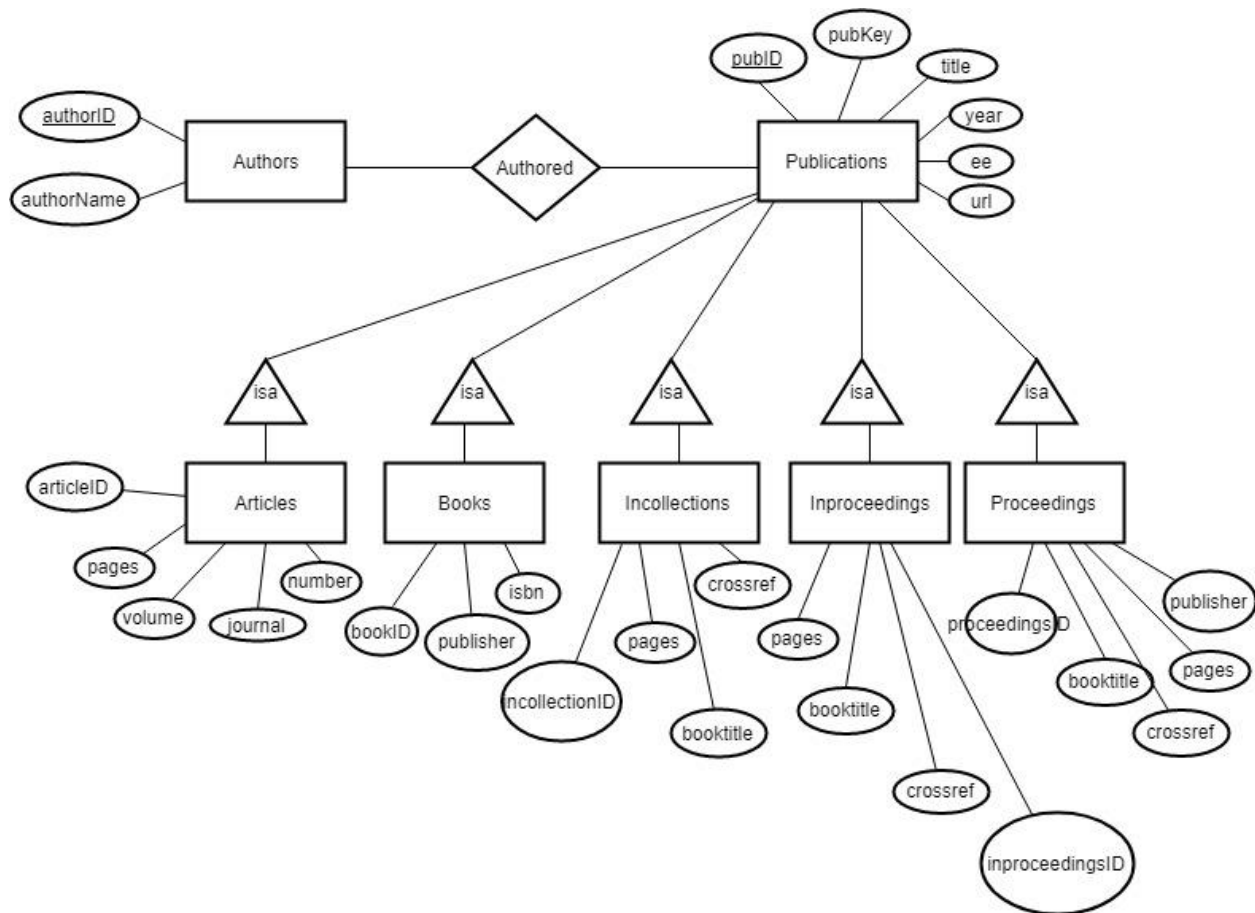
Journals: refers to publications that contains “journals”, typically articles.

Earliest publications record defined as the records that are published in the earliest year.

Month attributes are extracted from m-date attributes.

We half our tables/data by deleting rows by pubID > half digit to ensure data consistency throughout all the tables.

### 1.3 ER Diagram



Authors(authorID, authorName)

Authored(authorID, pubID)

Publications(pubID, pubKey, title, year, ee, url)

Articles(articleID, pages, volume, journal, number, *pubID*)

Books(bookID, publisher, isbn, *pubID*)

Incollections(incollectionID, pages, crossref, booktitle, *pubID*)

Inproceedings(inproceedingsID, pages, crossref, booktitle, *pubID*)

Proceedings(proceedingsID, pages, crossref, booktitle, publisher, *pubID*)

## 1.4 SAX PARSER (JAVA)

The main objective of this section is to write a parser to extract the data from dblp.xml into respective .csv file to import to our database. We are using Java to implement our SAX parser. On average, our parser took 67 seconds to output the respective csv files after reading dblp.xml.

```

MainParser.java
1 package xmlparser;
2
3 import java.io.BufferedOutputStream;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28 public class MainParser {
29
30     public static void main(String[] args) {
31         // TODO Auto-generated method stub
32
33         //To record runtime of the program
34         long start = System.currentTimeMillis();
35
36         try {
37             //Name of file to be read
38             String xmlFile = "dblp.xml";
39
40             //Name of output files
41             File publicationsCSV = new File("publications.csv");
42             File authorsCSV = new File("authors.csv");
43             File authoredCSV = new File("authored.csv");
44             File articlesCSV = new File("articles.csv");
45             File booksCSV = new File("books.csv");
46             File incollectionCSV = new File("incollection.csv");
47             File inproceedingsCSV = new File("inproceedings.csv");
48             File proceedingsCSV = new File("proceedings.csv");
49
50             //PrintWriters for respective .csv output
51             PrintWriter pubWriter = new PrintWriter(new OutputStreamWriter(new BufferedOutputStream
52                 (new FileOutputStream(publicationsCSV)), "UTF8"));
53             PrintWriter authoredWriter = new PrintWriter(new OutputStreamWriter(new BufferedOutputStream
54                 (new FileOutputStream(authoredCSV)), "UTF8"));
55             PrintWriter authorsWriter = new PrintWriter(new OutputStreamWriter(new BufferedOutputStream
56                 (new FileOutputStream(authorsCSV)), "UTF8"));
57             PrintWriter articlesWriter = new PrintWriter(new OutputStreamWriter(new BufferedOutputStream
58                 (new FileOutputStream(articlesCSV)), "UTF8"));
59             PrintWriter booksWriter = new PrintWriter(new OutputStreamWriter(new BufferedOutputStream
60                 (new FileOutputStream(booksCSV)), "UTF8"));
61             PrintWriter incollectionsWriter = new PrintWriter(new OutputStreamWriter(new BufferedOutputStream
62                 (new FileOutputStream(incollectionCSV)), "UTF8"));
63             PrintWriter inproceedingsWriter = new PrintWriter(new OutputStreamWriter(new BufferedOutputStream
64                 (new FileOutputStream(inproceedingsCSV)), "UTF8"));
65             PrintWriter proceedingsWriter = new PrintWriter(new OutputStreamWriter(new BufferedOutputStream
66                 (new FileOutputStream(proceedingsCSV)), "UTF8"));

```

```

MainParser.java
68
69 //JAVA SAX PARSER Object
70 InputSource input = new InputSource(xmlFile);
71 SAXParserFactory factory = SAXParserFactory.newInstance();
72 SAXParser saxParser = factory.newSAXParser();
73 XMLReader reader = saxParser.getXMLReader();
74 reader.setErrorHandler(new MyErrorHandler());
75 MySaxParser sp = new MySaxParser(pubWriter,authoredWriter,authorsWriter,articlesWriter,booksWriter,incollectionsWriter,inproceedingsWriter,proceedingsWriter);
76 reader.setContentHandler(sp);
77 reader.parse(input);
78
79 //Close PrintWriters
80 pubWriter.close();
81 authoredWriter.close();
82 authorsWriter.close();
83 articlesWriter.close();
84 booksWriter.close();
85 incollectionsWriter.close();
86 inproceedingsWriter.close();
87 proceedingsWriter.close();
88
89 //Printing out duration of program
90 long end = System.currentTimeMillis();
91 NumberFormat formatter = new DecimalFormat("#0.00000");
92 System.out.print("Execution time is " + formatter.format((end - start) / 1000d) + " seconds");
93
94 catch(Exception e){
95     e.printStackTrace();
96 }
97
98
99
100
101
102
103
104
105
106
107
108
109
110

```

## CZ4031 Assignment 1 Report: Querying Databases Efficiently

```

MainParser.java
110
111 public static class MySaxParser extends DefaultHandler {
112
113     PrintWriter pubWriter, authoredWriter, authorsWriter, articlesWriter, booksWriter, incollectionsWriter
114     , inproceedingsWriter, proceedingsWriter;
115
116     Map<String, String> map = new HashMap<String, String>();
117     Hashtable ht = new Hashtable();
118     Hashtable htwo = new Hashtable();
119     int pubID = 0;
120     int articleID = 0;
121     int bookID = 0;
122     int incollectionID = 0;
123     int inproceedingsID = 0;
124     int proceedingsID = 0;
125     int authorID = 0;
126
127     String elementName;
128     boolean insideStartElement;
129
130
131     public MySaxParser(PrintWriter pubWriter, PrintWriter authoredWriter, PrintWriter authorsWriter
132     , PrintWriter articlesWriter, PrintWriter booksWriter, PrintWriter incollectionsWriter
133     , PrintWriter inproceedingsWriter, PrintWriter proceedingsWriter){
134
135         this.pubWriter = pubWriter;
136         this.authoredWriter = authoredWriter;
137         this.authorsWriter = authorsWriter;
138         this.articlesWriter = articlesWriter;
139         this.booksWriter = booksWriter;
140         this.incollectionsWriter = incollectionsWriter;
141         this.inproceedingsWriter = inproceedingsWriter;
142         this.proceedingsWriter = proceedingsWriter;
143
144     }
145
146
147
148 public void startDocument(){
149     pubWriter.println("pubID,pubKey,title,year,month,ee,url");
150     authorsWriter.println("authorID,authorName");
151     authoredWriter.println("authorID,pubID");
152     articlesWriter.println("articleID,pages,volume,journal,number,pubID");
153     booksWriter.println("bookID,publisher,isbn,pubID");
154     incollectionsWriter.println("incollectionID,pages,booktitle,isbn,crossref,pubID");
155     inproceedingsWriter.println("inproceedingsID,pages,crossref,booktitle,pubID");
156     proceedingsWriter.println("proceedingsID,pages,crossref,booktitle,publisher,pubID");
157
158 }
159
160
161 @Override
162 public void startElement(String namespaceURI, String localName, String qName, Attributes attributes) throws SAXException {
163     elementName = qName;
164
165     if(qName.equals("article") || qName.equals("book") || qName.equals("incollection") || qName.equals("inproceedings")
166     || qName.equals("proceedings")){
167
168         pubID++;
169         insideStartElement = true;
170         //for each attribute,
171         for(int i = 0; i < attributes.getLength(); i++){
172
173             if(attributes.getQName(i).equals("mdate")){
174                 String month = attributes.getValue(i).substring(5,7);
175                 if(month.equals("01")){
176                     month = "January";
177                 }
178                 else if(month.equals("02")){
179                     month = "February";
180                 }
181                 else if(month.equals("03")){
182                     month = "March";
183                 }
184                 else if(month.equals("04")){
185                     month = "April";
186                 }
187                 else if(month.equals("05")){
188                     month = "May";
189                 }
190                 else if(month.equals("06")){

```

## CZ4031 Assignment 1 Report: Querying Databases Efficiently

```
191         month = "June";
192     }else
193     if(month.equals("07")){
194         month = "July";
195     }else
196     if(month.equals("08")){
197         month = "August";
198     }else
199     if(month.equals("09")){
200         month = "September";
201     }else
202     if(month.equals("10")){
203         month = "October";
204     }else
205     if(month.equals("11")){
206         month = "November";
207     }else
208     if(month.equals("12")){
209         month = "December";
210     }
211     map.put(attributes.getQName(i),month);
212 }
213
214 }
215
216 else
217     map.put(attributes.getQName(i),attributes.getValue(i));
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
```

```
MainParser.java
230
231 @Override
232 public void endElement(String namespaceURI,String localname,String qName){
233
234     if(qName.equals("article") || qName.equals("book") || qName.equals("incollection") || qName.equals("inproceedings") || qName.equals("proceedings")){
235
236         insideStartElement= false;
237         String pubOutput = "";
238         String articleOutput = "";
239         String booksOutput = "";
240         String incollectionsOutput = "";
241         String inproceedingsOutput = "";
242         String proceedingsOutput = "";
243
244         //publications csv will always be written
245         pubOutput = pubID + "," + map.get("key") + //get pubkey
246             "\n" + map.get("title").replace("\\", "").replace("\n", "") + //get title
247             "\n" + map.get("year") + //get year
248             "\n" + map.get("mdate") +
249             "\n" + map.get("ee").replace("\\", "") + "\n" + //get ucl, could be empty
250             map.get("uri").replace("\\", "") + "\n"; //get ee could be empty
251
252         //if publication is an article, write to article csv
253         if(qName.equals("article")){
254             articleID++;
255             articleOutput = articleID + //unique key for article table
256                 "\n" + map.get("pages") + //get pages
257                 "\n" + map.get("volume") + //get volume
258                 "\n" + map.get("journal") + //get journal
259                 "\n" + map.get("number") + "\n"; //get number
260             pubID; //pubID foreignkey
261
262             //System.out.println(pubOutput);
263             pubWriter.println(pubOutput);
264             articlesWriter.println(articleOutput);
265         }
266
267         //if publication is a book, write to book csv
268         if(qName.equals("book")){
269             bookID++;
270         }
271     }
272 }
```

## CZ4031 Assignment 1 Report: Querying Databases Efficiently

```

273         booksOutput = bookID +
274             "\n" + map.get("publisher")+
275             "\n" + map.get("isbn")+ "\n" +
276             pubID;
277
278         pubWriter.println(pubOutput);
279         booksWriter.println(booksOutput);
280     }
281
282
283     //if publication is a incollection
284     if(qName.equals("incollection")){
285         incollectionID++;
286
287         incollectionsOutput = incollectionID +
288             "\n" + map.get("pages") +
289             "\n" + map.get("booktitle").replace("\n", "") +
290             "\n" + map.get("isbn")+ "\n" +
291             "\n" + map.get("crossref") + "\n" +
292             pubID;
293
294         pubWriter.println(pubOutput);
295         incollectionsWriter.println(incollectionsOutput);
296     }
297
298
299     //if publication is a inproceedings
300     if(qName.equals("inproceedings")){
301         inproceedingsID++;
302
303         inproceedingsOutput = inproceedingsID +
304             "\n" + map.get("pages") +
305             "\n" + map.get("crossref") +
306             "\n" + map.get("booktitle").replace("\n", "") +
307             "\n" + pubID;
308
309         pubWriter.println(pubOutput);
310         inproceedingsWriter.println(inproceedingsOutput);
311     }
312

```

```

MainParser.java
334 @Override
335 public void characters(char ch[], int start, int length) throws SAXException {
336
337     String tempString = new String(ch, start, length).trim();
338
339
340     if(!tempString.equals("")){
341         if(elementName.equals("author") || elementName.equals("editor") && insideStartElement ){
342             //System.out.println(tempString);
343             Object result = ht.get(tempString);
344             //add a new entry if null
345
346             //tempString = author's name
347             //result = author ID
348
349             if(result==null){
350
351
352                 authorID++;
353                 ht.put(tempString, authorID);
354                 htwo.put(authorID, pubID);
355                 authoredWriter.println(authorID + "," + pubID);
356                 tempString = tempString.replace("\n", "NoPubIDReference");//handle exceptional cases
357                 authorsWriter.println(authorID + "," + tempString);
358
359
360             }
361             else if(pubID!= (int)htwo.get(result)){
362                 //System.out.println(result + "," + pubID);
363                 //ht.put(tempString, authorID);
364                 htwo.put(result, pubID);
365                 authoredWriter.println(result + "," + pubID);
366             }
367         }
368
369         if(insideStartElement){
370             map.put(elementName,tempString);
371         }
372     }
373
374
375
376

```



## 1.5 Tables Creation

### 1.5.1 Articles Table

```
CREATE TABLE public.articles
(
  "articleID" integer NOT NULL DEFAULT nextval("articles_articleID_seq"::regclass),
  pages text,
  volume text,
  journal text,
  "number" text,
  "pubID" integer,
  CONSTRAINT articles_pkey PRIMARY KEY ("articleID"),
  CONSTRAINT "pubID" FOREIGN KEY ("pubID")
  REFERENCES public.publications ("pubID") MATCH SIMPLE
)
```

### 1.5.2 Authored Table

```
CREATE TABLE public.authored
(
  "authorID" integer NOT NULL,
  "pubID" integer NOT NULL,
  CONSTRAINT authored_pkey PRIMARY KEY ("authorID", "pubID"),
  CONSTRAINT "FK_authorID" FOREIGN KEY ("authorID")
  REFERENCES public.authors ("authorID") MATCH SIMPLE
  CONSTRAINT "FK_pubID" FOREIGN KEY ("pubID")
  REFERENCES public.publications ("pubID") MATCH SIMPLE
)
```

### 1.5.3 Authors Table

```
CREATE TABLE public.authors
(
  "authorID" integer NOT NULL DEFAULT nextval("authors_authorID_seq"::regclass),
  "authorName" text COLLATE pg_catalog."default",
  CONSTRAINT authors_pkey PRIMARY KEY ("authorID")
)
```

### 1.5.4 Books Table

```
CREATE TABLE public.books
(
  "bookID" integer NOT NULL DEFAULT nextval("books_bookID_seq"::regclass),
  publisher text COLLATE pg_catalog."default",
  isbn text,
  "pubID" integer,
  CONSTRAINT books_pkey PRIMARY KEY ("bookID"),
  CONSTRAINT "pubID" FOREIGN KEY ("pubID")
    REFERENCES public.publications ("pubID") MATCH SIMPLE
)
```

### 1.5.5 Incollections Table

```
CREATE TABLE public.incollections
(
  "incollectionsID" integer NOT NULL DEFAULT nextval("incollections_incollectionID_seq"::regclass),
  pages text,
  booktitle text,
  isbn text,
  crossref text,
  "pubID" integer,
  CONSTRAINT incollections_pkey PRIMARY KEY ("incollectionsID"),
  CONSTRAINT "pubID" FOREIGN KEY ("pubID")
    REFERENCES public.publications ("pubID") MATCH SIMPLE
)
```

### 1.5.6 Inproceedings Table

```
CREATE TABLE public.inproceedings
(
  "inproceedingsID" integer NOT NULL DEFAULT nextval("inproceedings_inproceedingsID_seq"::regclass),
  pages text COLLATE pg_catalog."default",
  crossref text COLLATE pg_catalog."default",
  booktitle text COLLATE pg_catalog."default",
  "pubID" integer,
  CONSTRAINT inproceedings_pkey PRIMARY KEY ("inproceedingsID"),
  CONSTRAINT "pubID" FOREIGN KEY ("pubID")
    REFERENCES public.publications ("pubID") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
)
```

### 1.5.7 Proceedings Table

```
CREATE TABLE public.proceedings
(
  "proceedingsID" integer NOT NULL DEFAULT nextval("proceedings_proceedingID_seq"::regclass),
  pages text,
  crossref text,
  booktitle text,
  publisher text,
  "pubID" integer,
  CONSTRAINT proceedings_pkey PRIMARY KEY ("proceedingsID"),
  CONSTRAINT "pubID" FOREIGN KEY ("pubID")
    REFERENCES public.publications ("pubID") MATCH SIMPLE
)
```

### 1.5.8 Publications Table

```
CREATE TABLE public.publications
(
  "pubID" integer NOT NULL DEFAULT nextval("publications_pibID_seq"::regclass),
  "pubKey" text,
  title text,
  year text,
  month text,
  ee text,
  url text,
  CONSTRAINT publications_pkey PRIMARY KEY ("pubID")
)
```

We then use Pgadmin's import function to copy and insert values from the .csv files into our table.

## 2 Queries and Optimizing Queries

### 2.1 SQL Queries

#### 2.1.1 Question 1

**For each type of publication, count the total number of publications of that type between 2000-2017. Your query should return a set of (publication-type, count) pairs. For example (article, 20000), (inproceedings, 30000), ... (not the real answer).**

Solution: COUNT the number of rows in each sub tables (articles, proceedings, inproceedings, books, incollections) where year >= 2000 AND year <= 2017 and join them.

```
(SELECT 'Article' AS "Publication-type", COUNT(*) AS "Count" FROM articles
INNER JOIN publications ON "articles"."pubID" = "publications"."pubID"
WHERE year >= '2000' AND year <= '2017')
UNION
(SELECT 'Book' AS "Publication-type", COUNT(*) AS "Count" FROM books
INNER JOIN publications ON "books"."pubID" = "publications"."pubID"
WHERE year >= '2000' AND year <= '2017')
UNION
(SELECT 'Incollection' AS "Publication-type", COUNT(*) AS "Count" FROM incollections
INNER JOIN publications ON "incollections"."pubID" = "publications"."pubID"
WHERE year >= '2000' AND year <= '2017')
UNION
(SELECT 'Proceedings' AS "Publication-type", COUNT(*) AS "Count" FROM proceedings
INNER JOIN publications ON "proceedings"."pubID" = "publications"."pubID"
WHERE year >= '2000' AND year <= '2017')
UNION
(SELECT 'Inproceedings' AS "Publication-type", COUNT(*) AS "Count" FROM inproceedings
INNER JOIN publications ON "inproceedings"."pubID" = "publications"."pubID"
WHERE year >= '2000' AND year <= '2017');
```

	Publication-type text	Count bigint
1	Article	1401146
2	Incollection	43919
3	Inproceedings	1759048
4	Book	8502
5	Proceedings	29456

### 2.1.2 Question 2

**Find all the conferences that have ever published more than 200 papers in one year and are held in July. Note that one conference may be held every year (e.g., KDD runs many years, and each year the conference has a number of papers).**

Solution: We will solve this question by splitting the query into smaller steps:

- 1.) Filter out only conference that are held in JULY
- 2.) GROUP BY year and select rows that has > 200 papers

According to the dblp website, proceedings and inproceedings are classified under conference, hence we will be selecting data from proceedings and inproceedings tables. We will also be using attribute month obtained from “mdate”.

First, we select conferences that are held in ‘JULY’.

```
CREATE VIEW p_conf_papers_in_july AS (
SELECT "publications"."title","publications"."year","proceedings"."proceedingsID",booktitle,"proceedings"."pubID" FROM proceedings
INNER JOIN publications ON "proceedings"."proceedingsID" = "publications"."pubID" WHERE "publications"."month" = 'July'
)

CREATE VIEW ip_conf_papers_in_july AS (
SELECT "publications"."title","publications"."year","inproceedings"."inproceedingsID",booktitle,"inproceedings"."pubID" FROM inproceedings
INNER JOIN publications ON "inproceedings"."inproceedingsID" = "publications"."pubID" WHERE "publications"."month" = 'July'
)
```

Sample Resulting View (Actual result has 77924 rows)

	title text	year text	inproceedin integer	booktitle text	pubID integer
1	Efficient an...	2017	244	ICCS	815822
2	n/3, O(n	2017	467	Complex Adaptive Syst...	816045
3	On the rela...	2017	542	WCIT	816120
4	Reversible ...	2017	631	Complex Adaptive Syst...	816209
5	Reactive sy...	2017	877	Complex Adaptive Syst...	816455
6	Pseudopoly...	2017	1077	CSER	816655

Next, we group by years to see how many publications were published in a conference in July.

```
CREATE VIEW sub_p_group_conf_papers_abv_200_in_july AS (
SELECT booktitle,"p_conf_papers_in_july"."year",COUNT(*) as "NUMPAPERS" from p_conf_papers_in_july
GROUP BY booktitle,"p_conf_papers_in_july"."year"
)
```

```
CREATE VIEW sub_ip_group_conf_papers_abv_200_in_july AS (
SELECT booktitle,"ip_conf_papers_in_july"."year",COUNT(*) as "NUMPAPERS" from ip_conf_papers_in_july
GROUP BY booktitle,"ip_conf_papers_in_july"."year"
)
```

Finally, we select rows where “NUMPAPERS” > 200 and UNION the views.

```
CREATE VIEW ip_conf_papers_abv_200_in_july AS (
SELECT * FROM sub_ip_group_conf_papers_abv_200_in_july
WHERE "sub_ip_group_conf_papers_abv_200_in_july"."NUMPAPERS" > '200'
)

CREATE VIEW p_conf_papers_abv_200_in_july AS (
SELECT * FROM sub_p_group_conf_papers_abv_200_in_july
WHERE "sub_p_group_conf_papers_abv_200_in_july"."NUMPAPERS" > '200'
)

CREATE VIEW conf_abv_200_in_july AS (
SELECT "ip_conf_papers_abv_200_in_july"."booktitle" AS "CONFERENCE_NAME", "ip_conf_papers_abv_200_in_july"."year",
```

## CZ4031 Assignment 1 Report: Querying Databases Efficiently

```
"ip_conf_papers_abv_200_in_july"."NUMPAPERS" FROM ip_conf_papers_abv_200_in_july
```

```
UNION ALL
```

```
SELECT "p_conf_papers_abv_200_in_july"."booktitle" AS "CONFERENCE_NAME", "p_conf_papers_abv_200_in_july"."year",
```

```
"p_conf_papers_abv_200_in_july"."NUMPAPERS" FROM p_conf_papers_abv_200_in_july)
```

Final result:

Data Output <a href="#">Explain</a> <a href="#">Messages</a> <a href="#">History</a>			
	CONFERENCE_NAME text	year text	NUMPAPERS bigint
1	ASICON	2008	227
2	ASICON	2017	269
3	BCS HCI	2011	629
4	CHI	2009	240
5	CHI Extended Abstracts	2009	283
6	CIKM	2006	226
7	CIKM	2008	408
8	CIKM	2009	1030
9	CIKM	2010	336
10	CIKM	2011	552
11	CIKM	2014	873
12	CIKM	2015	226
13	CIKM	2017	343
14	CVPR	2005	361
15	CVPR	2011	752
16	CVPR Workshops	2011	276
17	EUSIPCO	2014	307
18	FTCS	2009	321
19	GLOBECOM	2016	458
20	GLOBECOM	2017	766
21	HIPC	2005	622
22	ICC	2017	281
23	IGARSS	2017	213
24	IHM	2008	282
26	ITNG	2009	1806
27	LICS	2017	217
28	PIMRC	2017	238
29	PPDP	2016	216
30	PRO-VE	2009	284
31	PRO-VE	2014	260
32	RANLP	2008	217
33	RANLP	2017	248
34	SAGT	2014	225
35	SIGIR	2017	226
36	TSD	2008	476
37	TSD	2016	416
38	UMAP	2009	463
39	UMAP Workshops	2009	255
40	User Modeling	2009	267
41	WEIS	2009	226

## 2.1.3 Question 3

### 2.1.3.1 Question 3a

Find the publications of author = “X” (replace X with a real name in your database) at year 2015 (List all the available information of each publication).

Solution: For our question, we will be using the author “Xiaokui Shu” as X.

We will first need to join authors and publications using the authored table. After that, we must list all information of each publication (unique to their subtype), hence we will be using OUTER JOIN to fulfil these criteria and CONCAT similar columns.

CZ4031 on postgres@PostgreSQL 9.6

```

1 CREATE VIEW authors_authored AS (
2   SELECT "authors"."authorID" FROM authors INNER JOIN authored ON "authors"."authorID" = "authored"."authorID"
3 )
4 CREATE VIEW authors_authored_publications AS (
5   SELECT
6     "authors_authored"."authorID", "authors_authored"."authorName", "authors_authored"."pubID",
7     "publications"."year", "publications"."title", "publications"."ee", "publications"."url", "publications"."month" FROM authors_authored
8   JOIN publications ON authors_authored."pubID" = publications."pubID"
9 )
10
11 CREATE VIEW all_publications_by_X AS (
12   (SELECT "authors_authored_publications"."authorID", "authors_authored_publications"."authorName", "authors_authored_publications"."year",
13     "authors_authored_publications"."title", "authors_authored_publications"."pubID",
14     "articles"."articleID", "books"."bookID", "incollections"."incollectionsID", "proceedings"."proceedingsID", "inproceedings"."inproceedingsID",
15     CONCAT("inproceedings"."pages", "articles"."pages", "proceedings"."pages", "inproceedings"."pages") AS "pages", CONCAT("inproceedings"."crossref",
16     "proceedings"."crossref", "incollections"."crossref") AS "crossref", CONCAT("inproceedings"."booktitle", "proceedings"."booktitle") AS "booktitle",
17     "articles"."volume", "articles"."journal", "articles"."number",
18     "authors_authored_publications"."ee", "authors_authored_publications"."url",
19     CONCAT("books"."publisher", "proceedings"."publisher") AS "publisher",
20     CONCAT("books"."isbn", "incollections"."isbn") AS "isbn"
21   FROM authors_authored_publications
22   full OUTER JOIN articles ON "authors_authored_publications"."pubID" = "articles"."pubID"
23   full OUTER JOIN books ON "authors_authored_publications"."pubID" = "books"."pubID"
24   full OUTER JOIN incollections ON "authors_authored_publications"."pubID" = "incollections"."pubID"
25   full OUTER JOIN inproceedings ON "authors_authored_publications"."pubID" = "inproceedings"."pubID"
26   full OUTER JOIN proceedings ON "authors_authored_publications"."pubID" = "proceedings"."pubID"
27   WHERE "authors_authored_publications"."authorName" = 'Xiaokui Shu' AND "authors_authored_publications"."year" = '2015')
28 )

```

	authorID integer	authorName text	year integer	title text	pubID integer	articleID integer	bookID integer	incollectionsID integer	proceedingsID integer	inproceedingsID integer	pages text	crossref text	booktitle text	volume text	journal text	number text	ee text	url text	publisher text	isbn text
1	76502	Xiaokui Shu	2015	Un...	2007866	[null]	[null]	[null]	[null]	267743	401-4...	conf/ccs...	ACM Co...	[null]	[null]	[null]	htt...	db/...		
2	76502	Xiaokui Shu	2015	Priv...	2199525	[null]	[null]	[null]	[null]	455547	195-2...	conf/cod...	CODASPY	[null]	[null]	[null]	htt...	db/...		
3	76502	Xiaokui Shu	2015	Rap...	2199666	[null]	[null]	[null]	[null]	455684	147-1...	conf/cod...	CODASPY	[null]	[null]	[null]	htt...	db/...		
4	76502	Xiaokui Shu	2015	Rap...	2218454	[null]	[null]	[null]	[null]	474190	191-1...	conf/inf...	INFOCO...	[null]	[null]	[null]	htt...	db/...		
5	76502	Xiaokui Shu	2015	A F...	2720407	[null]	[null]	[null]	[null]	967180	270-2...	conf/rai...	RAID	[null]	[null]	[null]	htt...	db/...		
6	76502	Xiaokui Shu	2015	Pro...	1185493	1180957	[null]	[null]	[null]	[null]	255-2...			49	Security	1	htt...	db/...		
7	76502	Xiaokui Shu	2015	Priv...	46616	46616	[null]	[null]	[null]	[null]	1092-...			10	IEEE Tr...	5	htt...	db/...		



### 2.1.3.2 Question 3b

**Find the publications of author = “X” (replace X with a real name in your database) at year “Y” at conference “Z” (replace Y and Z with real value so that the query will return some tuples as results.)**

Solution: First to find out some real values for Z (So that there will be result returns), we pick “Y” as 2010 and ran a simple query to find conference that are held at 2015 using:

```
SELECT * FROM publications
INNER JOIN inproceedings ON "publications"."pubID" = "inproceedings"."pubID"
WHERE "publications"."year" = '2010' AND "publications"."pubKey" like '%conf%'
```

After which we pick “Z” as “SIGMETRICS”.

```
SELECT * FROM publications
INNER JOIN inproceedings ON "publications"."pubID" = "inproceedings"."pubID"
WHERE "publications"."year" = '2010' AND "inproceedings"."booktitle" = 'SIGMETRICS'
```

Next, we will create a joined view between conference and authors\_authored\_publication view that we previously created.

```
CREATE VIEW authors_authored_conf AS (
SELECT
"authors_authored_publications"."pubID","authors_authored_publications"."authorID","authors_authored_publications"."authorName","authors_authored_publications"."year","authors_authored_publications"."title",
"authors_authored_publications"."ee","authors_authored_publications"."url","inproceedings"."booktitle","inproceedings"."crossref","inproceedings"."pages"
FROM authors_authored_publications INNER JOIN inproceedings ON "authors_authored_publications"."pubID" = "inproceedings"."pubID"
)
```

Lastly, we will then replace “X” with “Peter Nobel” and run a final query.

```
SELECT * from authors_authored_conf WHERE "authors_authored_conf"."authorName" = 'Peter Nobel'
AND "authors_authored_conf"."year" = '2010' AND "authors_authored_conf"."booktitle" = 'SIGMETRICS'
```

To produce the final result:

```
15 SELECT * from authors_authored_conf WHERE "authors_authored_conf"."authorName" = 'Peter Nobel'
16 AND "authors_authored_conf"."year" = '2010' AND "authors_authored_conf"."booktitle" = 'SIGMETRICS'
17
18
```

Data Output										
	pubID	authorID	authorName	year	title	ee	url	booktitle	crossref	pages
	integer	integer	text	text	text	text	text	text	text	text
1	2020517	274704	Peter Nobel	2010	Pra...	htt...	db/...	SIGMETR...	conf/sig...	1-12

### 2.1.3.3 Question 3c

#### Find authors who published at least 2 papers at conference “Z” at year “Y”

For this question, we set “Y” to be 2015 else query will take too long.

Using previous view “authors\_authored\_conf”, we create a new View that displays all authors that published papers in ‘2015’.

```
CREATE VIEW authors_publish_atLeast_2papers AS (
SELECT "authors_authored_conf"."authorName","authors_authored_conf"."title","authors_authored_conf"."year", COUNT(*) AS
"NUMPAPER" FROM authors_authored_conf
WHERE "authors_authored_conf"."year" = '2015'
GROUP BY "authors_authored_conf"."authorName","authors_authored_conf"."title","authors_authored_conf"."year"
)
```

After which, using:

```
SELECT * FROM authors_publish_atLeast_2papers WHERE "NUMPAPER" > 1
```

On “authors\_publish\_atLeast\_2papers view, it will return a result of 643 rows.

Sample data of 25 rows:

```
1 CREATE VIEW authors_publish_atLeast_2papers AS (
2 SELECT "authors_authored_conf"."authorName","authors_authored_conf"."title","authors_authored_conf"."year", COUNT(*) AS "NUMPAPER" FROM authors_authored_conf
3 WHERE "authors_authored_conf"."year" = '2015'
4 GROUP BY "authors_authored_conf"."authorName","authors_authored_conf"."title","authors_authored_conf"."year"
5 )
6
7 SELECT * FROM authors_publish_atLeast_2papers WHERE "NUMPAPER" > 1
```

Data Output	Explain	Messages	History
authorName text	title text	year text	NUMPAPER bigint
1 Aayush Gupta	Pricing Games for Hybrid Object Stores in the Cloud: Provider v...	2015	2
2 ábor Lencse	Modelling of virtualized servers.	2015	2
3 Abraham Prieto	Embodied Evolution for Collective Indoor Surveillance and Loca...	2015	2
4 ács	Modelling of virtualized servers.	2015	2
5 Adam Wierman	Distributed optimization decomposition for joint economic disp...	2015	2
6 Aditi Majumder	A multi-projector display system of arbitrary shape, size and re...	2015	2
7 Agata Niescieruk	Spiteful Bidding in the Dollar Auction.	2015	2
8 Ahmad Assaf	Roomba: An Extensible Framework to Validate and Build Datas...	2015	2
9 Aihua Zhang	Image Splicing Detection Based on Markov Features in QDCT D...	2015	2
10 Aijun Bai	PLEASE: Palm Leaf Search for POMDPs with Large Observation ...	2015	2
11 Akemi G	ézier Curves.	2015	2
12 Ákos Kov	Modelling of virtualized servers.	2015	2
13 Akshaya Kumar Mishra	DESIRE: Discontinuous energy seam carving for image retarget...	2015	2
14 Ala Shaabana	SICILIA: a smart sensor system for clothing insulation inferenc...	2015	2
15 Alan Ferrari	Poster: Can Smart Devices Protect Us from Violent Crime?	2015	2
16 Albert Lozano	An abstract journey: production focus on the abstract thought ...	2015	2
17 Alberto Broggi	Vision-based Perception for Intelligent Vehicles.	2015	2
18 Alessandro Bianchi	Applying Predicate Abstraction to Abstract State Machines.	2015	2
19 Alex Deng	Objective Bayesian Two Sample Hypothesis Testing for Online ...	2015	2
20 Alex Dilello	RAMP: accelerating wireless sensor hardware design with a re...	2015	2
21 Alexander Fian	Nature as microelectronic fab: Bioelectronics: Materials, transis...	2015	2
22 Alexander Sch	TriAL-QL: Distributed Processing of Navigational Queries.	2015	2
23 Alexander Wong	DESIRE: Discontinuous energy seam carving for image retarget...	2015	2
24 Alexandra A. Pereira Kl...	A Flexible Collaborative Innovation Model for SOA Services Pro...	2015	2
25 Alexandra Ion	Proprioceptive Interaction.	2015	2

## 2.1.4 Question 4

### 2.1.4.1 Question 4a

**Find (a) all authors who published at least 10 PVLDB papers and published at least 10 SIGMOD papers, and (b) all authors who published at least 15 PVLDB papers but never published a KDD paper. (Note that you need to do some digging to find out how DBLP spells the name of conferences and journals).**

Solution: First we create 2 separate views which consist of 1.) authors who published at least 10 PVLDB papers and 2.) authors who published at least 10 SIGMOD papers.

```
CREATE VIEW count_pvldbpaper AS(  
SELECT "authors"."authorName", COUNT(*) AS "pvldbCount" FROM authors  
INNER JOIN authored ON "authors"."authorID" = "authored"."authorID"  
INNER JOIN publications ON "authored"."pubID" = "publications"."pubID"  
WHERE "pubKey" LIKE '%pvldb%'  
GROUP BY "authors"."authorName");
```

```
CREATE VIEW count_sigmodpaper AS(  
SELECT "authors"."authorName", COUNT(*) AS "sigmodCount" FROM authors  
INNER JOIN authored ON "authors"."authorID" = "authored"."authorID"  
INNER JOIN publications ON "authored"."pubID" = "publications"."pubID"  
WHERE "pubKey" LIKE '%sigmod%'  
GROUP BY "authors"."authorName");
```

## CZ4031 Assignment 1 Report: Querying Databases Efficiently

Sample view data:

	authorName text	pvldbCount bigint
1	A. Ghitescu	1
2	A. Pavan	1
3	A. Prasad Sistla	1
4	A. Somani	1
5	Aaditya Kulkarni	1
6	Aameek Singh	1
7	Aamod Sane	1
8	Aapo Kyrola	1
9	Aaron Davidson	2
10	Aaron J. Elmore	9
11	Aaron Zalewski	1
12	Aastha Madaan	1
13	Abdallah Elguindy	1
14	Abdelkader Sellami	2
15	Abdeltawab M. Hendawi	1
16	Abderrahmen Kammoun	1
17	Abdul Quamar	4
18	Abdur Chowdhury	1
19	Abdussalam Alawini	1
20	Abel N. Kho	1

	authorName text	sigmodCount bigint
1	á	9
2	à	3
3	ä	1
4	á Mesquita	1
5	A Min Tjoa	1
6	A. B. Tonik	1
7	A. Bolour	1
8	A. Camille B...	1
9	A. Desai Nar...	1
10	A. El Masri	1
11	A. H. Teger	1
12	A. Howard	1
13	A. J. Korenjak	1
14	A. K. Bhushan	2
15	A. Khivesera	1
16	A. Kumaran ...	1
17	A. L. Dean	2
18	A. Lewis Bas...	2
19	A. Mahboob	1
20	A. Metaxides	1
21	A. Prasad Si...	5
22	A. S	1
23	A. Singer	1
24	A. Timothy M...	1
25	Aakash Goel	1

Finally, to satisfy the 2 constraints, we did a join on the two views.

```
SELECT "authors"."authorName", "pvldbCount", "sigmodCount" FROM authors
LEFT JOIN "count_pvldbpaper" ON "count_pvldbpaper"."authorName" = "authors"."authorName"
LEFT JOIN "count_sigmodpaper" ON "count_sigmodpaper"."authorName" = "authors"."authorName"
WHERE ("pvldbCount" >= '10' AND "sigmodCount" >= '10');
```

Sample data of result:

CZ4031 on postgres@PostgreSQL 9.6

```
1 SELECT "authors"."authorName", "pvldbCount", "sigmodCount" FROM authors
2 LEFT JOIN "count_pvldbpaper" ON "count_pvldbpaper"."authorName" = "authors"."authorName"
3 LEFT JOIN "count_sigmodpaper" ON "count_sigmodpaper"."authorName" = "authors"."authorName"
4 WHERE ("pvldbCount" >= '10' AND "sigmodCount" >= '10');|
```

Data Output

Explain

Messages

History

	authorName text	pvldbCount bigint	sigmodCount bigint
1	Laks V. S. L...	12	27
2	Dan Suciu	21	43
3	David Maier	14	30
4	Shivnath Ba...	13	23
5	Ugur	10	13
6	Jeffrey F. Na...	14	59
7	Jiawei Han O...	20	44
8	Andrew Pavlo	12	13
9	Hector Garci...	15	51
10	Jignesh M. P...	21	25
11	Gao Cong	14	14
12	Michael Ston...	14	77
13	Michael J. Fr...	15	64
14	Neoklis Poly...	13	22
15	Philip S. Yu	12	23
16	Anthony K. ...	13	23
17	Gustavo Alo...	18	20
18	Alon Y. Halevy	10	38
19	Joseph M. H...	17	48
20	Magdalena ...	22	25
21	Xifeng Yan	12	14
22	Yanlei Diao	10	10
23	Anastasia Ai...	23	23
24	Çetintemel	10	13
25	Divyakant A...	17	26

### 2.1.4.2 Question 4b

**All authors who published at least 15 PVLDB papers but never published a KDD paper. (Note that you need to do some digging to find out how DBLP spells the name of conferences and journals).**

Using count\_pvldbpaper view from part 4a), we created a new view which only selects pvldb counts which are more than 15.

```
CREATE VIEW count_pvldbpaper_more_than_15 AS (
SELECT "count_pvldbpaper"."authorName","count_pvldbpaper"."pvldbCount" FROM count_pvldbpaper
WHERE "count_pvldbpaper"."pvldbCount" >= 15)
```

Next, we create views such that it contains list of authors who published KDD before.

```
CREATE VIEW authors_publish_no_kdd_p AS (
SELECT "authors_authored_publications"."authorName" FROM authors_authored_publications
INNER JOIN proceedings ON "authors_authored_publications"."pubID" = "proceedings"."pubID"
WHERE "proceedings"."booktitle" NOT LIKE '%KDD%'
GROUP BY "authors_authored_publications"."authorName")
```

```
CREATE VIEW authors_publish_no_kdd AS (
SELECT "authors_authored_publications"."authorName" FROM authors_authored_publications
INNER JOIN inproceedings ON "authors_authored_publications"."pubID" = "inproceedings"."pubID"
WHERE "inproceedings"."booktitle" NOT LIKE '%KDD%'
GROUP BY "authors_authored_publications"."authorName")
```

Finally, we use INNER JOIN on authors who publish > 15 PVLDB papers and who do not publish KDD, (authors who publish > 15 PVLDB UNION authors who do not publish kdd in proceedings/inproceedings).

```
(SELECT "count_pvldbpaper_more_than_15"."authorName","count_pvldbpaper_more_than_15"."pvldbCount"
FROM count_pvldbpaper_more_than_15 INNER JOIN authors_publish_no_kdd
ON "count_pvldbpaper_more_than_15"."authorName" = "authors_publish_no_kdd"."authorName"
GROUP BY "count_pvldbpaper_more_than_15"."authorName","count_pvldbpaper_more_than_15"."pvldbCount")
UNION
(SELECT "count_pvldbpaper_more_than_15"."authorName","count_pvldbpaper_more_than_15"."pvldbCount"
FROM count_pvldbpaper_more_than_15 INNER JOIN authors_publish_no_kdd_p
ON "count_pvldbpaper_more_than_15"."authorName" = "authors_publish_no_kdd_p"."authorName"
GROUP BY "count_pvldbpaper_more_than_15"."authorName","count_pvldbpaper_more_than_15"."pvldbCount")
```

Another possible solution would probably be using NOT IN but the runtime duration was too long.

```
CREATE VIEW count_pvldbpaper_more_than_15_no_kdd AS(
SELECT "count_pvldbpaper_more_than_15"."authorName","count_pvldbpaper_more_than_15"."pvldbCount"
FROM count_pvldbpaper_more_than_15,authors_publish_no_kdd,authors_publish_no_kdd_p
WHERE "count_pvldbpaper_more_than_15"."authorName" IN ("authors_publish_no_kdd"."authorName") OR
"count_pvldbpaper_more_than_15"."authorName"
IN ("authors_publish_no_kdd_p"."authorName")
GROUP BY "count_pvldbpaper_more_than_15"."authorName","count_pvldbpaper_more_than_15"."pvldbCount")
```

Sample data, actual data contain 37 rows.

CZ4031 on postgres@PostgreSQL 9.6

```

1  (SELECT "count_pvldbpaper_more_than_15"."authorName", "count_pvldbpaper_more_than_15"."pvldbCount"
2  FROM count_pvldbpaper_more_than_15 INNER JOIN authors_publish_no_kdd
3  ON "count_pvldbpaper_more_than_15"."authorName" = "authors_publish_no_kdd"."authorName"
4  GROUP BY "count_pvldbpaper_more_than_15"."authorName", "count_pvldbpaper_more_than_15"."pvldbCount")
5  UNION
6  (SELECT "count_pvldbpaper_more_than_15"."authorName", "count_pvldbpaper_more_than_15"."pvldbCount"
7  FROM count_pvldbpaper_more_than_15 INNER JOIN authors_publish_no_kdd_p
8  ON "count_pvldbpaper_more_than_15"."authorName" = "authors_publish_no_kdd_p"."authorName"
9  GROUP BY "count_pvldbpaper_more_than_15"."authorName", "count_pvldbpaper_more_than_15"."pvldbCount")
10

```

Data Output Explain Messages History

	authorName text	pvldbCount bigint
1	Samuel Mad...	35
2	Divyakant A...	17
3	Ren	17
4	Thomas Neu...	20
5	é	25
6	Gustavo Alo...	18
7	Jeffrey Xu Yu	24
8	Tim Kraska	17
9	Aditya G. Pa...	18
10	Wenfei Fan	19
11	Jiawei Han O...	20
12	Divesh Sriva...	44
13	Michael J. Fr...	15
14	Joseph M. H...	17
15	Amr El Abbadi	15
16	Elke A. Rund...	15
17	Donald Koss...	18
18	Tova Milo	17
19	Xuemin Lin	18
20	H. V. Jagadish	24
21	Nikos Mamo...	19
22	Dan Suciu	21

### 2.1.5 Question 5

**For each 10 consecutive years starting from 1970, i.e., [ 1970, 1979], [ 1980, 1989 ],..., [2010, 2019], compute the total number of conference publications in DBLP in that 10 years. Hint: for this query, you may want to compute a temporary table with all distinct years.**

First, we create a view where we only consider conference publications.

```
CREATE VIEW conf_publications AS(
(SELECT "publications"."pubID","publications"."year" FROM publications INNER JOIN proceedings ON
"publications"."pubID" = "proceedings"."pubID")
UNION
(SELECT "publications"."pubID","publications"."year" FROM publications INNER JOIN inproceedings ON
"publications"."pubID" = "inproceedings"."pubID")
)
```

Next, we create different views for each 10 consecutive years and categorize the conference publications base on the years.

```
CREATE VIEW year7079 AS (SELECT "pubID" FROM conf_publications WHERE "year" >= '1970' AND "year" <= '1979');
CREATE VIEW year8089 AS (SELECT "pubID" FROM conf_publications WHERE "year" >= '1980' AND "year" <= '1989');
CREATE VIEW year9099 AS (SELECT "pubID" FROM conf_publications WHERE "year" >= '1990' AND "year" <= '1999');
CREATE VIEW year0009 AS (SELECT "pubID" FROM conf_publications WHERE "year" >= '2000' AND "year" <= '2009');
CREATE VIEW year1019 AS (SELECT "pubID" FROM conf_publications WHERE "year" >= '2010' AND "year" <= '2019');

(SELECT '1970-1979' AS "TenConsecutiveYears", COUNT(*) AS "PublicationNum" FROM "year7079")
UNION
(SELECT '1980-1989' AS "TenConsecutiveYears", COUNT(*) AS "PublicationNum" FROM "year8089")
UNION
(SELECT '1990-1999' AS "TenConsecutiveYears", COUNT(*) AS "PublicationNum" FROM "year9099")
UNION
(SELECT '2000-2009' AS "TenConsecutiveYears", COUNT(*) AS "PublicationNum" FROM "year0009")
UNION
(SELECT '2010-2019' AS "TenConsecutiveYears", COUNT(*) AS "PublicationNum" FROM "year1019")
ORDER BY "TenConsecutiveYears" ASC;
```



## CZ4031 Assignment 1 Report: Querying Databases Efficiently

CZ4031 on postgres@PostgreSQL 9.6

```

1 CREATE VIEW conf_publications AS(
2 (SELECT "publications"."pubID","publications"."year" FROM publications INNER JOIN proceedings ON
3 "publications"."pubID" = "proceedings"."pubID")
4 UNION
5 (SELECT "publications"."pubID","publications"."year" FROM publications INNER JOIN inproceedings ON
6 "publications"."pubID" = "inproceedings"."pubID")
7 )
8
9 CREATE VIEW year7079 AS (SELECT "pubID" FROM conf_publications WHERE "year" >= '1970' AND "year" <= '1979');
10 CREATE VIEW year8089 AS (SELECT "pubID" FROM conf_publications WHERE "year" >= '1980' AND "year" <= '1989');
11 CREATE VIEW year9099 AS (SELECT "pubID" FROM conf_publications WHERE "year" >= '1990' AND "year" <= '1999');
12 CREATE VIEW year0009 AS (SELECT "pubID" FROM conf_publications WHERE "year" >= '2000' AND "year" <= '2009');
13 CREATE VIEW year1019 AS (SELECT "pubID" FROM conf_publications WHERE "year" >= '2010' AND "year" <= '2019');
14
15
16
17 (SELECT '1970-1979' AS "TenConsecutiveYears", COUNT(*) AS "PublicationNum" FROM "year7079")
18 UNION
19 (SELECT '1980-1989' AS "TenConsecutiveYears", COUNT(*) AS "PublicationNum" FROM "year8089")
20 UNION
21 (SELECT '1990-1999' AS "TenConsecutiveYears", COUNT(*) AS "PublicationNum" FROM "year9099")
22 UNION
23 (SELECT '2000-2009' AS "TenConsecutiveYears", COUNT(*) AS "PublicationNum" FROM "year0009")
24 UNION
25 (SELECT '2010-2019' AS "TenConsecutiveYears", COUNT(*) AS "PublicationNum" FROM "year1019");
26
27
28

```

Data Output Explain Messages History

	TenConsecutiveYears text	PublicationNum bigint
1	2010-2019	1003885
2	1970-1979	12784
3	2000-2009	784628
4	1980-1989	52263
5	1990-1999	216667

### 2.1.6 Question 6

**Find the most collaborative authors who published in a conference or journal whose name contains “data” (e.g., ACM SIGKDD International Conference on Knowledge Discovery and Data Mining). That is, for each author determine its number of collaborators, and then find the author with the most number of collaborators. Hint: for this question you may want to compute a temporary table of coauthors.**

**Solution:** First we select journals and conference which contains data by creating a view where we check booktitle and journals WHERE the attributes ILIKE ‘%DATA%’. After which, using the pubID we INNER JOIN authors table to get authors who published such journals and conference. By doing this, we filter down to only authors who published conference or journal which contains “Data”. This will reduce the number of tuples which are needed for cross join later.

## CZ4031 Assignment 1 Report: Querying Databases Efficiently

```
CREATE VIEW conf_journal_data AS(
SELECT "proceedings"."pubID", "proceedings"."booktitle" FROM proceedings WHERE "proceedings"."booktitle" ILIKE '%DATA%'
UNION
SELECT "inproceedings"."inproceedingsID", "inproceedings"."booktitle" FROM inproceedings WHERE "inproceedings"."booktitle" ILIKE
'%DATA%'
UNION
SELECT "articles"."pubID", "articles"."journal" AS "booktitle" FROM articles WHERE "articles"."journal" ILIKE '%DATA%'
)
```

```
CREATE VIEW pub_conf_journal_data AS (

SELECT "publications"."pubID", "publications"."title"

FROM publications

INNER JOIN conf_journal_data ON "publications"."pubID" = "conf_journal_data"."pubID"

)
```

```
CREATE VIEW author_pub_conf_journal_data AS (
SELECT "authors_authored"."authorID", "authors_authored"."authorName", "pub_conf_journal_data"."pubID", "pub_conf_journal_data"."title"

FROM authors_authored

INNER JOIN pub_conf_journal_data ON "authors_authored"."pubID" = "pub_conf_journal_data"."pubID"

)
```

We then create a collaborator table using a form of cross join.

```
CREATE VIEW collaborator_1 AS(

SELECT a."authorID", b."authorID" as colla_id

FROM author_pub_conf_journal_data a

JOIN author_pub_conf_journal_data b ON a."pubID" = b."pubID" and NOT a."authorID" = b."authorID"

);
```

Actual view has a result of 336652 rows:

	authorID integer	colla_id integer
1	36	23
2	35	23
3	34	23
4	36	34
5	35	34
6	23	34
7	36	35
8	34	35
9	23	35
10	35	36
11	34	36
12	23	36
13	10	43
14	43	10
15	51	50
16	50	51
17	77	76
18	76	77
19	123	122
20	122	123
21	130	129
22	129	130
23	133	132
24	132	133
25	19878	19881

## CZ4031 Assignment 1 Report: Querying Databases Efficiently

```
CREATE VIEW collaborator_count AS(  
  
SELECT "collaborator_1"."authorID", count(DISTINCT "collaborator_1"."colla_id") AS colla_num  
  
FROM collaborator_1  
  
GROUP BY "collaborator_1"."authorID"  
  
ORDER BY colla_num DESC  
  
);
```

As you can see, authorID 367 have a total of 951 collaborations.

	<b>authorID</b> integer	<b>colla_num</b> bigint
1	367	951
2	2643	544
3	203	501
4	96	458
5	1676	418
6	1049	381
7	166	378
8	1347	374
9	4121	344
10	4427	313
11	4065	292
12	2218	263
13	425	263
14	614	247
15	1189	245
16	2169	236
17	4115	236
18	198	226
19	659972	206
20	238628	202
21	3913	199
22	4285	197
23	270455	193
24	919	190
25	589	188

Lastly, we project the author name with the highest collaborator count.

```
SELECT "authors"."authorName"  
FROM collaborator_count  
JOIN authors  
ON "collaborator_count"."authorID" = "authors"."authorID" AND colla_num = (SELECT MAX(colla_num) FROM  
collaborator_count)
```

	<b>authorName</b> text
1	Jos

### 2.1.7 Question 7

**Data analytics and data science are very popular topics. Find the top 10 authors with the largest number of publications that are published in conferences and journals whose titles contain word “Data” in the last 5 years.**

Solution: First, we filter by publications which must contain DATA in the past 5 years (2013-2017).

```
CREATE VIEW data5y AS(
SELECT * FROM publications
WHERE "title" LIKE '%Data%' AND "year" >= '2013' AND "year" <= '2017');
```

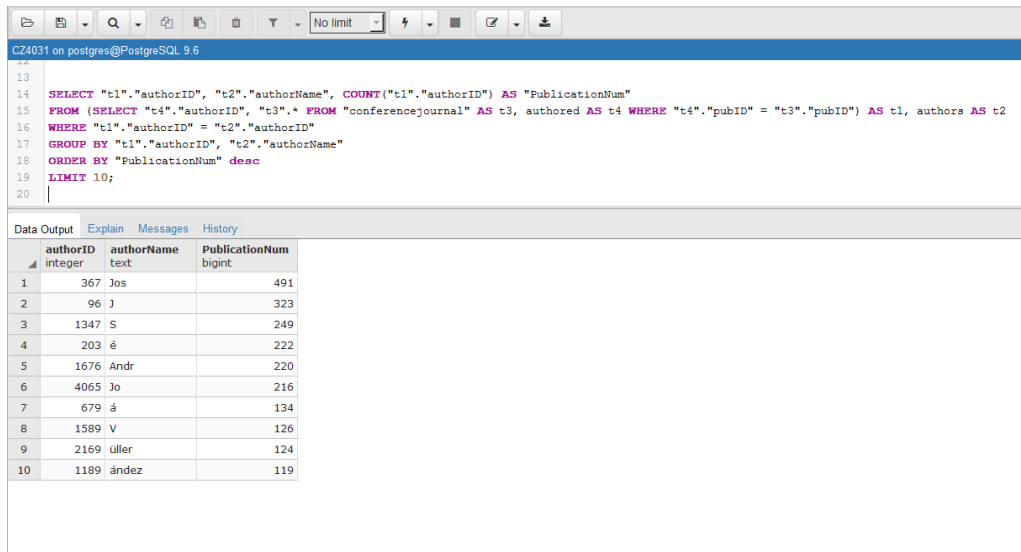
After which, we select “conferences” and “journals” category and create a separate view.

```
CREATE VIEW conferencejournal AS(
SELECT * FROM ((SELECT * FROM "data5y" WHERE "pubKey" LIKE '%conf/%')
UNION
(SELECT * FROM "data5y" WHERE "pubKey" LIKE '%journals/%')) AS TEMP);
```

Finally, we count the numbers of each publication from each author, sort by DESC, and limit to the top 10 largest number.

```
SELECT "t1"."authorID", "t2"."authorName", COUNT("t1"."authorID") AS "PublicationNum"
FROM (SELECT "t4"."authorID", "t3".* FROM "conferencejournal" AS t3, authored AS t4 WHERE "t4"."pubID" = "t3"."pubID") AS t1,
authors AS t2
WHERE "t1"."authorID" = "t2"."authorID"
GROUP BY "t1"."authorID", "t2"."authorName"
ORDER BY "PublicationNum" desc
LIMIT 10;
```

Final output, limit to top 10 most publications.



The screenshot shows a PostgreSQL query editor window titled "CZ4031 on postgres@PostgreSQL 9.6". The query is as follows:

```

13
14 SELECT "t1"."authorID", "t2"."authorName", COUNT("t1"."authorID") AS "PublicationNum"
15 FROM (SELECT "t4"."authorID", "t3".* FROM "conferencejournal" AS t3, authored AS t4 WHERE "t4"."pubID" = "t3"."pubID") AS t1, authors AS t2
16 WHERE "t1"."authorID" = "t2"."authorID"
17 GROUP BY "t1"."authorID", "t2"."authorName"
18 ORDER BY "PublicationNum" desc
19 LIMIT 10;
20

```

The results are displayed in a table with the following columns: authorID (integer), authorName (text), and PublicationNum (bigint). The table contains 10 rows of data, sorted by PublicationNum in descending order.

	authorID integer	authorName text	PublicationNum bigint
1	367	Jos	491
2	96	J	323
3	1347	S	249
4	203	é	222
5	1676	Andr	220
6	4065	Jo	216
7	679	á	134
8	1589	V	126
9	2169	üller	124
10	1189	ández	119

## 2.1.8 Question 8

List the name of the conferences such that it has ever been held in June, and the corresponding proceedings (in the year where the conf was held in June) contain more than 100 publications.

According to the dblp website, proceedings and inproceedings are classified under conference, hence we will be selecting data from proceedings and inproceedings tables. We will also be using attribute month obtained from “mdate”.

Solution: We will solve this question by splitting the query into smaller steps.

First, we select conferences (proceedings and inproceedings) that are held in ‘JUNE’.

--For Proceedings--

```

CREATE VIEW p_conf_names_in_june AS (
SELECT "publications"."title","publications"."year","proceedings"."proceedingsID",booktitle,"proceedings"."pubID" FROM proceedings
INNER JOIN publications ON "proceedings"."proceedingsID" = "publications"."pubID" WHERE "publications"."month" = 'June'
)

```

--For Inproceedings--

```

CREATE VIEW ip_conf_names_in_june AS (
SELECT "publications"."title","publications"."year","inproceedings"."inproceedingsID",booktitle,"inproceedings"."pubID" FROM inproceedings
INNER JOIN publications ON "inproceedings"."inproceedingsID" = "publications"."pubID" WHERE "publications"."month" = 'June'
)

```

Next, we group by years to see how many publications was published in a conference in June.

```
CREATE VIEW sub_p_conf_names_abv_100_publications_in_june AS (
SELECT "publications"."title", "publications"."year", "proceedings"."proceedingsID", booktitle, "proceedings"."pubID",
COUNT("publications"."pubID") AS "NumOfPublications" FROM proceedings
INNER JOIN publications ON "proceedings"."proceedingsID" = "publications"."pubID" WHERE "publications"."month" = 'June'
GROUP BY "publications"."title", "publications"."year", "proceedings"."proceedingsID", booktitle, "proceedings"."pubID"
)
```

Next, we select rows where “NUMOFPUBLICATIONS”> 100 for just the corresponding proceedings (in the year where the conf was held in June) as stated in the question from the proceedings table.

```
CREATE VIEW p_conf_names_abv_100_publications_in_june AS (
SELECT * FROM sub_p_conf_names_abv_100_publications_in_june
WHERE "sub_p_conf_names_abv_100_publications_in_june"."NumOfPublications" > '100'
)
```

Finally, we UNION the views of the conferences that are held in “June” and “the corresponding proceedings (in the year where the conference was held in June) where “NUMOFPUBLICATIONS”> 100 as stated in the question from the proceedings table.

```
SELECT "p_conf_names_in_june"."booktitle" AS "CONFERENCE_NAME", "p_conf_names_in_june"."year" FROM p_conf_names_in_june
UNION ALL
SELECT "ip_conf_names_in_june"."booktitle" AS "CONFERENCE_NAME", "ip_conf_names_in_june"."year" FROM ip_conf_names_in_june
UNION ALL
SELECT "p_conf_names_abv_100_publications_in_june"."booktitle" AS "CONFERENCE_NAME",
"p_conf_names_abv_100_publications_in_june"."year"
FROM p_conf_names_abv_100_publications_in_june
```

Sample Resulting View (Actual result has 479194 rows):

	CONFERENCE_NAME text	year integer
1	Trans. Pattern Langua...	2002
2	Trans. Rough Sets	2000
3	Trans. Rough Sets	2000
4	Trans. Rough Sets	2002
5	Trans. Computational ...	2000
6	Trans. Computational ...	2012
7	DCM	2006
8	PxTP@CADE	1988
9	FICS	1998
10	PDMC	2001
11	TERMGRAPH	1998
12	ACAC	1997
13	TARK	2017

## 2.1.9 Question 9

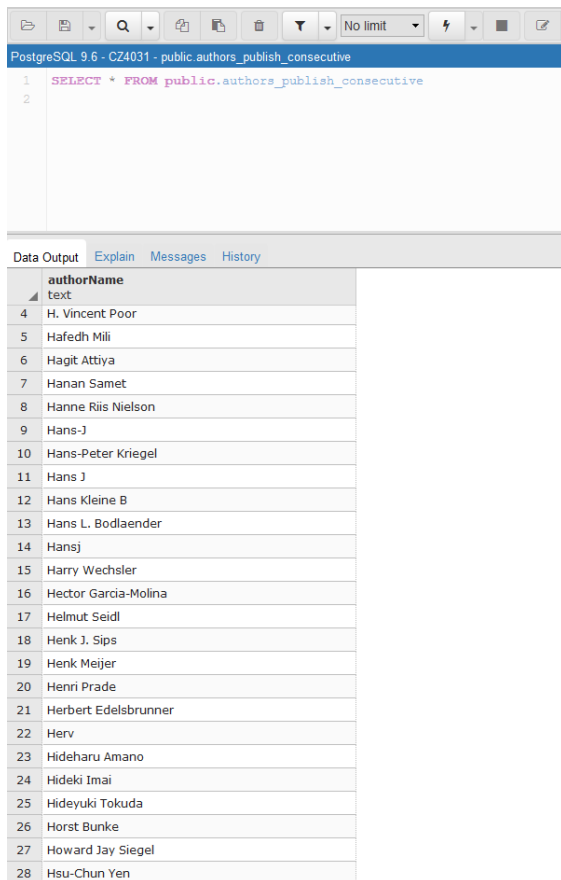
### 2.1.9.1 Question 9a

**Find authors who have published at least 1 paper every year in the last 30 years, and whose family name start with 'H'. (b) Find the names and number of publications for authors who have the earliest publication record in DBLP.**

```
CREATE VIEW authors_publish_consecutive AS (
SELECT "authors_authored_publications"."authorName"
FROM authors_authored_publications
WHERE "authors_authored_publications"."authorName" ILIKE 'H%'AND ("authors_authored_publications"."year" between '1987' AND '2017')
group by "authors_authored_publications"."authorName"
having count(distinct "authors_authored_publications"."year" ) = 2017-1987+1
)
```

```
SELECT * FROM authors_publish_consecutive
```

Final result of 28 rows:



PostgreSQL 9.6 - CZ4031 - public.authors\_publish\_consecutive

```
1 SELECT * FROM public.authors_publish_consecutive
2
```

authorName
4 H. Vincent Poor
5 Hafedh Mili
6 Hagit Attiya
7 Hanan Samet
8 Hanne Riis Nielson
9 Hans-J
10 Hans-Peter Kriegel
11 Hans J
12 Hans Kleine B
13 Hans L. Bodlaender
14 Hansj
15 Harry Wechsler
16 Hector Garcia-Molina
17 Helmut Seidl
18 Henk J. Sips
19 Henk Meijer
20 Henri Prade
21 Herbert Edelsbrunner
22 Herv
23 Hideharu Amano
24 Hideki Imai
25 Hideyuki Tokuda
26 Horst Bunke
27 Howard Jay Siegel
28 Hsu-Chun Yen



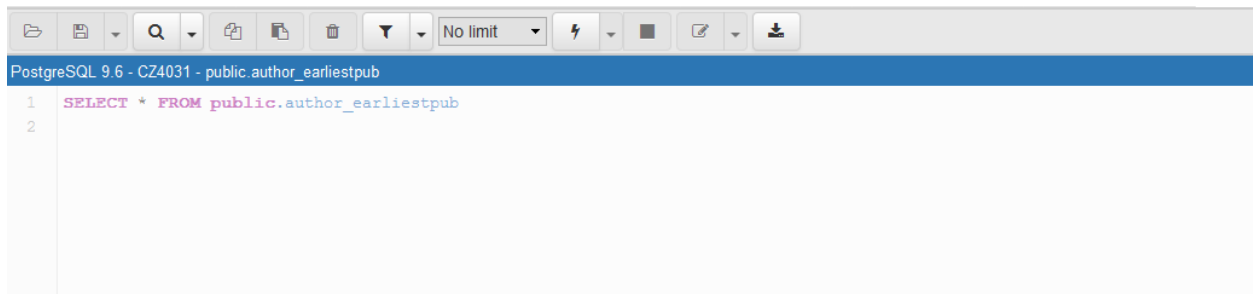
### 2.1.9.2 Question 9b

**Find the names and number of publications for authors who have the earliest publication record in DBLP.**

**Based on the information we have, we use “years” attribute to determine the earliest publication record. Our findings indicate that the earliest records are published in the year 1936.**

```
CREATE VIEW earliestPubs AS (
SELECT "publications"."pubID" from publications where year='1936'
)
```

```
CREATE VIEW author_earliestPub AS (
SELECT "authors_authored"."authorID","authors_authored"."authorName","authors_authored"."pubID" FROM authors_authored
INNER JOIN earliestPubs ON "authors_authored"."pubID" = "earliestPubs"."pubID"
)
```



PostgreSQL 9.6 - CZ4031 - public.author\_earliestpub

```
1 SELECT * FROM public.author_earliestpub
2
```

	authorID integer	authorName text	title text	year text	pubID integer
1	738315	Arnold F. Emch	Implication and Deducibility.	1936	782149
2	738339	Frederic Brenton Fitch	A System of Formal Logic Without an Analogue to the...	1936	782443
3	738265	W. V. Quine	Set-Theoretic Foundations for Logic.	1936	782567
4	738265	W. V. Quine	Toward a Calculus of Concepts.	1936	783195
5	738422	C. J. Ducasse	List of Officers and Members of the Association for Sy...	1936	783237
6	402275	J. Barkley Rosser	ödel and Church.	1936	783282
7	738617	C. I. Lewis	Emchs Calculus and Strict Implication.	1936	783745
8	738446	Alonzo Church	Correction to	1936	783938
9	738446	Alonzo Church	A Note on the Entscheidungsproblem.	1936	783978
10	738425	Emil L. Post	Finite Combinatory Processes-Formulation 1.	1936	784053
11	738315	Arnold F. Emch	Addendum to the Article	1936	784281
12	402275	J. Barkley Rosser	Constructibility as a Criterion for Existence.	1936	786229

### 2.1.10 Question 10

**Design a join query that is not in the above list.**

**We will use this query “Find all authors who publishes both articles and books Subcategory”.**

First, we need to use the view “authors\_authored\_publications” and using pubID to join on “books” table to find with authors publish at least a book.

```
CREATE VIEW book_authors AS (
SELECT "authors_authored_publications"."authorID", "authors_authored_publications"."authorName", "authors_authored_publications"."pubID"
FROM authors_authored_publications
INNER JOIN books ON "authors_authored_publications"."pubID" = "books"."pubID"
)
```

We then use the same view to join on “articles” table to find which authors publish a least an article.

```
CREATE VIEW articles_authored AS (
SELECT "authors_authored_publications"."authorID", "authors_authored_publications"."authorName"
FROM authors_authored_publications
INNER JOIN books ON "authors_authored_publications"."pubID" = "books"."pubID"
GROUP BY "authors_authored_publications"."authorID", "authors_authored_publications"."authorName"
)
```

Finally, we use INNER JOIN on both VIEWS to find COMMON authors who publish at least an article and at least a book before.

```
SELECT "articles_authored"."authorName", "articles_authored"."authorID" FROM
articles_authored
INNER JOIN book_authors ON "articles_authored"."authorName" = "book_authors"."authorName"
GROUP BY "articles_authored"."authorName", "articles_authored"."authorID"
```

## CZ4031 Assignment 1 Report: Querying Databases Efficiently

Sample data of 25 (Actual result has 78103 rows):

Data Output	authorName text	authorID integer
1	-Oliver	2000180
2	A	11945
3	á	679
4	à	1935
5	ä	650
6	ã	166966
7	ä-Aro	788487
8	à-Labastie	924690
9	á-Maestre	35866
10	ã Ara	375527
11	a Bernardini	2000177
12	á Brand	1979939
13	á Caetano	511938
14	á Cristina Vascon...	421060
15	á de Souza	417399
16	ä Geeraert	1962632
17	á Guzm	1983083
18	à Hachana	749257
19	á Leite	1983209
20	á Ramalho	1997797
21	á Silva	62754
22	á Vay	317829
23	á Volot	1551211
24	A.-A. M. Khamiss	1994900
25	A.-N. S. Abo-Rezeq	1994100

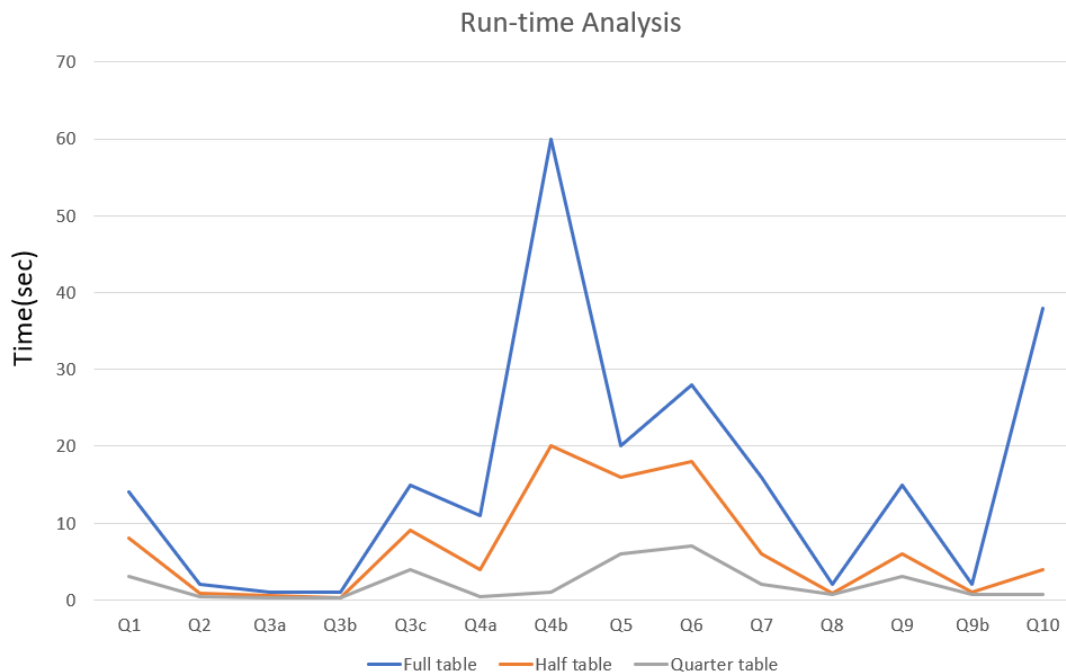
## 2.2 Queries Analysis

### Queries Analysis

	Full Table	Half Table	Quarter Table
Q1	14 secs	8 secs	3 secs
Q2	2 secs	874msec	411msec
Q3a	1 sec	512msec	283msec
Q3b	993msec	307msec	210msec (No result)
Q3c	15 secs	9 secs	4secs
Q4a	11 secs	4 secs	461msec (No result)
Q4b	60 secs	20 secs	1secs (No result)
Q5	20 secs	16 secs	6secs
Q6	28 secs	13 secs	7 secs
Q7	16 secs	6secs	2 secs
Q8	2 secs	891msec	747msec
Q9a	15 secs	6 secs	3secs
Q9b	2 secs	1 sec	769msec
Q10	38 secs	4 secs	690msec

All results were taken from the average of 3 runs.

Some of the query produce no result because data was removed when halving the tables



We recorded the run time for each query after it produced the result. We half the database/tables using DELETE FROM statements. From the table and chart, we concluded that halving the database does not necessary reduce the runtime by exactly half. For most cases, it does reduce the runtime between 40-60%. On rare cases, halving the data would reduce the runtime by 80%. This could be the way the DBMS run its query.

To summarize the results, we say that the runtime for the queries will be faster with lesser data.

### 3 Build Index and Study the Effect of Index

#### 3.1 CREATE INDEX Statements

PostgreSQL automatically generate indexes for all primary keys. These indexes are **articles(articleID)**, **authored(authorID, pubID)**, **authors(authorsID)**, **books(bookID)**, **incollections(incollectionsID)**, **inproceedings(inproceedingID)**, **proceedings(proceedingID)** and **publications(pubID)**.

##### Automatically created

```
CREATE INDEX articlesIndex ON articles("articleID")
CREATE INDEX authoredIndex_pubID ON authored("pubID")
CREATE INDEX authoredIndex_authorID ON authored("authorID")
CREATE INDEX authorsIndex_authorID ON authors("authorID")
CREATE INDEX booksIndex_bookID ON books("bookID")
CREATE INDEX incollections_ID ON incollections("incollectionsID")
CREATE INDEX inproceedings_iid ON inproceedings("inproceedingsID")
CREATE INDEX proceedings_iid ON proceedings("proceedingsID")
CREATE INDEX publications_pubID ON publications("pubID")
```

---

Besides the index that were created by PostgreSQL, we can also manually create indexes to speed up some of our queries. By default, PostgreSQL uses B-Tree, which will help in locating record faster in the database. The following are indexes that we created, mainly on attributes that are used by “GROUP BY”, “ORDER BY”, equality, and range queries. Some other reasons why we selectively select attributes to build index were also because indexes take up extra memory on disk, and it may be expensive to build attributes that are not used in such queries.

```
CREATE INDEX articlesIndex_pid ON articles("pubID")
CREATE INDEX authorsIndex_authorName ON authors("authorName")
CREATE INDEX booksIndex_pubID ON books("pubID")
CREATE INDEX incollections_pubID ON incollections("pubID")
CREATE INDEX incollections_booktitle ON incollections("booktitle")
CREATE INDEX inproceedings_pubID ON inproceedings("pubID")
CREATE INDEX inproceedings_booktitle ON inproceedings("booktitle")
```

## CZ4031 Assignment 1 Report: Querying Databases Efficiently

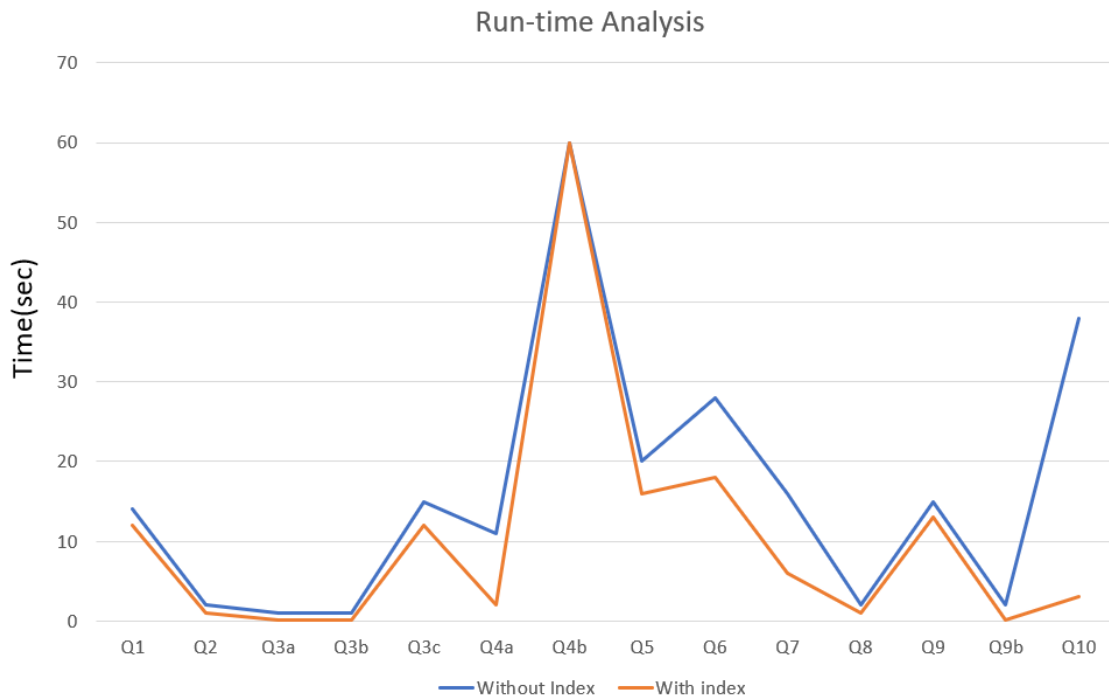
```
CREATE INDEX proceedings_pubID ON proceedings("pubID")
CREATE INDEX proceedings_booktitle ON proceedings("booktitle")
CREATE INDEX publications_pubKey ON publications("pubKey")
CREATE INDEX publications_title ON publications("title")
CREATE INDEX publications_year ON publications("year")
CREATE INDEX publications_month ON publications("month")
```

For this assignment, we mainly use single-column index. Even after using index, the timing of Q6 still exceeds 30min, hence we note the timing as “undefined”. This is due to fact that the it is a cross JOIN on 1.4million tuples on 1.4million tuples, hence resulting in an enormous dataset.

Our studies show that most of the queries were sped up by the indexes that we created. To learn more about the effects of index, 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. We also note that from the query plans, DBMS may not make use of the indexes that we have create, thus we do not have control over when the indexes are used.

	Full Table without Index	Full Table with Index
Q1	16 secs	15 secs
Q2	2 secs	1 sec
Q3a	1 sec	90msec
Q3b	993msec	83msec
Q3c	15 secs	12 secs
Q4a	11 secs	2 secs
Q4b	60 secs	60 secs
Q5	20 secs	16 secs
Q6	28 secs	18 secs
Q7	16 secs	6 secs
Q8	2 secs	1 sec
Q9a	15 secs	13 secs
Q9b	2 secs	80msec
Q10	38 secs	1 sec

\*Highlighted rows are queries that timing are improved drastically by the index.



To sum up our findings in general, index speeds up query time dramatically even up to 90%(Q3b, Q10). Also, not all indexes that were created were used. Even using index, some queries may not be speed up(Q4b).

### 3.2 In-depth look at how Index are used on each query

Usage of index seen in Query Plan

Q1 –

Index Scan using `publications_pubid` on publications

Index Cond comparing publications\_pubID with proceedings.pubID

Index Cond comparing publications\_pubID with books.pubID

Not much noticeable speed up as index.

Q2 –

Index Scan using [proceedings\\_iid](#) on proceedings

Index Scan using [publications\\_pubid](#) on publications

Index Cond comparing month='July'

Runtime is improved as equality statements are being used by the query

Q3a –

Index Scan using [authorsindex\\_authurname](#) on authors

Index Scan using [publications\\_pubid](#) on publications

Index Scan using [articlesindex\\_pid](#) on articles

Index Scan using [booksindex\\_pubid](#) on books

Index Scan using [incollections\\_pubid](#) on incollections

Index Scan using [inproceedings\\_pubid](#) on inproceedings

Index Scan using [proceedings\\_pubid](#) on proceedings

Index Cond comparing "pubID" = authored."pubID"

Index Cond comparing "authorName" = 'Xiaokui Shu'

Index Cond comparing "authorID" = authors."authorID"

Index Cond comparing authored."pubID" = "pubID"

Runtime is improved as equality and range statements are being used by the query

Q3b –

Index Scan using [authorsindex\\_authurname](#) on authors

Index Scan using [authoredindex\\_authorid](#) on authored

Index Scan using [inproceedings\\_pubid](#) on inproceedings

Index Scan using [publications\\_pubid](#) on publications

Index Cond comparing "authorID" = authors."authorID"

Index Cond comparing "authorName" = 'Peter Nobel'

Index Cond comparing "pubID" = authored."pubID"

Index Cond comparing "pubID" = authored."pubID"

Runtime is improved as equality statements are being used by the query



Q3c –

Index Cond comparing year = '2015'::text

Runtime is improved as “GROUP BY” statements are being used by the query

Q4a –

Index Scan using [authoredindex\\_pubid](#) on authored

Index Scan using [authorsindex\\_authorid](#) on authors authors

Index Cond comparing "pubID" = publications."pubID"

Index Cond comparing "authorID" = authored."authorID"

Index Cond comparing "authorName" = authors."authorName"

Runtime is improved as “GROUP BY” and equality statements are being used by the query

Q4b –

Index Scan using [authoredindex\\_pubid](#) on authored

Index Scan using [authorsindex\\_authorid](#) on authors

Index Scan using [authoredindex\\_pubid](#) on authored authored

Index Scan using [authorsindex\\_authorid](#) on authors authors

Index Cond comparing "pubID" = publications."pubID"

Index Cond comparing "authorID" = authored."authorID"

Not much noticeable speed up

Q5 –

Bitmap Index Scan on [publications\\_year](#)

Bitmap Index Scan on [publications\\_year](#)

Index Cond comparing the ranges (year >= '1970'::text) AND (year <= '1979'::text)

Index Cond comparing the ranges (year >= '1980'::text) AND (year <= '1989'::text)

Index Cond comparing the ranges (year >= '1990'::text) AND (year <= '1999'::text)

Index Cond comparing the ranges (year >= '2000'::text) AND (year <= '2009'::text)

Runtime is improved as “ORDER BY” and range statements are being used by the query

Q6 –

Index Scan using `authoredindex_pubid` on authored

Index Only Scan using `authorsindex_authorid` on authors

Index Only Scan using `publications_pubid` on publications

Index Cond comparing "authorID" = authored."authorID"

Index Cond comparing "pubID" = proceedings."pubID"

Runtime is improved as “GROUP BY”, “ORDER BY” and equality statements are being used by the query

Q7-

Bitmap Index Scan on `publications_year`

Index Scan using `authoredindex_pubid` on authored

Index Cond comparing the ranges ((year >= '2013'::text) AND (year <= '2017'::text))

Index Cond comparing "pubID" = publications."pubID"

Runtime is improved by a large margin as “GROUP BY”, “ORDER BY”, range and equality statements are being used by the query

Q8 –

Index Scan using `proceedings_iid` on proceedings

Index Scan using `publications_pubid` on publications

Runtime is improved as “GROUP BY” and range statements are being used by the query

INNER JOIN publications ON "proceedings"."proceedingsID" = "publications"."pubID"  
WHERE "publications"."month" = 'June'

GROUP BY

"publications"."title", "publications"."year", "proceedings"."proceedingsID", booktitle, "proceedings"."pubID"

Q9a –

Index Scan using publications\_pkey on publications

Index Cond comparing "pubID" = authored."pubID"

Runtime is improved as “GROUP BY” and range statements are being used by the query

WHERE "authors\_authored\_publications"."authorName" ILIKE 'H%' AND  
("authors\_authored\_publications"."year" between '1987' AND '2017')

GROUP BY "authors\_authored\_publications"."authorName"

Q9b -

Bitmap Index Scan on publications\_year

Index Scan using authoredindex\_pubid on authored

Index Scan using authorsindex\_authoid on authors

Index Cond comparing year = '1936'::text

Index Cond comparing "pubID" = publications."pubID"

Index Cond comparing "authorID" = authored."authorID"

Runtime is improved as equality statement is being used by the query.( SELECT  
"publications"."pubID" from publications where year='1936')

Q10 –

Index Scan using authorsindex\_authoid on authors authors

Index Scan using authoredindex\_pubid on authored

Index Only Scan using publications\_pubid on publications

Index Only Scan using publications\_pubid on publications publications

Index Cond comparing "pubID" = books."pubID"

Index Cond comparing "pubID" = publications."pubID"

Runtime is improved as “GROUP BY” is being used by the query.

Based on the query plans, we have seen the used of indexes that we created, as well as the indexes that were automatically created by PostgreSQL. The use of these indexes in the query plan tally with the result that were produced from the queries after the indexes were created.

## 4 Advanced Part: Study the Effect of Cache

Caching is an important aspect to tuning a database's performance. A database system is mostly concern about input/output(I/O). PostgreSQL caches data in RAM which can greatly improve performance. The configuration parameters for PostgreSQL's caching is called the shared buffer size and its default size is 128MB in our system. The shared buffer is an array of 8KB blocks. Before Postgres checks out the disk for data, it will first access the pages in the shared buffer, and if there is a hit, it will not perform an I/O on the disk.

In our study, we try to vary the different values of shared buffer to see if there is any change in result for our queries runtime. Postgres can cache table data, indexes, as well as query plan.

Since our RAM is 8GB, we will try to increase our shared\_buffer size up to 2GB(2048MB), 1/4<sup>th</sup> of max RAM.

Indexed results:

	64MB	128MB	256MB	512MB	2048MB
Q1	13 secs	12 secs	11 secs	11secs	12secs
Q2	1sec	1 sec	1 sec	1 sec	1 sec
Q3a	86msec	90msec	89msec	75msec	84msec
Q3b	79msec	83msec	80msec	80msec	79msec
Q3c	12 secs	12 secs	11 secs	11 secs	10 secs
Q4a	2 secs	2 secs	2 secs	1 secs	1 secs
Q4b	60 secs	60 secs	60 secs	60 secs	60sec
Q5	21 secs	16 secs	16 secs	17 secs	15 secs
Q6	28 secs	28 secs	28 secs	28 secs	28 secs
Q7	6secs	6 secs	5 secs	5secs	5secs
Q8	1 sec	1 sec	1sec	1sec	1sec
Q9a	17 secs	13 secs	12 secs	13secs	13secs
Q9b	86msec	80msec	71msecs	87msec	80msec
Q10	1sec	1 sec	1 secs	1secs	1 secs

As seen from the above result, the runtime of the queries runtime was not affected by the change in shared buffer. Some of the potential reason why we feel that it does not change is due to the "work\_mem" parameter in the configuration. It may also be because of the hit and miss rate due to how the queries are affected by the query plan.

-----End of Report -----