ECE 356 Project Report

Group Number: 20

Group Members:

Shuangqi Wu (s352wu)

Kehan Xing (k5xing)

Jing Liu (j762liu)

Date: 2021/12/23

Table of Contents

1. Introduction

In this project, our goal is to design and implement a database client application for Internet Service Provider (ISP) network administrators to efficiently perform data query operations, such as retrieving, updating, adding, and deleting certain data by entering simple command-line inputs on a terminal interface. To achieve this object, we designed an Entity-Relationship Model, implemented a client application, add test cases covering both server and client sides, and exercised data mining practice. The dataset we used for our project is the Unicauca Network Flows Dataset, which is obtained from https://www.kaggle.com/jsrojas/labeled-network-traffic-flows-114-applications.

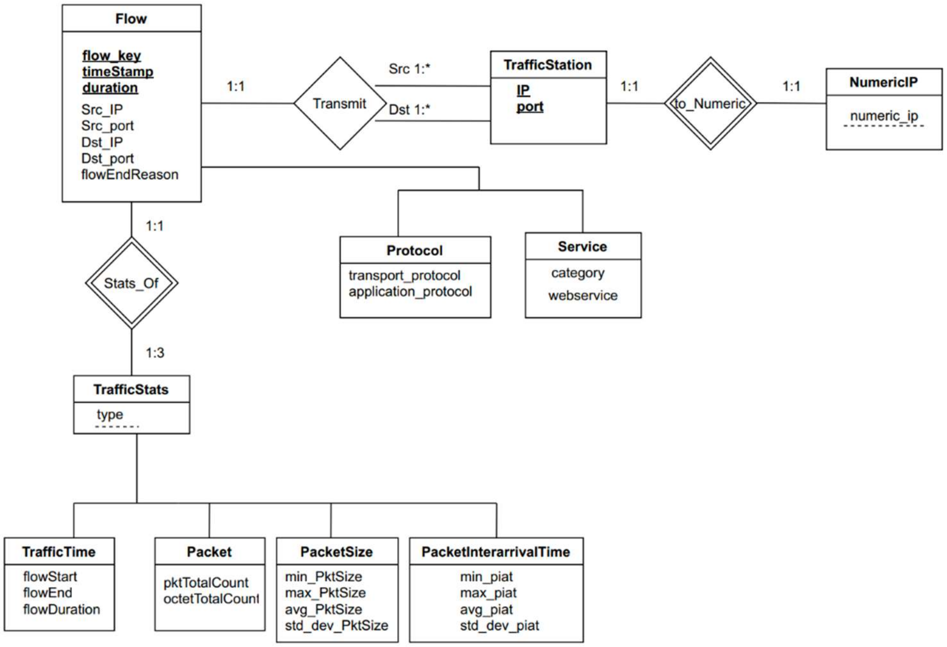
GitLab repository: https://git.uwaterloo.ca/s352wu/ece-356-project.git

2. ER Model Design

2.1 Overview

ER Model is very useful for structuring raw data into different entity sets and specifying the relationship between these entity sets. In the source dataset, there are 50 attributes with different data types and values, we classified these attributes into 10 entities (including strong and weak entities), add a few extra attributes to help identify an entity, and determined how the entities are related to each other.

Our ER Model for the project is shown below:



2.2 Explanation

- The entity **TrafficStation** consists of two attributes: **IP** and **port**, which are also the primary key. This entity stores all IP addresses and port numbers for all available internet traffic flows.
- The entity **NumericIP** is a weak entity related to the entity **TrafficStation** by the relation **to_Numeric**. The discriminator for this weak entity is the attribute **numeric_ip**. The purpose of this entity is to convert all IP addresses to decimal format.
- The entity **Flow** consists of eight attributes where **flow_key**, **timeStamp**, and **duration** act as the primary key to uniquely identify each internet traffic flow. The reason we decided to use the combination of these attributes as the primary key is that flow_key itself is non-unique among all flows. Moreover, each internet traffic flow must have an IP address and a port number for both source and destination hosts, and the reason why the flow is ended. However, these IP addresses, port numbers, and flow end reasons may appear many times for different flows. The attributes **Src_IP, Src_port, Dst_IP, Dst_port** are the foreign keys that reference the primary key in the entity **TrafficStation**, which is shown by the relation **Transmit**.
- The two specializations of the entity **Flow** are **Protocol** and **Service**. The entity **Protocol** consists of the information about the transport and application protocols of each internet traffic flow, and the entity **Service** consists of the information about the category of the communication and the web services of each internet traffic flow.
- The entity **TrafficStats** is a weak entity depending on the entity **Flow.** The discriminator for this weak entity is **type.** It is also a meta entity for the statistics data, which stores the type of an internet traffic flow. For each flow, it includes three types of statistic data (Both Direction, Forward, and Backward). The **Stats_Of** relation associates an internet traffic flow with one of the three types.
- The entities **TrafficTime**, **Packet**, **PacketSize**, and **PacketInterarrvialTime** are the specializations of **TrafficStats**. Each of them contains specific statistical information (e.g., flow time, packet number, packet size, and packet arrival time) related to an internet traffic flow.

3. Relational Schema

3.1 Overview

The next step is to create a relational schema for the ER Model presented above. The SQL files for creating the relation schema are ***load_data.sql*** and ***create_er.sql***. First, we run ***load_data.sql*** to load the data in all 50 columns from the source dataset CSV file into a table named **Internet_Traffic,** and then we use ***create_er.sql*** to create tables for each entity shown in the ER.

3.2 Explanation

The file ***create_er.sql*** has detailed information regarding how the tables are created, their primary and foreign keys, the domain of each attribute, etc. Some notable remarks are shown as follows:

- Each internet traffic flow is identified by the primary key attributes: flow_key, timeStamp, and duration. There are about 15 flows that have the same value for the primary key attribute. After checking the source dataset, it seems that these are the duplicated records and therefore they are ignored when inserting data into the table **Flow** and any other tables with foreign keys references to **Flow**.
- In addition to primary keys, the tables which have foreign key constraints can also use the foreign keys as indexes to improve the query executing performance.
- When creating the tables **TrafficStats**, **TrafficTime**, **Packet**, **PacketSize**, and **PacketInterarrivalTime**, each of which performs three data insertions. The reason is that there are three types of internet traffic flow: Both Direction (Total), Forward, and Backward, and each type of flow has its own statistics. In the source dataset, all three types of traffic flow are mixed in a single table. Therefore, we need to classify and label each flow with the corresponding type and insert the statistics into each table.
- For the tables **TrafficStats**, **TrafficTime**, **Packet**, **PacketSize**, and **PacketInterarrivalTime**, there is a manually added attribute, type, to specify the direction of an internet traffic flow.

4. Client Application

4.1 Ideal client requirements

Based on the description of the project, the dataset we have been given and the goal of the client application articulated in the Introduction section, there is a list of ideal requirements we thought the client application should be able to accomplish:

1. The user who uses the client application does not need to know the structure of the database. They only need to know what attributes are available.
2. The ability to create user accounts with privilege restrictions. Users with low privilege can only query data and cannot change any data with the database. On the other hand, users with high privilege can both query data and modify the database.
3. The ability to log in to the client application with one of the created users.
4. The ability to re-assign privilege to an existing user with low privilege by one of the high privilege users.
5. The ability to do data manipulation on internet traffic data. Operations should include everything MySQL querying support. For example, all the aggregation operations, GROUP BY operation, natural/inner/outer/ join and so on. The purpose is that it will be easier for users to search the internet traffic flows based on what they want to know. For instance, a user may want to know the number of flows that happened during a period of time.
6. The ability to do data definition on internet traffic data. Ideal operations include updating multiple data in a table, adding new rows into the database, deleting rows from the database.
7. The ability to annotate attributes from different tables. Specifically, users can annotate data by creating custom views.

4.2 Actual client proposed

Due to time constrain, only part of the above functionalities can be implemented. Therefore, the actual client specifications will be a subset of the ideal client requirements. General descriptions of the implemented commands are stated below, a detailed description will be provided in section 4.3. The functions implemented are as follow:

1. The user who uses the client application does not need to know the structure of the database. They only need to know what attributes are available. The rule is accomplished except the add operation. In order to do an add operation, the user has to know the name of some tables.
2. The ability to create user accounts with privilege restrictions. Users with low privilege can only query data and cannot change any data with the database. On the other hand, users with high privilege can both query data and modify the database.
3. The ability to log in to the client application with one of the created users.
4. The ability to do data manipulation on internet traffic data with restrictions. Only general select operations and a subset of aggregation functions are implemented.

5. The ability to do data definition on internet traffic data with restrictions. Only update and add operations are implemented.
6. The ability to annotate attributes from different tables. Specifically, users can annotate data by creating custom views.

4.3 CLI usage description

Available attributes

This section shows what attributes are available for this CLI. The attributes are:

flow_key, Src_IP, Dst_IP, Src_port, Dst_port, timeStamp, duration, flowEndReason, web_service, category, proto, application_protocol, numeric_ip, flowStart flowEnd, flowDuration, pktTotalCount, octetTotalCount, max_ps, min_ps, avg_ps, std_dev_ps, min_piat, max_piat, avg_piat, std_dev_piat

Commands

This section shows what commands are available for this CLI.

Note: {optional conditions} will be described in Conditions section.

-dc command

Description: the command counts distinct values of an attribute with/without conditions applied.

Usage: -dc [attribute] {optional conditions}

Supported attributes: all

Example: -dc web_service -or --flowEndReason ge3 -c eqSystem

-c command

Description: the command counts values of an attribute with/without conditions applied.

Usage: -c [attribute] {optional conditions}

Supported attributes: all

Example: -c flowStart -sp eq50096 -fs lt1555954590

-ls command

Description: the command searches and displays values of one or more attributes with/without conditions applied.

Usage: -ls [attribute1 attribute2 …] {optional conditions}

Supported attributes: all

Example: -ls Src_IP -pc lt20000 -l 10

-max command

Description: the command searches the maximum values of an attribute with/without conditions applied.

Usage: -max [attribute] {optional conditions}

Supported attributes: "flowDuration", "max_ps", "min_ps", "avg_ps", "std_dev_ps", "min_piat", "max_piat", "avg_piat", "std_dev_piat", "pktTotalCount", "octetTotalCount"

Example: -max forward_avg_ps -or -ps gt1000 -ps lt100

-min command

Description: the command searches the minimum values of an attribute with/without conditions applied.

Usage: -min [attribute] {optional conditions}

Supported attributes: "flowDuration", "max_ps", "min_ps", "avg_ps", "std_dev_ps", "min_piat", "max_piat", "avg_piat", "std_dev_piat", "pktTotalCount", "octetTotalCount"

Example: -min proto -t gt2345678

-sum command

Description: the command computes the sum of values of an attribute with/without conditions applied.

Usage: -sum [attribute] {optional conditions}

Supported attributes: "flowDuration", "max_ps", "min_ps", "avg_ps", "std_dev_ps", "min_piat", "max_piat", "avg_piat", "std_dev_piat", "pktTotalCount", "octetTotalCount"

Example: -sum forward_flowDuration --proto in6,1

-avg command

Description: the command computes the average of values of an attribute with/without conditions applied.

Usage: -avg [attribute] {optional conditions}

Supported attributes: "flowDuration", "max_ps", "min_ps", "avg_ps", "std_dev_ps", "min_piat", "max_piat", "avg_piat", "std_dev_piat", "pktTotalCount", "octetTotalCount"

Example: -avg backward_min_ps -p eq17 --flowDuration le20000

-update command

Description: the command updates values of one or more attributes with/without conditions applied.

Usage: -update [attribute1 value1 attribute2 value2 …] {optional conditions}

Supported attributes: all

Example: -update proto 6 total_pktTotalCount 100 --flowEndReason gt8

-add command

1. Description: add a new row to a table that references table Flow. The users
   Usage: -add [existing table which has FK references Flow] [PK] [other attrs]
   Supported tables and attributes:
   - Protocol(flow_key, timeStamp, duration, proto, application_protocol)
   - Services(flow_key, timeStamp, duration, category, web_service)
   - TrafficStats(flow_key, timeStamp, duration, type)

   Example: -add Protocol flow_key "11111" timeStamp 1555966843.83052 duration 20 proto 1 application_protocol "Unknown"
2. Description: add a custom view with one or more attributes with/without conditions applied.
   Usage: -add [new view name] [attribute1 attribute2 …] {optional conditions}
   Supported attributes: all
   Example: -add test_table -t all -d gt100 -c eqNetwork

Conditions

This section describes optional conditions available for users. Multiple conditions can be used at the same time.

General condition usage: -[condition] [value]

Note: Condition commands are followed after commands specified in the Commands section.

Value specifications

This part describes how to specify values following the condition command.

eq[value]:

It allows the command to search for values equal to [value], followed after one of the condition commands.

Example: -dc proto -t eq10000

gt[value]:

It allows the command to search for values greater than [value], following after one of the condition commands.

Example: -dc proto -t gt10000

lt[value]:

It allows the command to search for values less than [value], following after one of the condition commands.

Example: -dc proto -t lt10000

ge[value]:

It allows the command to search for values greater than or equal to [value], following after one of the condition commands.

Example: -dc proto -t ge10000

le[value]:

It allows the command to search for values less than or equal to [value], following after one of the condition commands.

Example: -dc proto -t le10000

ne[value]:

It allows the command to search for values not equal to [value], following after one of the condition commands.

Example: -dc proto -t ne10000

in[value1,value2, …]:

It allows the command to search for values in [value1,value2, …], following after one of the condition commands.

Example: -dc proto -t in(10000, 20000)

ni[value1,value2, …]:

It allows the command to search for values not in [value1,value2, …], following after one of the condition commands.

Example: -dc proto -t ni(10000, 20000)

all:

It allows the command to search for all values, following after one of the condition commands.

Descriptions of condition commands

-t condition command

Description: This command specifies a timestamp constraint.

Example: -dc total_max_ps -t eq10000

-d condition command

Descritpion: this command specifies duration constraint.

Example: -dc total_max_ps -d le2000

-p condition command

Description: This command specifies a protocol constraint.

Example: -dc total_max_ps -p eq6

-s condition command

Description: This command specifies a web service constraint.

Example: -dc total_max_ps -s eqHTTP

-c condition command

Description: This command specifies a category constraint.

Example: -dc total_max_ps -c eqWeb

-pc condition command

Description: This command specifies a total number of packets constraint.

Example: -dc total_max_ps -pc lt20000

-ps condition command

Description: This command specifies a total number of bytes constraint.

Example: -dc total_max_ps -pc lt20000

-ap condition command

Description: This command specifies an application protocol constraint.

Example: -dc total_max_ps -ap in(Unknow,TLS)

-sp condition command

Description: This command specifies a source port constraint.

Example: -dc total_max_ps -sp eq67

-dp condition command

Description: This command specifies a destination port constraint.

Example: -dc total_max_ps -dp eq67

-fs condition command

Description: This command specifies a flow start constraint.

Example: -dc total_max_ps -fs ge9000

-fe condition command

Description: This command specifies a flow end constraint.

Example: -dc total_max_ps -fe ge9000

-fd condition command

Description: This command specifies a flow duration constraint.

Example: -dc total_max_ps -fsd gt8700

-n condition command

Description: this command specifies numeric source ip constraint.

Example: -dc total_max_ps -n ge30000000

-l condition command

Description: This command specifies that the query outputs are limited by a certain number. This is mainly designed for select queries. Value specification rules do not apply here, users can input a value directly.

Example: -ls total_max_ps -l 10

-w condition command

Description: This command specifies the name of the file the user wants to print the query output to. This is mainly designed for select queries. Value specification rules do not apply here, user can input a file name directly.

Example: -ls total_max_ps -w Output.txt

--[attribute name] condition command

Description: This command allows users to use any attribute as a condition.

Example: -dc total_max_ps --flowEndReason eq4

-and

Description: This command specifies that an "and" operator will be used for all conditions in where clause. No value follows after this command. Note that if nothing is specified, the default operator will be "and".

Example: -dc total_max_ps --flowEndReason eq4 -n ge30000000 -and

-or

Description: This command specifies that an "or" operator will be used for all conditions in where clause. No value follows after this command. Note that if nothing is specified, the default operator will be "and".

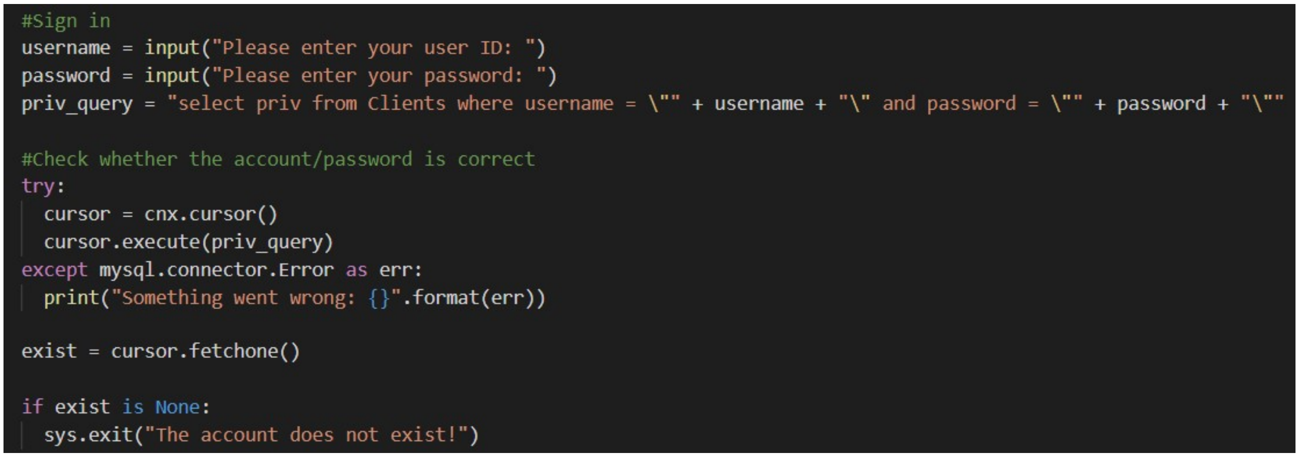
Example: -dc total_max_ps --flowEndReason eq4 -n ge30000000 -or

4.4 Implementation

client_app.py

The main program of the CLI is implemented in this file. It includes the process of authentication, and it calls the function in create_account.py when account creation is needed. After authentication, it reads the input commands and passes them to the function in handle_command.py.

A table called Clients is created in the database to store user authentication information. The program uses the priv_query in Figure 4.4.1 to fetch the privilege of the user as well as determine the existence of the input username & password. If there's no such user, the program will exit with the message "The account does not exist!" Otherwise, it will start to read input commands.

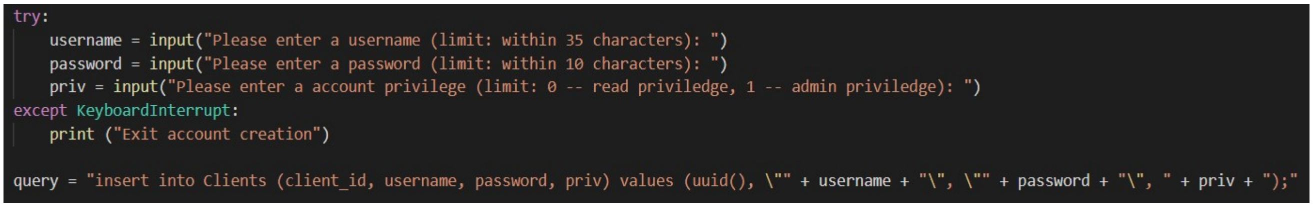


```
#Sign in
username = input("Please enter your user ID: ")
password = input("Please enter your password: ")
priv_query = "select priv from Clients where username = \"" + username + "\" and password = \"" + password + "\""

#Check whether the account/password is correct
try:
  cursor = cnx.cursor()
  cursor.execute(priv_query)
except mysql.connector.Error as err:
  print("Something went wrong: {}".format(err))

exist = cursor.fetchone()

if exist is None:
  sys.exit("The account does not exist!")
```

Figure 4.4.1 Process of Authentication

create_account.py

A user is required to enter the username, password and privilege when creating an account. The user either has read-only permission to the database or admin permission which can update and insert new data. This information will be stored in the Clients table using the "insert into" command as shown in Figure 4.4.2.



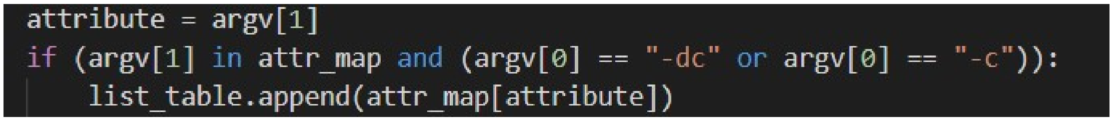
```
try:
    username = input("Please enter a username (limit: within 35 characters): ")
    password = input("Please enter a password (limit: within 10 characters): ")
    priv = input("Please enter a account privilege (limit: 0 -- read priviledge, 1 -- admin priviledge): ")
except KeyboardInterrupt:
    print ("Exit account creation")

query = "insert into Clients (client_id, username, password, priv) values (uuid(), \"" + username + "\", \"" + password + "\", " + priv + ");"
```

Figure 4.4.2 Process of Account Creation

handle_command.py

This program calls the function handle_command when a user input a new command. There're six steps in total to read the command and translate it to a MySQL query, then execute the query and print the result. There're three important variables in this process which are the list to store the tables involved (*list_table*), the list to store the conditions (*list_cond*) and the variable/list to store the attributes involved (*attribute*).

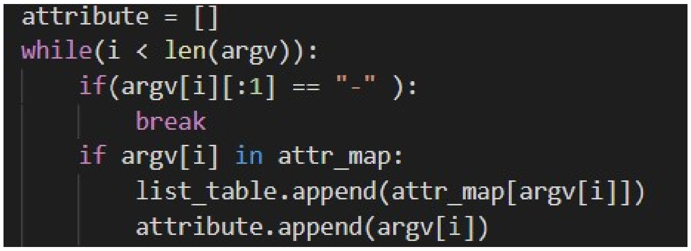
The first step is to read the command type and the attributes. The allowed command type is introduced in Section 4.3. For most commands, there's only one attribute involved. For example, we can only count the number of rows of one attribute at a time. In this case, the variable *attribute is* simply equal to the name of that attribute.



```
attribute = argv[1]
if (argv[1] in attr_map and (argv[0] == "-dc" or argv[0] == "-c")):
    list_table.append(attr_map[attribute])
```

Figure 4.4.3 Example of Processing most command types

However, for commands like "-ls", there could be multiple attributes. For example, we can list the flow key and the source IP at the same time. In this case, we continue reading the attributes until the argument starts with "-" which is the symbol of a condition. The attributes are appended to the variable *attribute* which is a list in this case.

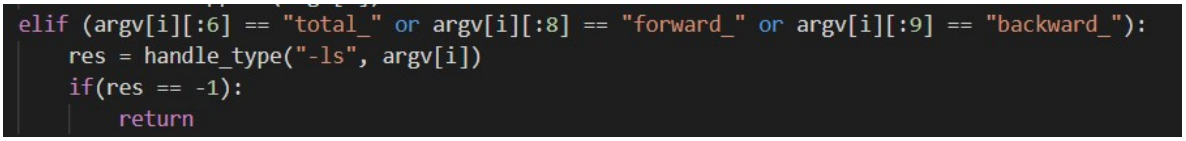


```
attribute = []
while(i < len(argv)):
    if(argv[i][:1] == "-" ):
        break
    if argv[i] in attr_map:
        list_table.append(attr_map[argv[i]])
        attribute.append(argv[i])
```

Figure 4.4.4 Example of Processing Commands with Multiple Attributes

The map *attr_map* in map_file.py is used to append tables to *list_table* as shown in Figure 4.4.3 and Figure 4.4.4. It is a map with attribute names as keys and corresponding table names as values. The tables of the attributes involved in this step as well as the second step should be appended to list_table, so they can be joined together later.

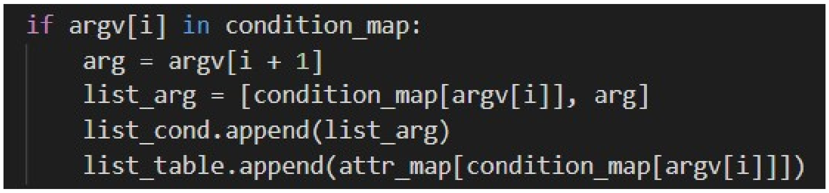
There's a special case of the attributes. In the ER model, we combine the statistics of the entire flow, forward flow and backward flow in one table with an additional attribute 'type' to separate them. Therefore, the attribute name could start with 'total_', 'forward_' and 'backward_'. A function called handle_type is created to translate this attribute name to the attribute name in the table. For example, 'total_flowDuration' will be translated to flowDuration with the condition 'type = total'. The function can be used as the way in Figure 4.4.5.



```
elif (argv[i][:6] == "total_" or argv[i][:8] == "forward_" or argv[i][:9] == "backward_"):
    res = handle_type("-ls", argv[i])
    if(res == -1):
        return
```

Figure 4.4.5 Example of Implementing handle_type Function

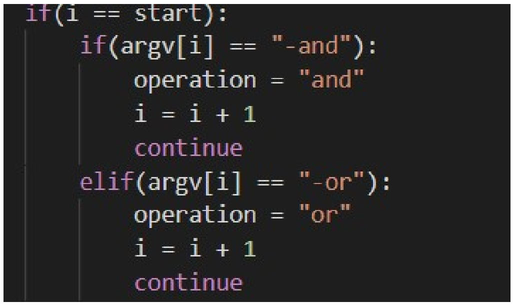
The second step is to read the conditions. There could be multiple conditions with the format as '--flowEndReason eq3' which means 'flowEndReaon = 3'. The flowEndReason can be replaced with any other attribute in the *attr_map,* and there're also some shortcuts that are introduced in Section 4.3. The attributes and the conditions are stored in *list_cond*, and the tables corresponding to the attributes are stored in *list_table* as shown in Figure 4.4.6.



```
if argv[i] in condition_map:
    arg = argv[i + 1]
    list_arg = [condition_map[argv[i]], arg]
    list_cond.append(list_arg)
    list_table.append(attr_map[condition_map[argv[i]]])
```

Figure 4.4.6 Example of Processing a Condition

The user can also specify the relation of the conditions before inputting the conditions. The default relation is an intersection which is 'and'. The user can input either '-and' or '-or' to specify the relation. This is optional. The default relation will be used if there's no such input.

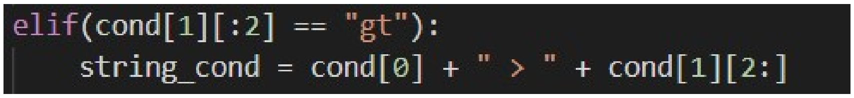


```
if(i == start):
    if(argv[i] == "-and"):
        operation = "and"
        i = i + 1
        continue
    elif(argv[i] == "-or"):
        operation = "or"
        i = i + 1
        continue
```

Figure 4.4.7 Code of Specifying Relation of Conditions

The third step is to translate the conditions to equations. The program read the conditions in *list_cond* and translated them to equations that can be recognized by MySQL. The equations are stored in the list list_string_cond. For example, '-d gt100' will be translated to 'duration > 100', and '-p in6,17' will be translated to 'proto in (6,17)'. There are eight types of signs supported, which are listed in the table below:

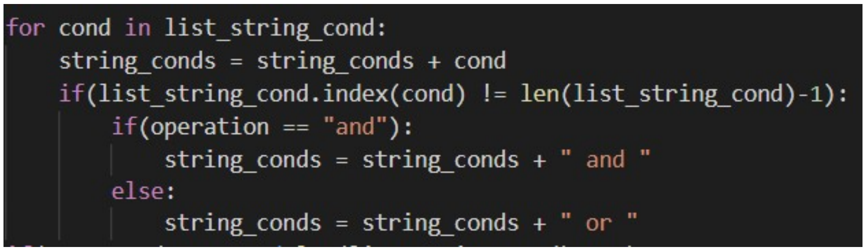
condition	sign	explanation
eq	=	equal to
gt	>	greater than
lt	<	less than
ge	>=	greater than or equal to
le	<=	less than or equal to
ne	<>	not equal to
in	in	in
ni	not in	not in



```
elif(cond[1][:2] == "gt"):
    string_cond = cond[0] + " > " + cond[1][2:]
```

Figure 4.4.8 Example of Translating Conditions

The fourth step is to combine the conditions with input relation (and, or). The combined condition is stored in a string called *string_conds.* The format of it is '*cond* and *cond* and *cond…'* or *'cond* or *cond…'* according to the input relation. The process is shown in the following figure:

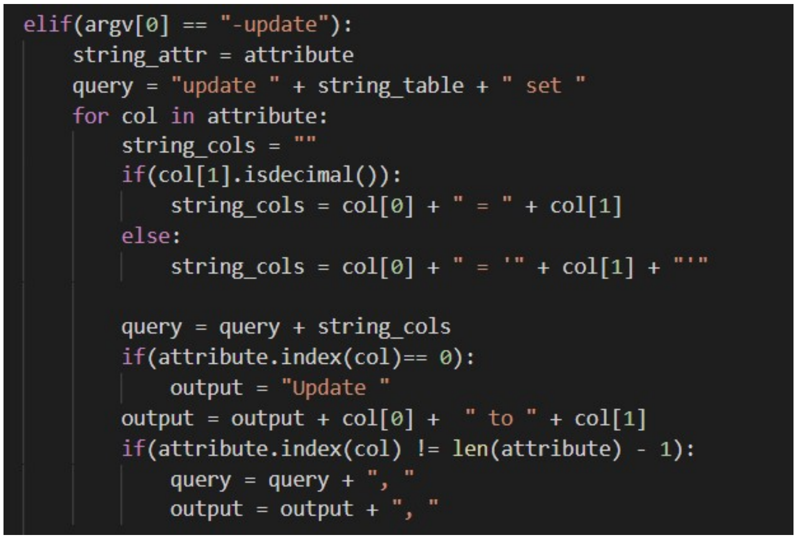


```
for cond in list_string_cond:
    string_conds = string_conds + cond
    if(list_string_cond.index(cond) != len(list_string_cond)-1):
        if(operation == "and"):
            string_conds = string_conds + " and "
        else:
            string_conds = string_conds + " or "
```

Figure 4.4.9 Process of combining relations

There is a special case here which is the type attribute described in the first step. The type attribute will always be 'or' to each other and 'and' to the other conditions. For example, if the user input '-ls forward_flowDuration backward_flowDuration -or -p eq6 --flowEndReason ne2', the program will translate it to (type = 'forward or type = 'backward') and (proto = 6 or flowEndReason <> 2).

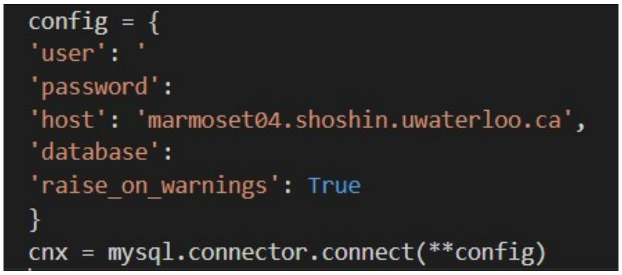
The fifth step is to create the MySQL query. In this process, each command type is considered separately because the format of the query is different. The tables in *list_table* here are combined using 'natural join' and stored in the variable *string_table*. When creating the MySQL query, the previous variables *string_table, attribute* and *string_conds* are used, and the created query is stored in the variable *query*. An example of the command "-update" is shown below:



```
elif(argv[0] == "-update"):
    string_attr = attribute
    query = "update " + string_table + " set "
    for col in attribute:
        string_cols = ""
        if(col[1].isdecimal()):
            string_cols = col[0] + " = " + col[1]
        else:
            string_cols = col[0] + " = '" + col[1] + "'"

        query = query + string_cols
        if(attribute.index(col)== 0):
            output = "Update "
        output = output + col[0] +  " to " + col[1]
        if(attribute.index(col) != len(attribute) - 1):
            query = query + ", "
            output = output + ", "
```

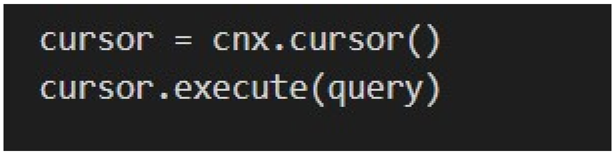
Figure 4.4.10 Example of creating MySQL query

The sixth step is to execute the query and print the result. This is the last step of handing command. We connect to the databases using *mysql.connector.connect*, and then use *cnx.cursor* to execute the queries. The result can either be printed to the kernel or printed to an input file using the option '-w'. The process of connecting to the database and executing the queries is like the following:



```
config = {
'user': '
'password':
'host': 'marmoset04.shoshin.uwaterloo.ca',
'database':
'raise_on_warnings': True
}
cnx = mysql.connector.connect(**config)
```

Figure 4.4.11 Process of connecting to the database



```
cursor = cnx.cursor()
cursor.execute(query)
```

Figure 4.4.12 Process of executing queries

5. Test Plan

5.1 Server test plan

Data error:

The table creation ignores flows with the same flow_key/timeStamp/duration (15 flows) considering them as duplicated records.

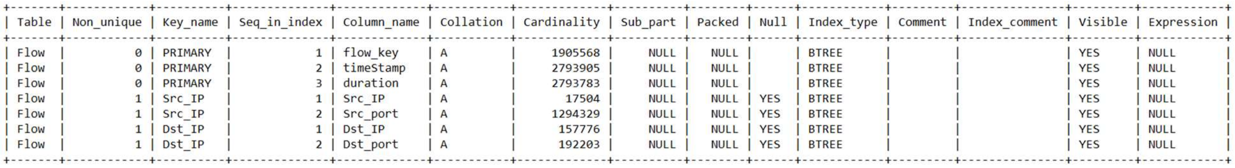
Test case 1

Purpose: check whether the indices of primary keys are successfully created in various tables

Test query1: show index from Flow;

Expected: flow_key, timeStamp, duration

Actual:

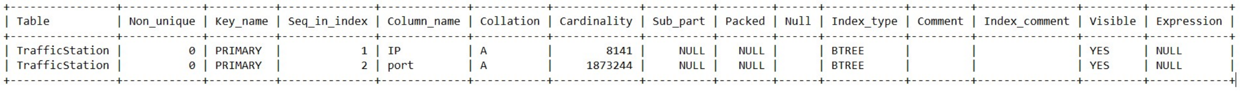


```
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Flow  |          0 | PRIMARY  |            1 | flow_key    | A         |     1905568 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Flow  |          0 | PRIMARY  |            2 | timeStamp   | A         |     2793905 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Flow  |          0 | PRIMARY  |            3 | duration    | A         |     2793783 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Flow  |          1 | Src_IP   |            1 | Src_IP      | A         |       17504 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| Flow  |          1 | Src_IP   |            2 | Src_port    | A         |     1294329 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| Flow  |          1 | Dst_IP   |            1 | Dst_IP      | A         |      157776 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| Flow  |          1 | Dst_IP   |            2 | Dst_port    | A         |      192203 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
```

Test query2: show index from TrafficStation;

Expected: primary key: IP, Port

Actual:

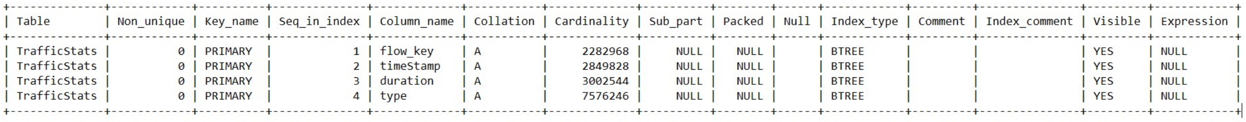


```
+----------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table          | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+----------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| TrafficStation |          0 | PRIMARY  |            1 | IP          | A         |        8141 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| TrafficStation |          0 | PRIMARY  |            2 | port        | A         |     1873244 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+----------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
```

Test query3: show index from TrafficStats;

Expected: flow_key, timeStamp, duration, type

Actual:

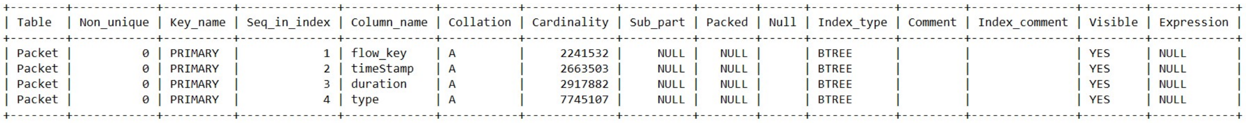


```
+--------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table        | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+--------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| TrafficStats |          0 | PRIMARY  |            1 | flow_key    | A         |     2282968 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| TrafficStats |          0 | PRIMARY  |            2 | timeStamp   | A         |     2849828 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| TrafficStats |          0 | PRIMARY  |            3 | duration    | A         |     3002544 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| TrafficStats |          0 | PRIMARY  |            4 | type        | A         |     7576246 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+--------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
```

Test query4: show index from Packet;

Expected: flow_key, timeStamp, duration, type

Actual:

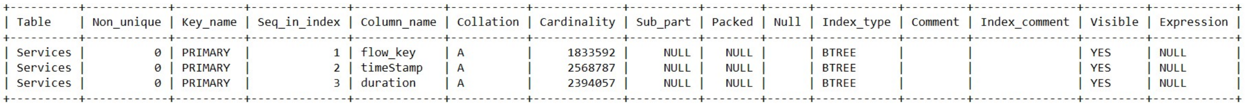


```
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Packet |          0 | PRIMARY  |            1 | flow_key    | A         |     2241532 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Packet |          0 | PRIMARY  |            2 | timeStamp   | A         |     2663503 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Packet |          0 | PRIMARY  |            3 | duration    | A         |     2917882 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Packet |          0 | PRIMARY  |            4 | type        | A         |     7745107 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
```

Test query5: show index from Services;

Expected: flow_key, timeStamp, duration

Actual:



```
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table    | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Services |          0 | PRIMARY  |            1 | flow_key    | A         |     1833592 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Services |          0 | PRIMARY  |            2 | timeStamp   | A         |     2568787 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Services |          0 | PRIMARY  |            3 | duration    | A         |     2394057 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
```

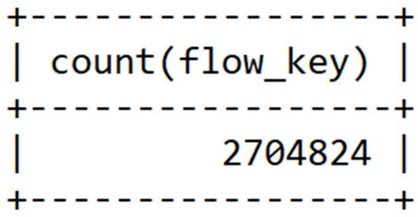
Test case 2

Purpose: check whether the data has been loaded completely by checking the row number

Test query1: select count(flow_key) from Flow;

Expected: 2704824 (csv file row count (2704840) - header (1) - ignored dupicated records(15))

Actual:

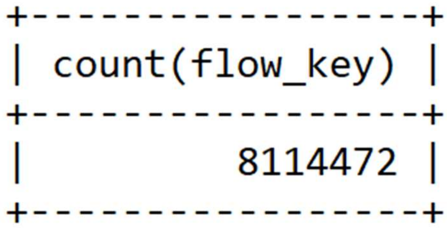


```
+-----------------+
| count(flow_key) |
+-----------------+
|         2704824 |
+-----------------+
```

Test query2: select count(flow_key) from TrafficTime;

Expected: 8114472 (2704824 * 3)

Actual:

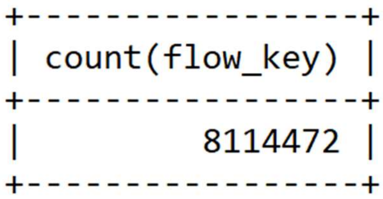


```
+-----------------+
| count(flow_key) |
+-----------------+
|         8114472 |
+-----------------+
```

Test query3: select count(flow_key) from PacketInterarrivalTime;

Expected: 8114472 (2704824 * 3)

Actual:



```
+-----------------+
| count(flow_key) |
+-----------------+
|         8114472 |
+-----------------+
```

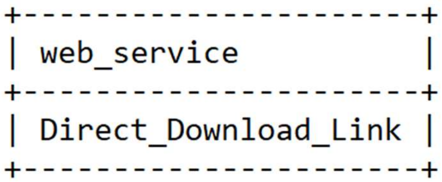
Test case 3

Purpose: check whether the char length allocated to the attribute is sufficient

Test query1: select distinct web_service from Services where length(web_service) = (select max(length(web_service)) from Services);

Expected: a complete name of the web_service with the longest length: Direct_Download_Link

Actual:

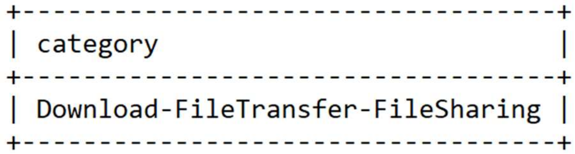


```
+----------------------+
| web_service          |
+----------------------+
| Direct_Download_Link |
+----------------------+
```

Test query2: select distinct category from Services where length(category) = (select max(length(category)) from Services);

Expected: a complete name of the category with the longest length: Download-FileTransfer-FileSharing

Actual:



```
+-----------------------------------+
| category                          |
+-----------------------------------+
| Download-FileTransfer-FileSharing |
+-----------------------------------+
```

Test case 4

Purpose: check the completeness of attributes and correctness of attributes types

Test query1: describe Flow;

Expected:

flow_key char(32),

timeStamp double precision,

duration double precision,

Src_IP varchar(15),

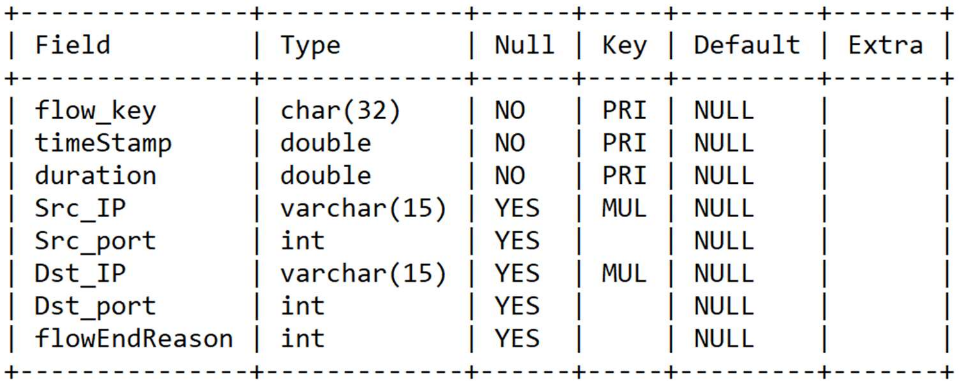
Src_port int,

Dst_IP varchar(15),

Dst_port int,

flowEndReason int

Actual:



```
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| flow_key      | char(32)    | NO   | PRI | NULL    |       |
| timeStamp     | double      | NO   | PRI | NULL    |       |
| duration      | double      | NO   | PRI | NULL    |       |
| Src_IP        | varchar(15) | YES  | MUL | NULL    |       |
| Src_port      | int         | YES  |     | NULL    |       |
| Dst_IP        | varchar(15) | YES  | MUL | NULL    |       |
| Dst_port      | int         | YES  |     | NULL    |       |
| flowEndReason | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

Test query2: describe Protocol;

Expected:

flow_key char(32),

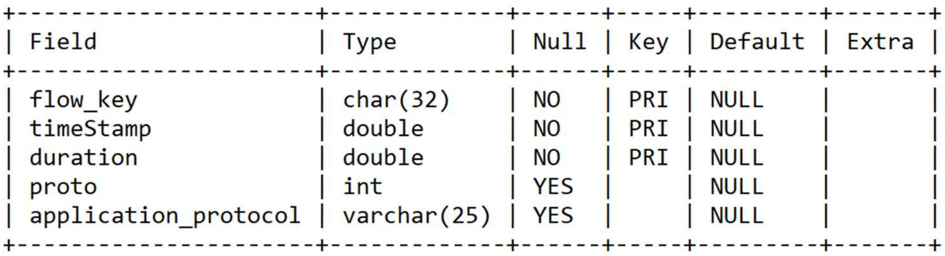
timeStamp double precision,

duration double precision,

proto int,

application_protocol varchar(25)

Actual:



```
+----------------------+-------------+------+-----+---------+-------+
| Field                | Type        | Null | Key | Default | Extra |
+----------------------+-------------+------+-----+---------+-------+
| flow_key             | char(32)    | NO   | PRI | NULL    |       |
| timeStamp            | double      | NO   | PRI | NULL    |       |
| duration             | double      | NO   | PRI | NULL    |       |
| proto                | int         | YES  |     | NULL    |       |
| application_protocol | varchar(25) | YES  |     | NULL    |       |
+----------------------+-------------+------+-----+---------+-------+
```

Test query 3: describe PacketSize;

Expected:

flow_key char(32),

timeStamp double precision,

duration double precision,

type enum('forward', 'backward', 'total'),

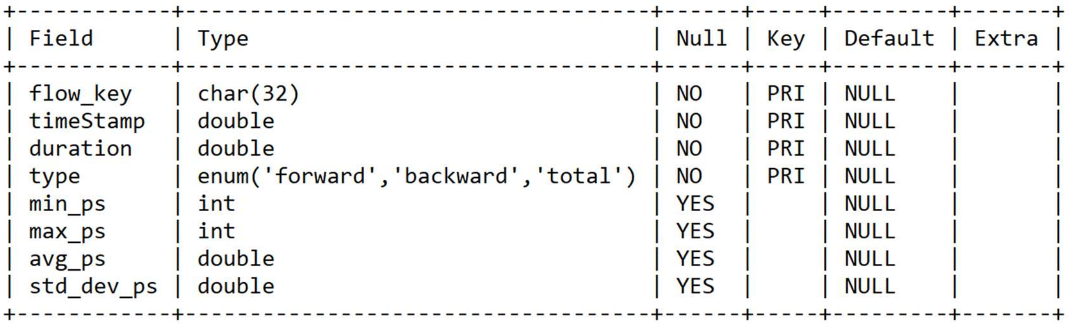
min_ps int,

max_ps int,

avg_ps double precision,

std_dev_ps double precision

Actual:



```
+------------+-----------------------------------+------+-----+---------+-------+
| Field      | Type                              | Null | Key | Default | Extra |
+------------+-----------------------------------+------+-----+---------+-------+
| flow_key   | char(32)                          | NO   | PRI | NULL    |       |
| timeStamp  | double                            | NO   | PRI | NULL    |       |
| duration   | double                            | NO   | PRI | NULL    |       |
| type       | enum('forward','backward','total') | NO   | PRI | NULL    |       |
| min_ps     | int                               | YES  |     | NULL    |       |
| max_ps     | int                               | YES  |     | NULL    |       |
| avg_ps     | double                            | YES  |     | NULL    |       |
| std_dev_ps | double                            | YES  |     | NULL    |       |
+------------+-----------------------------------+------+-----+---------+-------+
```

Test case 5

Purpose: check whether the foreign key indexes are successfully created in various tables

Test query1:

```
SELECT
    TABLE_NAME,
    COLUMN_NAME,
    CONSTRAINT_NAME,
    REFERENCED_TABLE_NAME,
    REFERENCED_COLUMN_NAME
FROM
    INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE REFERENCED_TABLE_NAME = 'Flow';
```

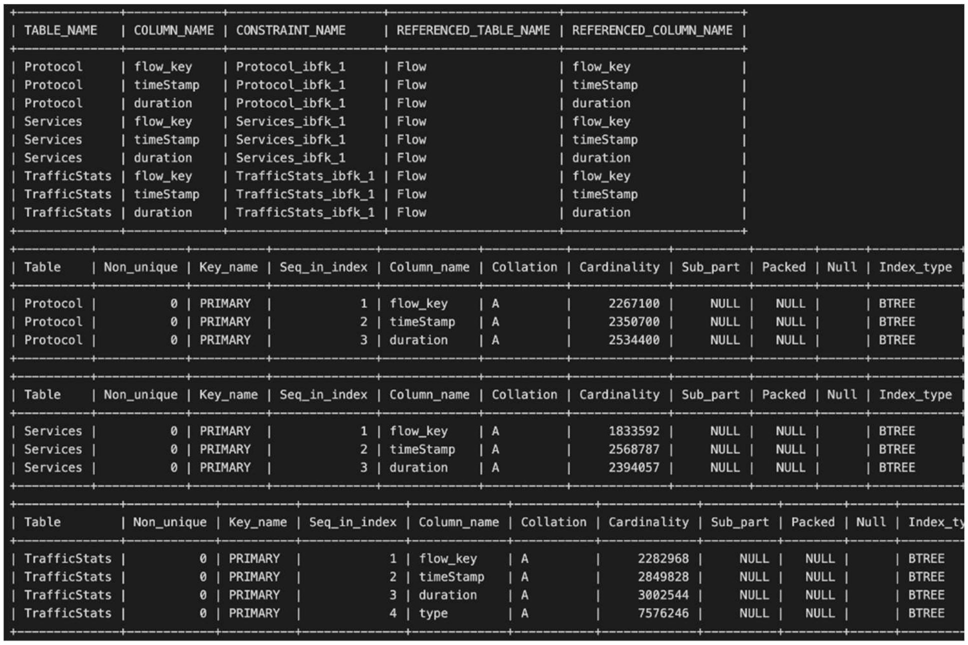
Expected:

Protocol(flow_key, timeStamp, duration) references Flow(flow_key, timeStamp, duration)

Services(flow_key, timeStamp, duration) references Flow(flow_key, timeStamp, duration)

TrafficStats(flow_key, timeStamp, duration) references Flow(flow_key, timeStamp, duration)

Actual:



```
+--------------+-------------+---------------------+-----------------------+------------------------+
| TABLE_NAME   | COLUMN_NAME | CONSTRAINT_NAME     | REFERENCED_TABLE_NAME | REFERENCED_COLUMN_NAME |
+--------------+-------------+---------------------+-----------------------+------------------------+
| Protocol     | flow_key    | Protocol_ibfk_1     | Flow                  | flow_key               |
| Protocol     | timeStamp   | Protocol_ibfk_1     | Flow                  | timeStamp              |
| Protocol     | duration    | Protocol_ibfk_1     | Flow                  | duration               |
| Services     | flow_key    | Services_ibfk_1     | Flow                  | flow_key               |
| Services     | timeStamp   | Services_ibfk_1     | Flow                  | timeStamp              |
| Services     | duration    | Services_ibfk_1     | Flow                  | duration               |
| TrafficStats | flow_key    | TrafficStats_ibfk_1 | Flow                  | flow_key               |
| TrafficStats | timeStamp   | TrafficStats_ibfk_1 | Flow                  | timeStamp              |
| TrafficStats | duration    | TrafficStats_ibfk_1 | Flow                  | duration               |
+--------------+-------------+---------------------+-----------------------+------------------------+
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Table    | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Protocol |          0 | PRIMARY  |            1 | flow_key    | A         |     2267100 |     NULL |   NULL |      | BTREE      |
| Protocol |          0 | PRIMARY  |            2 | timeStamp   | A         |     2350700 |     NULL |   NULL |      | BTREE      |
| Protocol |          0 | PRIMARY  |            3 | duration    | A         |     2534400 |     NULL |   NULL |      | BTREE      |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Table    | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Services |          0 | PRIMARY  |            1 | flow_key    | A         |     1833592 |     NULL |   NULL |      | BTREE      |
| Services |          0 | PRIMARY  |            2 | timeStamp   | A         |     2568787 |     NULL |   NULL |      | BTREE      |
| Services |          0 | PRIMARY  |            3 | duration    | A         |     2394057 |     NULL |   NULL |      | BTREE      |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
+--------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+--------
| Table        | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_ty
+--------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+--------
| TrafficStats |          0 | PRIMARY  |            1 | flow_key    | A         |     2282968 |     NULL |   NULL |      | BTREE
| TrafficStats |          0 | PRIMARY  |            2 | timeStamp   | A         |     2849828 |     NULL |   NULL |      | BTREE
| TrafficStats |          0 | PRIMARY  |            3 | duration    | A         |     3002544 |     NULL |   NULL |      | BTREE
| TrafficStats |          0 | PRIMARY  |            4 | type        | A         |     7576246 |     NULL |   NULL |      | BTREE
+--------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+--------
```

From the first table, we can see the foreign key constraints for Protocol, Services, and TrafficStats exist. From the remaining three tables, we can see the foreign keys can be used as indexes to look up the records in the table Flow.

Test query2:

```
SELECT
    TABLE_NAME,
    COLUMN_NAME,
    CONSTRAINT_NAME,
    REFERENCED_TABLE_NAME,
    REFERENCED_COLUMN_NAME
FROM
    INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE REFERENCED_TABLE_NAME = 'TrafficStation';
```

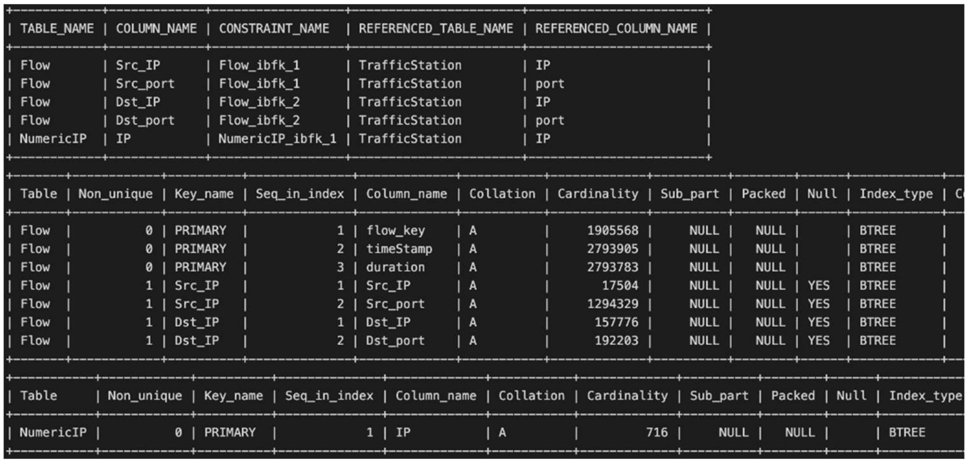
Expected:

Flow(Src_IP, Src_port) references TrafficStation(IP, port)

Flow(Dst_IP, Dst_port) references TrafficStation(IP, port)

NumericIP(IP) references TrafficStation(IP)

Actual:



```
+------------+-------------+------------------+-----------------------+------------------------+
| TABLE_NAME | COLUMN_NAME | CONSTRAINT_NAME  | REFERENCED_TABLE_NAME | REFERENCED_COLUMN_NAME |
+------------+-------------+------------------+-----------------------+------------------------+
| Flow       | Src_IP      | Flow_ibfk_1      | TrafficStation        | IP                     |
| Flow       | Src_port    | Flow_ibfk_1      | TrafficStation        | port                   |
| Flow       | Dst_IP      | Flow_ibfk_2      | TrafficStation        | IP                     |
| Flow       | Dst_port    | Flow_ibfk_2      | TrafficStation        | port                   |
| NumericIP  | IP          | NumericIP_ibfk_1 | TrafficStation        | IP                     |
+------------+-------------+------------------+-----------------------+------------------------+
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+--
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | C
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+--
| Flow  |          0 | PRIMARY  |            1 | flow_key    | A         |     1905568 |     NULL |   NULL |      | BTREE      |
| Flow  |          0 | PRIMARY  |            2 | timeStamp   | A         |     2793905 |     NULL |   NULL |      | BTREE      |
| Flow  |          0 | PRIMARY  |            3 | duration    | A         |     2793783 |     NULL |   NULL |      | BTREE      |
| Flow  |          1 | Src_IP   |            1 | Src_IP      | A         |       17504 |     NULL |   NULL | YES  | BTREE      |
| Flow  |          1 | Src_IP   |            2 | Src_port    | A         |     1294329 |     NULL |   NULL | YES  | BTREE      |
| Flow  |          1 | Dst_IP   |            1 | Dst_IP      | A         |      157776 |     NULL |   NULL | YES  | BTREE      |
| Flow  |          1 | Dst_IP   |            2 | Dst_port    | A         |      192203 |     NULL |   NULL | YES  | BTREE      |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+--
+-----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+-----------
| Table     | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type
+-----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+-----------
| NumericIP |          0 | PRIMARY  |            1 | IP          | A         |         716 |     NULL |   NULL |      | BTREE
+-----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+-----------
```

From the first table, we can see the foreign key constraints for Flow and NumericIP exist. From the remaining two tables, we can see the foreign keys can be used as indexes to look up the records in the table TrafficStation.

Test query3:

```
SELECT
    TABLE_NAME,
    COLUMN_NAME,
    CONSTRAINT_NAME,
    REFERENCED_TABLE_NAME,
    REFERENCED_COLUMN_NAME
FROM
    INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE REFERENCED_TABLE_NAME = ' TrafficStats';
```

Expected:

Packet(flow_key, timeStamp, duration) references TrafficStats(flow_key, timeStamp, duration)

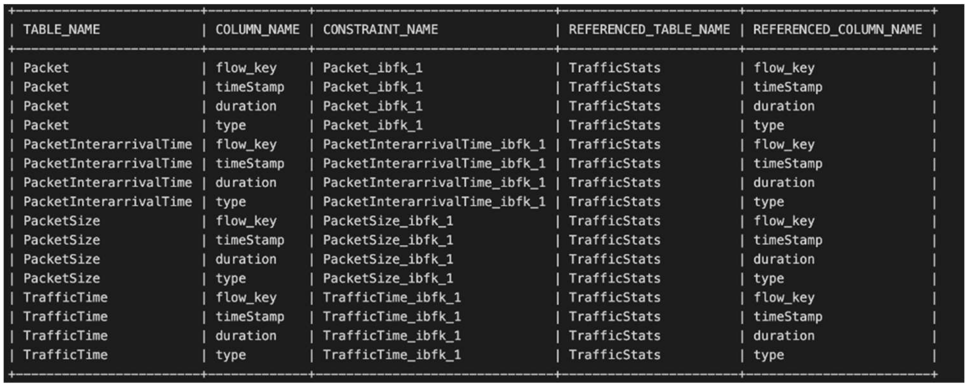
PacketInterarrivalTime(flow_key, timeStamp, duration)

references TrafficStats(flow_key, timeStamp, duration)

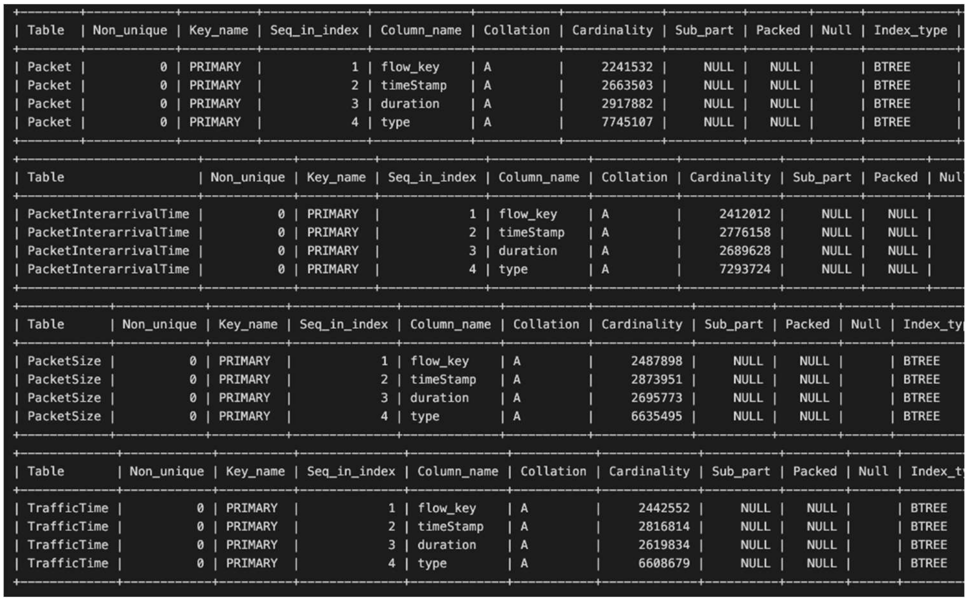
PacketSize(flow_key, timeStamp, duration) references TrafficStats(flow_key, timeStamp, duration)

TrafficTime(flow_key, timeStamp, duration) references TrafficStats(flow_key, timeStamp, duration)

Actual:



```
+-----------------------+-------------+-----------------------------+-----------------------+------------------------+
| TABLE_NAME            | COLUMN_NAME | CONSTRAINT_NAME             | REFERENCED_TABLE_NAME | REFERENCED_COLUMN_NAME |
+-----------------------+-------------+-----------------------------+-----------------------+------------------------+
| Packet                | flow_key    | Packet_ibfk_1               | TrafficStats          | flow_key               |
| Packet                | timeStamp   | Packet_ibfk_1               | TrafficStats          | timeStamp              |
| Packet                | duration    | Packet_ibfk_1               | TrafficStats          | duration               |
| Packet                | type        | Packet_ibfk_1               | TrafficStats          | type                   |
| PacketInterarrivalTime | flow_key   | PacketInterarrivalTime_ibfk_1 | TrafficStats        | flow_key               |
| PacketInterarrivalTime | timeStamp  | PacketInterarrivalTime_ibfk_1 | TrafficStats        | timeStamp              |
| PacketInterarrivalTime | duration   | PacketInterarrivalTime_ibfk_1 | TrafficStats        | duration               |
| PacketInterarrivalTime | type       | PacketInterarrivalTime_ibfk_1 | TrafficStats        | type                   |
| PacketSize            | flow_key    | PacketSize_ibfk_1           | TrafficStats          | flow_key               |
| PacketSize            | timeStamp   | PacketSize_ibfk_1           | TrafficStats          | timeStamp              |
| PacketSize            | duration    | PacketSize_ibfk_1           | TrafficStats          | duration               |
| PacketSize            | type        | PacketSize_ibfk_1           | TrafficStats          | type                   |
| TrafficTime           | flow_key    | TrafficTime_ibfk_1          | TrafficStats          | flow_key               |
| TrafficTime           | timeStamp   | TrafficTime_ibfk_1          | TrafficStats          | timeStamp              |
| TrafficTime           | duration    | TrafficTime_ibfk_1          | TrafficStats          | duration               |
| TrafficTime           | type        | TrafficTime_ibfk_1          | TrafficStats          | type                   |
+-----------------------+-------------+-----------------------------+-----------------------+------------------------+
```



```
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Packet |          0 | PRIMARY  |            1 | flow_key    | A         |     2241532 |     NULL |   NULL |      | BTREE      |
| Packet |          0 | PRIMARY  |            2 | timeStamp   | A         |     2663503 |     NULL |   NULL |      | BTREE      |
| Packet |          0 | PRIMARY  |            3 | duration    | A         |     2917882 |     NULL |   NULL |      | BTREE      |
| Packet |          0 | PRIMARY  |            4 | type        | A         |     7745107 |     NULL |   NULL |      | BTREE      |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
+------------------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+-----
| Table                  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Nul
+------------------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+-----
| PacketInterarrivalTime |          0 | PRIMARY  |            1 | flow_key    | A         |     2412012 |     NULL |   NULL |
| PacketInterarrivalTime |          0 | PRIMARY  |            2 | timeStamp   | A         |     2776158 |     NULL |   NULL |
| PacketInterarrivalTime |          0 | PRIMARY  |            3 | duration    | A         |     2689628 |     NULL |   NULL |
| PacketInterarrivalTime |          0 | PRIMARY  |            4 | type        | A         |     7293724 |     NULL |   NULL |
+------------------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+-----
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+--------
| Table      | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_ty
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+--------
| PacketSize |          0 | PRIMARY  |            1 | flow_key    | A         |     2487898 |     NULL |   NULL |      | BTREE
| PacketSize |          0 | PRIMARY  |            2 | timeStamp   | A         |     2873951 |     NULL |   NULL |      | BTREE
| PacketSize |          0 | PRIMARY  |            3 | duration    | A         |     2695773 |     NULL |   NULL |      | BTREE
| PacketSize |          0 | PRIMARY  |            4 | type        | A         |     6635495 |     NULL |   NULL |      | BTREE
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+--------
+-------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+-------
| Table       | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_t
+-------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+-------
| TrafficTime |          0 | PRIMARY  |            1 | flow_key    | A         |     2442552 |     NULL |   NULL |      | BTREE
| TrafficTime |          0 | PRIMARY  |            2 | timeStamp   | A         |     2816814 |     NULL |   NULL |      | BTREE
| TrafficTime |          0 | PRIMARY  |            3 | duration    | A         |     2619834 |     NULL |   NULL |      | BTREE
| TrafficTime |          0 | PRIMARY  |            4 | type        | A         |     6608679 |     NULL |   NULL |      | BTREE
+-------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+-------
```

From the first table, we can see the foreign key constraints for Packet, PacketInterarrivalTime, PacketSize, and TrafficTime exist. From the remaining four tables, we can see the foreign keys can be used as indexes to look up the records in the table TrafficStats.

5.2 Client app test plan

Query data tests

Test case 1

Purpose: test "-dc" command

Test query1:

Purpose: test no attribute error case

Command: -dc -d gt1000

Expected output: Error: Invalid attribute: -d

Test query2:

Purpose: test incorrect use of command value

Command: -dc proto -t 100000

Expected output: Error: Invalid comparison: 10

Test query3:

Purpose: test distinct count without conditions

Command: -dc total_max_ps -c eqWeb

Expected SQL query: select count(distinct max_ps) from PacketSize natural join Services where type = 'total' and category = 'Web';

Output: The result is: 13169

Test query4:

Purpose: test distinct count with multiple conditions using "or"

Command: -dc web_service -or --flowEndReason ge3 -c eqSystem

Expected SQL query: select count(distinct web_service) from Services natural join Flow where flowEndReason >= 3 or category = 'System';

Output: The result is: 110

Test case 2

Purpose: test "-c" command

Test query1:

Purpose: check whether "and" is applied in where clauses if "-and" is not specified

Command: -c flowStart -sp eq50096 -fs lt1555954590

Expected SQL query:

select count(flowStart) from TrafficTime natural join Flow where Src_port = 50096 and flowStart < 1555954590;

Expected output: 47

Actual output: The result is: 47

Test query2:

Purpose: check whether an attribute is counted properly

Command: -c flowEnd -and -pc gt20 -fd gt30

Expected SQL query:

select count(flowEnd) from TrafficTime natural join Packet where pktTotalCount > 20 and flowDuration > 30;

Output: The result is: 807051

Test query3:

Purpose: test count with multiple conditions using "or"

Command: -c flow_key -or -t gt1600000000 -ap eqHTTP

Expected SQL query:

select count(flow_key) from Flow natural join Protocol where timestamp > 1600000000 or application_protocol = 'HTTP'

Output:

The result is: 128372

Test case 3

Purpose: test "-ls" command

Test query1:

Purpose: test whether "-ls" command selects an attribute under a condition correctly.

Command: -ls Src_IP -pc lt2876909 -l 10

Expected query: select distinct Src_IP from Flow natural join Packet where pktTotalCount < 2876909 limit 10;

Output:

```
('192.168.125.194',)

('192.168.128.68',)

('192.168.122.154',)

('192.168.127.49',)

('192.168.122.11',)

('192.168.125.70',)

('192.168.127.13',)

('192.168.125.17',)

('192.168.122.76',)

('192.168.125.89',)
```

Test query2:

Purpose: test if results are printed to target file correctly

Command: -ls all -dp eq80 -l 20 -w test_query2.txt

Expected query: select * from Flow natural join Protocol natural join Services natural join NumericIP natural join TrafficTime natural join Packet natural join PacketSize natural join PacketInterarrivalTime where Dst_port = 80 limit 20;

Output: test_query2.txt

Test query3:

Purpose: test if "-ls" executes properly with multiple conditions

Command: -ls flowEndReason -or -p eq6 -s eqGoogle -l 10

Expected query: select distinct flowEndReason from Flow natural join Protocol natural join Services where proto = 6 or web_service = 'Google' limit 10

```
(2,)
```

```
(4,)

(3,)

(5,)
```

Test case 4

Purpose: test "-max" command

Test query1:

Purpose: test if the maximum value of an attribute is selected properly without any conditions

Command: -max total_max_ps

Expected SQL query:

```
   select max(max_ps) from PacketSize where type = 'total'
```

Output:

```
   The result is: 26320
```

Test query2:

Purpose: test if the maximum value of an attribute is selected properly with multiple conditions applying "-or" command

Command: -max forward_avg_ps -or -ps gt1000 -ps lt100

Expected SQL query:

```
   select max(avg_ps) from PacketSize natural join Packet where type = 'forward' and (octetTotalCount >
1000 or octetTotalCount < 100)
```

Output:

```
   The result is: 11624.0
```

Test query3:

Purpose: test if the maximum value of an attribute is selected properly with multiple conditions applying "-and" command

Command: -max backward_min_piat -and -s eqMicrosoft -c eqWeb

Expected SQL query:

```
   select max(min_piat) from PacketInterarrivalTime natural join Services where type = 'backward' and (web_service = 'Microsoft' and category = 'Web')
```

Output:

```
   The result is: 1643.9050681591
```

Test case 5

Purpose: test "-min" command

Test query1:

Purpose: test invalid attribute error case

Command: -min proto -t gt2345678

Expected output: Error: Invalid attribute: proto

Test query2:

Purpose: test if the minimum value of an attribute is selected properly with multiple conditions

Command: -min backward_avg_piat -and -ps eq3000 -t ge123450

Expected query:

```
   select min(avg_piat) from PacketInterarrivalTime natural join Packet natural join Flow where type = 'backward' and (octetTotalCount = 3000 and timestamp >= 123450)
```

Output:

```
   The result is: 7.86781311035156e-06
```

Test query3:

Purpose: test finding the minimum of an attribute applying "ne" for condition

Command: -min total_octetTotalCount --flowEndReason ne2

Expected query:

```
 select min(octetTotalCount) from Packet natural join Flow where type = 'total' and flowEndReason <> 2
```

Output:

```
   The result is: 80
```

Test case 6

Purpose: test "-sum" command

Test query1:

Purpose: test summing up values of an attribute applying "in" for condition

Command: -sum forward_flowDuration --proto in6,1

Expected query:

```
   select sum(flowDuration) from TrafficTime natural join Protocol where type = 'forward' and proto in (6, 1)
```

Output:

```
   The result is: 97599621.77848792
```

Test query2:

Purpose: test summing up values of an attribute applying "in" for condition

Command: -sum total_pktTotalCount -ap niUnknown,TLS

Expected query:

```
   select sum(pktTotalCount) from Packet natural join Protocol where type = 'total' and application_protocol not in ('Unknown', 'TLS')
```

Output:

```
   The result is: 54347695
```

Test query3:

Purpose: test summing up values of an attribute without any condition

Command: -sum total_avg_piat

Expected query:

```
select sum(avg_piat) from PacketInterarrivalTime where type = 'total'
```

Output:

```
The result is: 13005733.413340945
```

Test case 7

Purpose: test "-avg" command

Test query1:

Purpose: test multiple attributes error case (only one attribute is allowed)

Command: -avg total_pktTotalCount total_octetTotalCount

Expected output:

```
Error: Invalid condition: total_octetTotalCount
```

Test query2:

Purpose: test if command of the form "--flowDuration" works properly

Command: -avg backward_min_ps -p eq17 --flowDuration le20000

Expected query:

```
select avg(min_ps) from PacketSize natural join Protocol natural join TrafficTime where type = 'backward' and (proto = 17 and flowDuration <= 20000)
```

Output:

```
The result is: 111.3020
```

Test query3:

Purpose: test invalid condition operator error case (only "-and" and "-or" are allowed)

Command: -avg forward_avg_ps -xor -s eqAmazon

Expected output: Error: Invalid condition: -xor

Update/Input data tests

Test case 1

Purpose: test "-update" command

Test query1:

Purpose: test if updating a value is successful

Command: -update flowEndReason 7 --flowEndReason eq5

Expected query:

```
update Flow set flowEndReason = 7 where flowEndReason = 5
```

Test query2:

Purpose: test invalid use of "-update" command (value of min_ps is not declared)

Command: -update min_ps -dp eq0

Expected output:

```
Error: Invalid attribute: min_ps
```

Test query3:

Purpose: check that users without privilege cannot use "-update" command

Command: -update flowEndReason 8 --flowEndReason eq8

Expected output:

```
Error: Permission denied
```

Test query4:

Purpose: check update with multiple conditions

Command: -update proto 17 --flowEndReason eq7 --avg_piat gt1000

Expected query:

```
update Protocol natural join Flow natural join PacketInterarrivalTime

set proto = 17

where flowEndReason = 7 and avg_piat > 1000
```

Test query5:

Purpose: check update attribute in different tables

Command: -update proto 6 total_pktTotalCount 100 --flowEndReason gt8

Expected query:

update Protocol natural join Packet natural join Flow

set proto = 6, pktTotalCount = 100

where type = 'total' and flowEndReason > 8

Test case 2

Purpose: test if the command “-add” works properly

Test query1:

Purpose: users can add a new row to Protocol table which has foreign key constraint Protocol(flow_key, timeStamp, duration) references Flow(flow_key, timeStamp, duration)

Command: -add Protocol flow_key "11111" timeStamp 1555966843.83052 duration 20 proto 1 application_protocol "Unknown"

expected query:

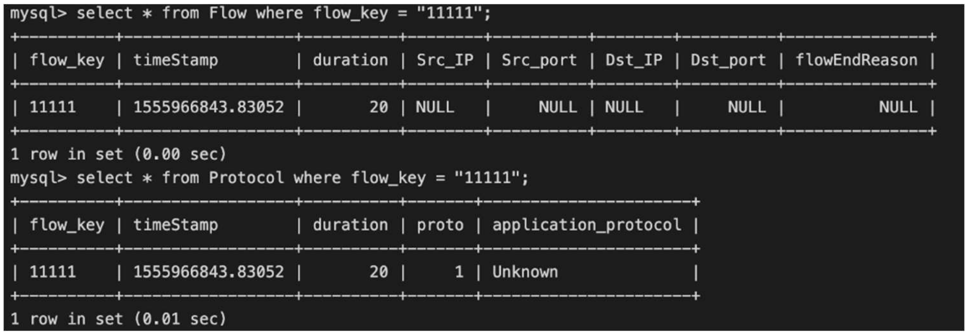
insert into Flow (flow_key, timeStamp, duration) values ("11111", 1555966843.83052, 20)

insert into Protocol (flow_key, timeStamp, duration, proto, application_protocol)

values ("11111", 1555966843.83052, 20, 1, "Unknown")

output:

Check if the new row is added to Flow and Protocol tables:



```
mysql> select * from Flow where flow_key = "11111";
+----------+------------------+----------+--------+----------+--------+----------+---------------+
| flow_key | timeStamp        | duration | Src_IP | Src_port | Dst_IP | Dst_port | flowEndReason |
+----------+------------------+----------+--------+----------+--------+----------+---------------+
| 11111    | 1555966843.83052 |       20 | NULL   |     NULL | NULL   |     NULL |          NULL |
+----------+------------------+----------+--------+----------+--------+----------+---------------+
1 row in set (0.00 sec)
mysql> select * from Protocol where flow_key = "11111";
+----------+------------------+----------+-------+----------------------+
| flow_key | timeStamp        | duration | proto | application_protocol |
+----------+------------------+----------+-------+----------------------+
| 11111    | 1555966843.83052 |       20 |     1 | Unknown              |
+----------+------------------+----------+-------+----------------------+
1 row in set (0.01 sec)
```

As we can see from the results above, the insertion is successful

Test query2:

Purpose: user can create a customized view by identifying the columns of his/her interest

Command: -add test_table -t all -d gt100 -c eqNetwork

expected query:

create view test_table as

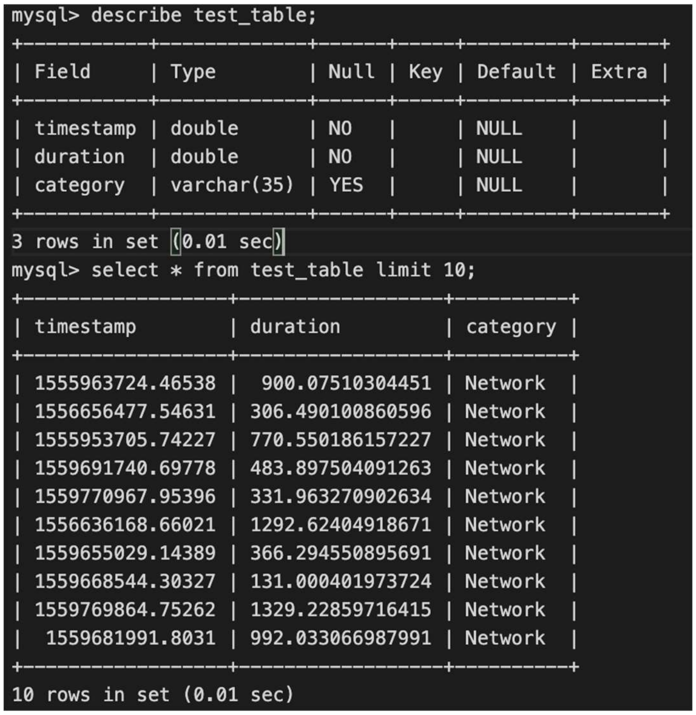
select timestamp, duration, category

from Flow natural join Services

where duration > 100 and category = 'Network';

output:

Check if the new table test_table was created



```
mysql> describe test_table;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| timestamp | double      | NO   |     | NULL    |       |
| duration  | double      | NO   |     | NULL    |       |
| category  | varchar(35) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
mysql> select * from test_table limit 10;
+------------------+------------------+----------+
| timestamp        | duration         | category |
+------------------+------------------+----------+
| 1555963724.46538 |  900.07510304451 | Network  |
| 1556656477.54631 | 306.490100860596 | Network  |
| 1555953705.74227 | 770.550186157227 | Network  |
| 1559691740.69778 | 483.897504091263 | Network  |
| 1559770967.95396 | 331.963270902634 | Network  |
| 1556636168.66021 | 1292.62404918671 | Network  |
| 1559655029.14389 | 366.294550895691 | Network  |
| 1559668544.30327 | 131.000401973724 | Network  |
| 1559769864.75262 | 1329.22859716415 | Network  |
|  1559681991.8031 | 992.033066987991 | Network  |
+------------------+------------------+----------+
10 rows in set (0.01 sec)
```

From the results above, we can see the new table is created successfully with desired columns

Test query3:

Purpose: if users entered an invalid command, stop handling the command and throw an error

-add Protocol -t all -d gt100 -c eqNetwork

expected output: The CLI should reject this command and throws an error

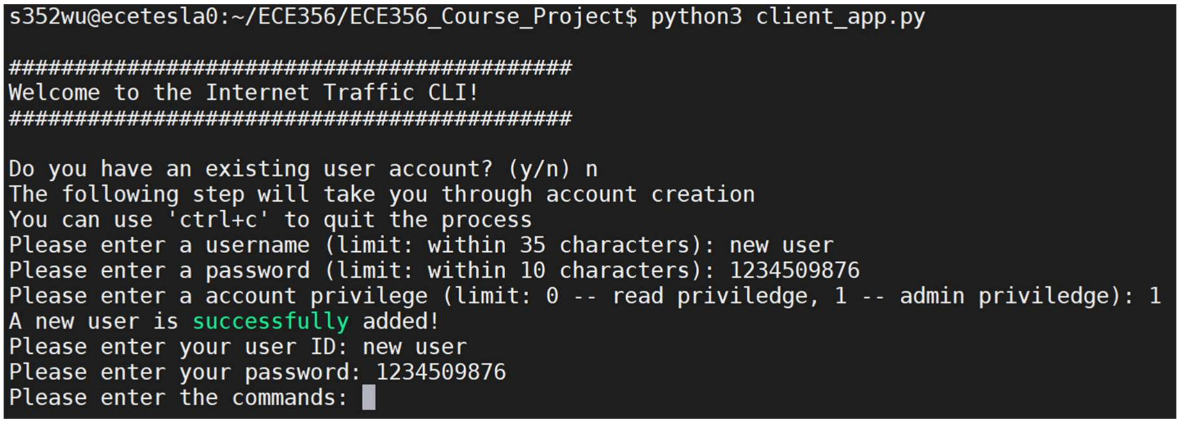
output: Error, the table Protocol already exists!

Create/login account tests

Test case 1

Purpose: test creating a new user and login CLI

Expected result: a new user is successfully created and login

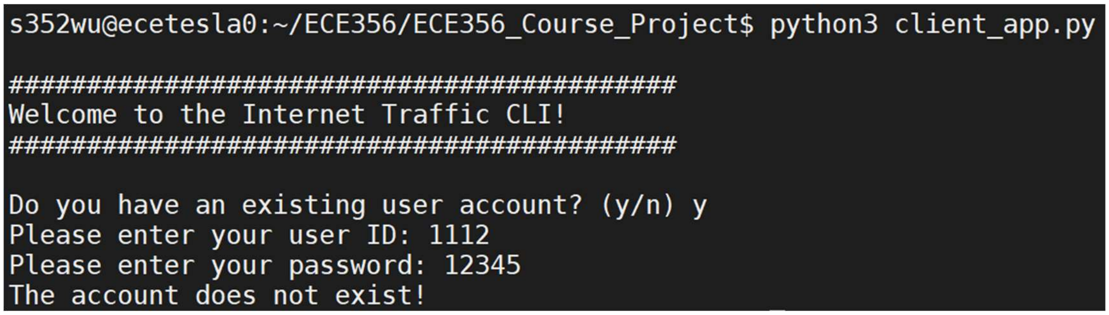


```
s352wu@ecetesla0:~/ECE356/ECE356_Course_Project$ python3 client_app.py

###########################################
Welcome to the Internet Traffic CLI!
###########################################

Do you have an existing user account? (y/n) n
The following step will take you through account creation
You can use 'ctrl+c' to quit the process
Please enter a username (limit: within 35 characters): new user
Please enter a password (limit: within 10 characters): 1234509876
Please enter a account privilege (limit: 0 -- read priviledge, 1 -- admin priviledge): 1
A new user is successfully added!
Please enter your user ID: new user
Please enter your password: 1234509876
Please enter the commands:
```

Test case 2

Purpose: test login CLI with incorrect username/password

Expected result: login should not succeed

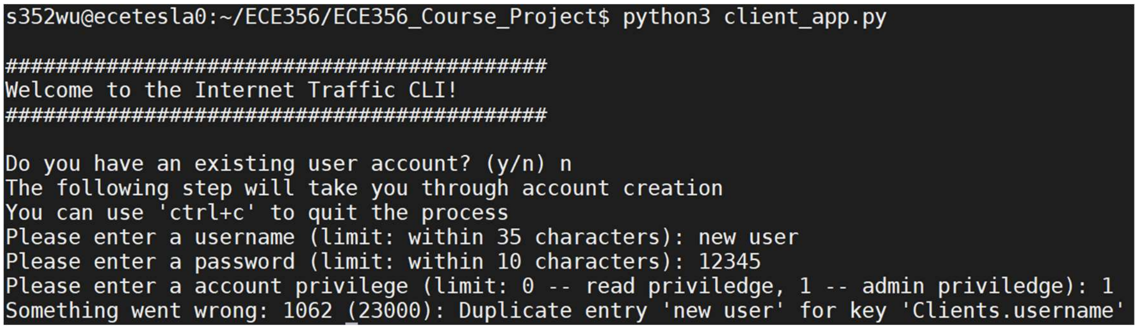


```
s352wu@ecetesla0:~/ECE356/ECE356_Course_Project$ python3 client_app.py

###########################################
Welcome to the Internet Traffic CLI!
###########################################

Do you have an existing user account? (y/n) y
Please enter your user ID: 1112
Please enter your password: 12345
The account does not exist!
```

Test case 3

Purpose: test creating a duplicate user (username should be unique)

Expected result: duplicate user cannot be created

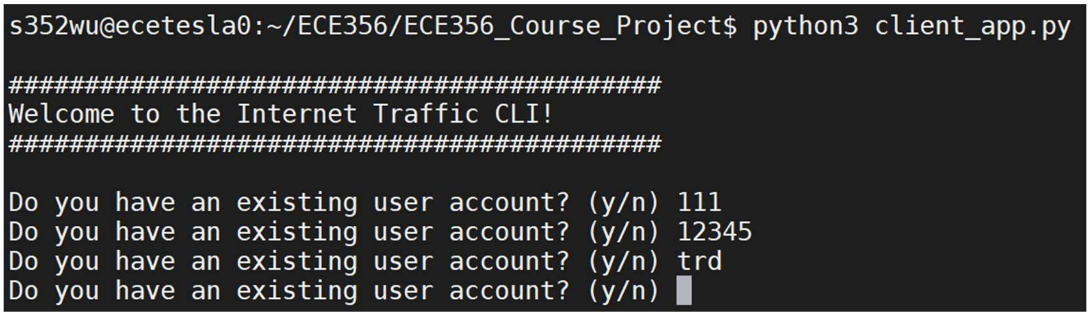


```
s352wu@ecetesla0:~/ECE356/ECE356_Course_Project$ python3 client_app.py

###########################################
Welcome to the Internet Traffic CLI!
###########################################

Do you have an existing user account? (y/n) n
The following step will take you through account creation
You can use 'ctrl+c' to quit the process
Please enter a username (limit: within 35 characters): new user
Please enter a password (limit: within 10 characters): 12345
Please enter a account privilege (limit: 0 -- read priviledge, 1 -- admin priviledge): 1
Something went wrong: 1062 (23000): Duplicate entry 'new user' for key 'Clients.username'
```

Test case 4

Purpose: check that users cannot enter words other than "y, n, yes, no" when login

Expected result: users will be asked to re-enter their choices



```
s352wu@ecetesla0:~/ECE356/ECE356_Course_Project$ python3 client_app.py

###########################################
Welcome to the Internet Traffic CLI!
###########################################

Do you have an existing user account? (y/n) 111
Do you have an existing user account? (y/n) 12345
Do you have an existing user account? (y/n) trd
Do you have an existing user account? (y/n) 
```

6. Data Mining

A user might be interested in knowing that, among all the internet traffic flows, how the frequency of the web services changes at different hours of a day, and thus to estimate the approximate number of a web service to be visited at different times in the future. This question can be answered by applying data mining techniques to the data we collected.

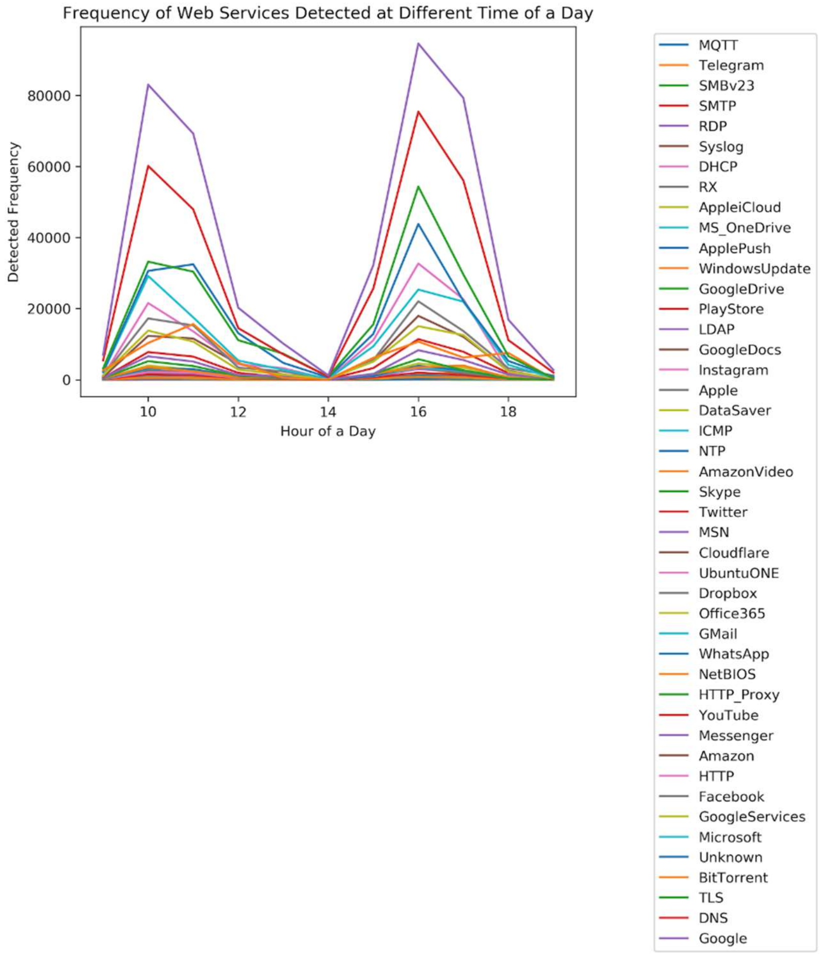
The technique applied in the data mining exercise is clustering. The following metrics are considered when exploring the problem.

- Web services
- Hours during the day

The data is collected by using ***data_mining.sql*** script to collect the web services and their corresponding detected frequencies at each hour, some irrelevant web services or hours are filtered out.

Since we are interested in the internet traffic flows in different hours during a day, we first group the flows by hours (based on the timestamp) from the table Flow. Additionally, in order to relate the usage frequency of web services, we further process through grouping by web services and compute the frequency of use of various web services. As a result, we obtain output data that shows how usage of web services evolves during a day.

To show the result, a line chart is plotted as below:



The result shows that, in general, we can see that the frequency of web services explodes during 10 am to 12 pm and 14 pm to 18 pm during a day. Furthermore, the maximum frequency exceeds 80,000 at 16pm, the most used web service is Google, following by DNS and TLS. It is also observed that for most web services, the frequencies stay roughly the same for both morning and afternoon.