

## **PROJECT 2: WHAT IS YOUR QUERY ACCESSING AND AT WHAT COST?**

### **SC3020 DATABASE SYSTEM PRINCIPLES**

**TOTAL MARKS: 100**

**Due Date: November 17, 2023; 11:59 PM**

The relational query execution engine implements a set of physical operators. An operator takes as input one or more data streams and produces an output data stream. Some examples of physical operators are sequential scan, index scan, and hash join. These physical operators are the building blocks for the execution of SQL queries. An abstract representation of such an execution is a physical operator tree (operator tree for brevity), denoted as  $T = (N, E)$ , where  $N$  is a set of nodes representing the operators and  $E$  is a set of edges representing data flow among the physical operators. The physical operator tree is the abstract representation of a **query execution plan** (QEP). The query execution engine is responsible for the execution of the QEP to generate results of a SQL query. You can view a QEP of a given query using the *EXPLAIN* feature of *PostgreSQL*.

It is often unclear to a user on what is happening “inside the box” of a database system when one poses an SQL query. For instance, which blocks/pages are being accessed by the query? Which tuples are there in these blocks? How much buffer is used? The goal of your project is to shed light to various questions associated with a QEP by **designing and implementing a software that facilitates visual exploration of disk blocks accessed by a given input SQL query as well as various features of the corresponding QEP**. Specifically, the input/output of your program is as follows:

- **Input:** An SQL query
- **Output:** (a) Visualization of disk blocks accessed by the query. It should also facilitate exploration of the content of these blocks interactively. (b) Visualizing different aspects of the QEP including buffer size, cost, etc.

**Hint:** See Question 4 of Example Class 7.

Specific tasks required for your project are as follows:

- Design and implement an algorithm that takes as input an SQL query and creates the above visual exploration framework. Your goal is to ensure generality of the solution (i.e., it can handle a wide variety of query plans on different database instances) as well as easy-to-use visual exploration. The better is the algorithm design for the task, the more credit you will receive.
- A user-friendly, graphical user interface (GUI) to enable the visual exploration goals. Note that a working GUI to show your results easily should suffice. It is not necessary to generate a fancy one as the course is not on GUI development.

You should use **Python** as the host language on **Windows** platform for your project. For students using **Mac** platform, you can **install Windows on your Mac** by following instructions in <https://support.apple.com/en-sg/HT201468>. The DBMS allowed in this project is **PostgreSQL**. The example dataset you should use for this project is **TPC-H** (see *Appendix*). You are free to use any off-the-shelf toolkits for your project.

Note that several parts of the project are left open-ended (e.g., how the GUI should look like? How should you visualize the disk blocks? What features of a QEP needs to be shown and how they should be explained?) intentionally so that the project does not curb a group's creative endeavors. You are free to make realistic assumptions to achieve these tasks.

## **SUBMISSION REQUIREMENTS**

Your submission should include the followings:

- You should submit **three** program files: *interface.py*, *explore.py*, and *project.py*. The file *interface.py* contains the code for the GUI. The *explore.py* contains the code for facilitating the exploration. The *project.py* is the main file that invokes all the necessary procedures. **Note that we shall be running the project.py file** (either from command prompt or using the Pycharm IDE) to execute the software. Make sure your code follows good coding practice: sufficient comments, proper variable/function naming, etc. We will execute the software to check its correctness using **different** query sets and dataset to check for the generality of the solution. We will also check quality of algorithm design.
- **Softcopy report** containing details of the software including formal descriptions of the key algorithms with examples. You should also discuss limitations of the software (if any).
- **Peer assessment report** from each member of the team. Each individual member of a team needs to assess contributions of the group members. Details of peer assessment form will be provided closer to the submission date.
- If you are using any **Large Language Model** (e.g., ChatGPT) for your project then you must acknowledge its usage appropriately and clearly specify where you have used it (in the report). Failure to do so may result in an F grade for the project.
- You must submit a document containing instructions to run your software successfully. You will not receive any credit if your software fails to execute based on your instructions.
- All submissions will be through NTU Learn. Details of submissions will be released nearer to the submission date.

*Note: Late submission will be penalized (10 marks per day).*

## **Appendix**

### **I. Creating TPC-H database in PostgreSQL**

Follow the following steps to generate the TPC-H data (this step may differ slightly due to different versions of TPC-H). Please modify the steps as deemed necessary. Also, build appropriate indexes.

- 1) Go to [http://www.tpc.org/tpc\\_documents\\_current\\_versions/current\\_specifications5.asp](http://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp) and download TPC-H Tools v2.18.0.zip. Note that the version may defer as the tool may have been updated by the developer.
- 2) Unzip the package. You will find a folder "dbgen" in it.
- 3) To generate an instance of the TPC-H database:
  - Open up tpch.vcproj using visual studio software.
  - Build the tpch project. When the build is successful, a command prompt will appear with "TPC-H Population Generator <Version 2.17.3>" and several \*.tbl files will be generated. You should expect the following .tbl files: customer.tbl, lineitem.tbl, nation.tbl, orders.tbl, part.tbl, partsupp.tbl, region.tbl, supplier.tbl
  - Save these .tbl files as .csv files
  - These .csv files contain an extra " | " character at the end of each line. These " | " characters are incompatible with the format that PostgreSQL is expecting. Write a small piece of code to remove the last " | " character in each line. Now you are ready to load the .csv files into PostgreSQL
  - Open up PostgreSQL. Add a new database "TPC-H".
  - Create new tables for "customer", "lineitem", "nation", "orders", "part", "partsupp", "region" and "supplier"
  - Import the relevant .csv into each table. Note that pgAdmin4 for PostgreSQL (windows version) allows you to perform import easily. You can select to view the first 100 rows to check if the import has been done correctly. If encountered error (e.g., ERROR: extra data after last expected column) while importing, create columns of each table first before importing. Note that the types of each column has to be set appropriately. You may use the SQL commands in Appendix II to create the tables.

Alternatively, you can also refer to <https://docs.verdictdb.org/tutorial/tpch/> for additional help on creating the TPC-H database



## II. SQL commands for creating TPC-H data tables

### Region table

```
5 CREATE TABLE public.region
6 (
7     r_regionkey integer NOT NULL,
8     r_name character(25) COLLATE pg_catalog."default" NOT NULL,
9     r_comment character varying(152) COLLATE pg_catalog."default",
10    CONSTRAINT region_pkey PRIMARY KEY (r_regionkey)
11 )
12 WITH (
13     OIDS = FALSE
14 )
15 TABLESPACE pg_default;
16
17 ALTER TABLE public.region
18     OWNER to postgres;
```

### 1) Nation table

```
5 CREATE TABLE public.nation
6 (
7     n_nationkey integer NOT NULL,
8     n_name character(25) COLLATE pg_catalog."default" NOT NULL,
9     n_regionkey integer NOT NULL,
10    n_comment character varying(152) COLLATE pg_catalog."default",
11    CONSTRAINT nation_pkey PRIMARY KEY (n_nationkey),
12    CONSTRAINT fk_nation FOREIGN KEY (n_regionkey)
13        REFERENCES public.region (r_regionkey) MATCH SIMPLE
14        ON UPDATE NO ACTION
15        ON DELETE NO ACTION
16 )
17 WITH (
18     OIDS = FALSE
19 )
20 TABLESPACE pg_default;
21
22 ALTER TABLE public.nation
23     OWNER to postgres;
```

## 2) Part table

```
5 CREATE TABLE public.part
6 (
7     p_partkey integer NOT NULL,
8     p_name character varying(55) COLLATE pg_catalog."default" NOT NULL,
9     p_mfgr character(25) COLLATE pg_catalog."default" NOT NULL,
10    p_brand character(10) COLLATE pg_catalog."default" NOT NULL,
11    p_type character varying(25) COLLATE pg_catalog."default" NOT NULL,
12    p_size integer NOT NULL,
13    p_container character(10) COLLATE pg_catalog."default" NOT NULL,
14    p_retailprice numeric(15,2) NOT NULL,
15    p_comment character varying(23) COLLATE pg_catalog."default" NOT NULL,
16    CONSTRAINT part_pkey PRIMARY KEY (p_partkey)
17 )
18 WITH (
19     OIDS = FALSE
20 )
21 TABLESPACE pg_default;
22
23 ALTER TABLE public.part
24     OWNER to postgres;
```

## 3) Supplier table

```
5 CREATE TABLE public.supplier
6 (
7     s_suppkey integer NOT NULL,
8     s_name character(25) COLLATE pg_catalog."default" NOT NULL,
9     s_address character varying(40) COLLATE pg_catalog."default" NOT NULL,
10    s_nationkey integer NOT NULL,
11    s_phone character(15) COLLATE pg_catalog."default" NOT NULL,
12    s_acctbal numeric(15,2) NOT NULL,
13    s_comment character varying(101) COLLATE pg_catalog."default" NOT NULL,
14    CONSTRAINT supplier_pkey PRIMARY KEY (s_suppkey),
15    CONSTRAINT fk_supplier FOREIGN KEY (s_nationkey)
16        REFERENCES public.nation (n_nationkey) MATCH SIMPLE
17        ON UPDATE NO ACTION
18        ON DELETE NO ACTION
19 )
20 WITH (
21     OIDS = FALSE
22 )
23 TABLESPACE pg_default;
24
25 ALTER TABLE public.supplier
26     OWNER to postgres;
```

#### 4) Partsupp table

```
5 CREATE TABLE public.partsupp
6 (
7     ps_partkey integer NOT NULL,
8     ps_suppkey integer NOT NULL,
9     ps_availqty integer NOT NULL,
10    ps_supplycost numeric(15,2) NOT NULL,
11    ps_comment character varying(199) COLLATE pg_catalog."default" NOT NULL,
12    CONSTRAINT partsupp_pkey PRIMARY KEY (ps_partkey, ps_suppkey),
13    CONSTRAINT fk_ps_suppkey_partkey FOREIGN KEY (ps_partkey)
14        REFERENCES public.part (p_partkey) MATCH SIMPLE
15        ON UPDATE NO ACTION
16        ON DELETE NO ACTION,
17    CONSTRAINT fk_ps_suppkey_suppkey FOREIGN KEY (ps_suppkey)
18        REFERENCES public.supplier (s_suppkey) MATCH SIMPLE
19        ON UPDATE NO ACTION
20        ON DELETE NO ACTION
21 )
22 WITH (
23     OIDS = FALSE
24 )
25 TABLESPACE pg_default;
26
27 ALTER TABLE public.partsupp
28     OWNER to postgres;
```

#### 5) Customer table

```
5 CREATE TABLE public.customer
6 (
7     c_custkey integer NOT NULL,
8     c_name character varying(25) COLLATE pg_catalog."default" NOT NULL,
9     c_address character varying(40) COLLATE pg_catalog."default" NOT NULL,
10    c_nationkey integer NOT NULL,
11    c_phone character(15) COLLATE pg_catalog."default" NOT NULL,
12    c_acctbal numeric(15,2) NOT NULL,
13    c_mktsegment character(10) COLLATE pg_catalog."default" NOT NULL,
14    c_comment character varying(117) COLLATE pg_catalog."default" NOT NULL,
15    CONSTRAINT customer_pkey PRIMARY KEY (c_custkey),
16    CONSTRAINT fk_customer FOREIGN KEY (c_nationkey)
17        REFERENCES public.nation (n_nationkey) MATCH SIMPLE
18        ON UPDATE NO ACTION
19        ON DELETE NO ACTION
20 )
21 WITH (
22     OIDS = FALSE
23 )
24 TABLESPACE pg_default;
25
26 ALTER TABLE public.customer
27     OWNER to postgres;
```

## 6) Orders table

```
5 CREATE TABLE public.orders
6 (
7     o_orderkey integer NOT NULL,
8     o_custkey integer NOT NULL,
9     o_orderstatus character(1) COLLATE pg_catalog."default" NOT NULL,
10    o_totalprice numeric(15,2) NOT NULL,
11    o_orderdate date NOT NULL,
12    o_orderpriority character(15) COLLATE pg_catalog."default" NOT NULL,
13    o_clerk character(15) COLLATE pg_catalog."default" NOT NULL,
14    o_shippriority integer NOT NULL,
15    o_comment character varying(79) COLLATE pg_catalog."default" NOT NULL,
16    CONSTRAINT orders_pkey PRIMARY KEY (o_orderkey),
17    CONSTRAINT fk_orders FOREIGN KEY (o_custkey)
18        REFERENCES public.customer (c_custkey) MATCH SIMPLE
19        ON UPDATE NO ACTION
20        ON DELETE NO ACTION
21 )
22 WITH (
23     OIDS = FALSE
24 )
25 TABLESPACE pg_default;
26
27 ALTER TABLE public.orders
28     OWNER to postgres;
```



## 7) Lineitem table

```
5 CREATE TABLE public.lineitem
6 (
7     l_orderkey integer NOT NULL,
8     l_partkey integer NOT NULL,
9     l_suppkey integer NOT NULL,
10    l_linenumber integer NOT NULL,
11    l_quantity numeric(15,2) NOT NULL,
12    l_extendedprice numeric(15,2) NOT NULL,
13    l_discount numeric(15,2) NOT NULL,
14    l_tax numeric(15,2) NOT NULL,
15    l_returnflag character(1) COLLATE pg_catalog."default" NOT NULL,
16    l_linestatus character(1) COLLATE pg_catalog."default" NOT NULL,
17    l_shipdate date NOT NULL,
18    l_commitdate date NOT NULL,
19    l_receiptdate date NOT NULL,
20    l_shipinstruct character(25) COLLATE pg_catalog."default" NOT NULL,
21    l_shipmode character(10) COLLATE pg_catalog."default" NOT NULL,
22    l_comment character varying(44) COLLATE pg_catalog."default" NOT NULL,
23    CONSTRAINT lineitem_pkey PRIMARY KEY (l_orderkey, l_partkey, l_suppkey, l_linenumber),
24    CONSTRAINT fk_lineitem_orderkey FOREIGN KEY (l_orderkey)
25        REFERENCES public.orders (o_orderkey) MATCH SIMPLE
26        ON UPDATE NO ACTION
27        ON DELETE NO ACTION,
28    CONSTRAINT fk_lineitem_partkey FOREIGN KEY (l_partkey)
29        REFERENCES public.part (p_partkey) MATCH SIMPLE
30        ON UPDATE NO ACTION
31        ON DELETE NO ACTION,
32    CONSTRAINT fk_lineitem_suppkey FOREIGN KEY (l_suppkey)
33        REFERENCES public.supplier (s_suppkey) MATCH SIMPLE
34        ON UPDATE NO ACTION
35        ON DELETE NO ACTION
36 )
37 WITH (
38     OIDS = FALSE
39 )
40 TABLESPACE pg_default;
41
42 ALTER TABLE public.lineitem
43     OWNER to postgres;
```