ECE 9014: *Identification and Detection of Denial-of-Service attacks to develop secure software systems.*

Group # 15:
Devashish Singh Rajput (drajput@uwo.ca)
Kanika Gandhi(kgandh27@uwo.ca)
Karanpartap Singh Aulakh(kaulakh4@uwo.ca)
Shrutam Parmar(sparma55@uwo.ca)

Department of Electrical & Computer Engineering
The University of Western Ontario

Table of Content

Overview

Cyber-crime has increased significantly over the few years, and with increasing technology, the number of chances that technology can be compromised. Denial of Service is an attack that much like it's name, maliciously denies the service or access to a network. This document describes how data taken from a an ELEGANT dataset of Denial of Service attacks is used to create logical models, conceptual and normalized models. These models are optimized to create physical schemas and populate data to understand the analysis of data better through the database model. By creating views and manipulating data, the data in this model has been easy to access and made reader friendly to allow people over a vast horizon to understand this database model.

The aim of this document is to be able to help people understand and analyze data for Denial-of-service attacks better to create awareness and imply preventive measures.

1 Group Deliverables

Work	Done By
1.1 Project Description	Kanika Gandhi+Karanpartap Singh Aulakh
1.2 Business Rules	Kanika Gandhi+Karanpartap Singh Aulakh
1.3 Conceptual Model	Kanika Gandhi
1.4 Logical Model	Karanpartap Singh Aulakh
1.5 Physical Schema	Devashish Singh Rajput
1.6 Data Population	Shrutam Parmar
1.7 Views	Devashish Singh Rajput+ Shrutam Parmar
1.8 Data Manipulation	Devashish Singh Rajput+ Shrutam Parmar

1.1 Project Description

Denial Of Service is a cyber attack which is much more common than it seems. This attack has been used for multiple malicious reasons such as ransom, online rivalry, dirty business games and politics and many more. This attack has been known to increase by 125 % on a quarter-to-quarter basis. DOS has compromised over thousands of systems, ruined business reputations, induced monetary losses and also caused severe outages.

The need analysis for the presence of a database model to be able to identify and detect this DOS attack has risen for the same reason. There are multiple datasets on the web that describe the statical data showing the factors that are allowing, affecting and posing as vulnerabilities for this attack to occur.

It is important to prevent such attacks in the future as only then will there be a development of safe and secure software systems. So, using one such data set (https://arxiv.org/abs/2103.09380) we have developed a database model that based on statistical data that can help prevent such DOS attacks. Extensions have been added to fine tune the database model to cater it further, making it specific to the situation type for denial-of-service attacks.

1.2 Business Rules

The following business rules are followed for this model:
1. Source and Destination should be connected to a network.
2. Source and Destination IPs should exist.
3. Destination should be LISTENING mode.
4. No packet should have more than 1 source IP and destination IP each.

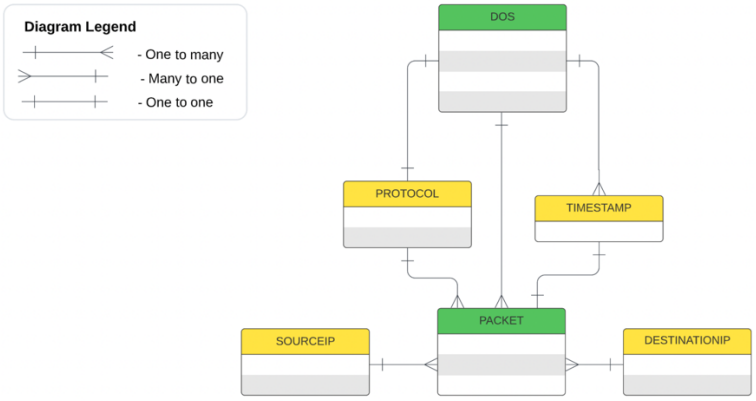
1.3 Conceptual Model

The conceptual model is based on the following relations:

Conceptual Model is the highest level of the representation of the data in the dataset. Users can get an overview of the whole data from seeing a conceptual model. The whole system has entities like DOS, Protocol, Timestamp, Packet, SourceIP and DestinationIP. Each entity is connected to another entity using relationship such as one to one, one to many, many to many and many to one.

Going by the diagram, DOS is connected to the Timestamp Entity in one-to-many relation. This simply means that DOS can have multiple timestamps and many timestamps correspond to one DOS. Going by the diagram, protocol and DOS have one to one relationship i.e., for each protocol there is a corresponding DOS attack. DOS and packet are in one-to-many relationship with each other meaning that for every DOS attack there are multiple packets and vice versa. SourceIP has Source_ID and Source attributes and DestinationIP has Destination_ID and Destination attributes. SourceIP and DestinationIP have one to many relationship with Packet entity means that one sourceIP can send multiple packet and likewise DestinationIP can receive multiple packets.

So, the conceptual model has turned out to be as follows:



1.4 Logical Model

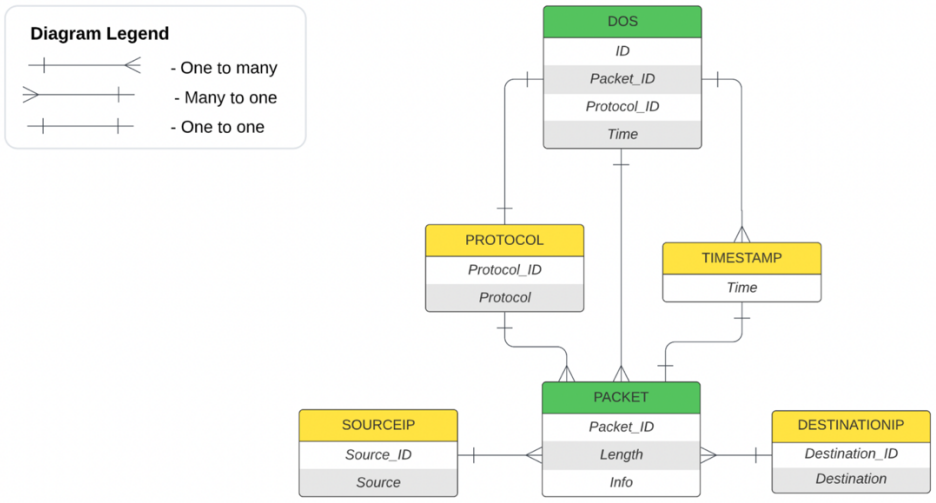
The logical model has been based off the relational model and conceptual model depicted above. The logical model is a foundation model for this is a model that contains the more structured explanation of the conceptual model. The model consists of 6 entities, namely DOS, PROTOCOL, TIMESTAMP, SOURCEIP, PACKET, DESTINATIONIP.

The TIMESTAMP entity is the simplest entity, rideen with anh attribute, ‘Time’ that defines the time at which the packets that are used for transmission in the attack are sent.

The PROTOCOL entity contains of two attributes, ‘Protocol_ID’ and ’Protocol’. The Protocol_ID uniquely defines the types of protocol through which the packets travel between two locations.

The SOURCEIP and DESTINATIONIP contain similar attributes. Namely, ‘Source_ID’ and ‘Source’ for the former, and the ‘Destination_ID’ and ‘Destination’ for the latter. The SOURCEIP entity contains the primary key Source_ID responsible for uniquely identifying different IP addresses through which a packet is initiated before transmission. The DESTINATIONIP entity contains the primary key Destination_ID responsible for uniquely identifying different IP addresses at which a packet is sent or reaches post transmission.

The DOS entity contains four different attributes, ‘ID’, ’Protocol_ID’, ’Packet_ID’, ’Time’ references the primary key values from other tables to this entity table.



1.4.1 Normalized Model

The normalized form consists of three forms, 1NF, 2NF, 3NF. The normal forms in this model are at the logical model state because, the first normal form state indicates atomicity in values. The values in the table are all in atomic state except the values in the 'Info' attribute, although this attribute contains values which cannot coexist without relation and the entity will be considered as null. This makes our model in first form. For partial dependency, when closely noticed none of the data should be partially dependent on half or a portion of the primary key of the model (Protocol_ID+Time+Packet_ID). This makes our model in the second form. Lastly, as none of the non-prime attributes are not depending on the primary key, hence this model is in third form. As this model is in 3NF, the logical model is our normalized model.

1.5 Physical Model: PostgreSQL

Tables Protocol, TIMESTAMP, SOURCEIP, DESTINATION IP, DOS, and PACKET are part of the general group work. First, we created a table called Protocol with a Protocol_ID column as the primary key and a Protocol column. The TIMESTAMP table is then created, with Time as a column and primary key. Following that, there is a SOURCEIP table with columns such as Source_ID, which is the primary key, and Source, and a DESTINATIONIP table with columns such as Destination ID, Destination. Following that, a DOS table is created. It has columns like ID, Packet_ID, Protocol_ID, and Time. The primary key in the DOS table is ID, and the foreign keys are Packet_ID and Protocol_ID (Key creation is done after populating the data). Then there's the PACKET table, which contains the Packet_ID, Length and Info as columns.

```sql
CREATE TABLE PROTOCOL (
  Protocol_ID integer PRIMARY KEY,
  Protocol varchar(255) NOT NULL
);

CREATE TABLE TIMESTAMP (
  Time float PRIMARY KEY
);

CREATE TABLE SOURCEIP (
  Source_ID integer PRIMARY KEY,
  Source varchar NOT NULL
);

CREATE TABLE DESTINATIONIP (
  Destination_ID integer PRIMARY KEY,
  Destination varchar NOT NULL
);

CREATE TABLE PACKET (
  Packet_ID integer PRIMARY KEY,
  Length int4 NOT NULL,
  Info varchar NOT NULL);


CREATE TABLE DOS (
  ID integer PRIMARY KEY,
  Packet_ID integer NOT NULL REFERENCES PACKET(Packet_ID) ,
  Protocol_ID integer NOT NULL REFERENCES PROTOCOL(Protocol_ID),
  Time float NOT NULL
);
```

1.6 Data Population: PostgreSQL

SET search_path TO 'public';

We create a table with raw data to include all the entities such as No., time, source, destination, Protocol, Length, Info, Source Port and Destination Port with primary key as No.

```
CREATE TABLE "raw_data" (
 "No." int4,
 "Time" float,
 "Source" varchar,
 "Destination" varchar,
 "Protocol" varchar,
 "Length" int4,
 "Info" varchar,
 "Source Port" int4,
 "Destination Port" int4,
 PRIMARY KEY ("No.")
);
```

Selecting all rows from the raw data

```
SELECT * FROM raw_data
```

Now we create table Protocol and insert data from raw data into it.

```
CREATE TABLE "PROTOCOL" (
  "Protocol_ID" Serial PRIMARY KEY,
  "Protocol" varchar
);
INSERT INTO "PROTOCOL"("Protocol")
SELECT "Protocol" FROM raw_data;
```

Now we create table Timestamp and insert data from raw data into it.

```
CREATE TABLE "TIMESTAMP"(
  Time float PRIMARY KEY
);
INSERT INTO "TIMESTAMP" ("time")
SELECT "Time" FROM raw_data;
```

Now we create table SourceIP and insert data from raw data into it.

```
CREATE TABLE "SOURCEIP"(
   "Source_ID" Serial PRIMARY KEY,
```

```
  "Source" varchar NOT NULL
);
INSERT INTO "SOURCEIP" ("Source")
SELECT "Source" FROM raw_data;

Now we create table DestinationIP and insert data from raw data into it.
CREATE TABLE "DESTINATIONIP" (
 Destination_ID Serial PRIMARY KEY,
 Destination varchar NOT NULL
);
INSERT INTO "DESTINATIONIP" ("Destination")
SELECT "Destination" FROM raw_data;
Now we create table Packet and insert data from raw data into it.
CREATE TABLE "PACKET" (
  "Packet_ID" Serial PRIMARY KEY,
  "Length" int4 NOT NULL,
  "Info" varchar
);
INSERT INTO "PACKET" ("Length", "Info")
SELECT "Length", "Info" from raw_data;
Now we create table DOS and insert data from raw data into it.

CREATE TABLE "DOS" (
 "ID" Serial PRIMARY KEY,
 "Packet_ID" Serial NOT NULL,
 "Protocol_ID" Serial NOT NULL,
 "Time" float NOT NULL
);
INSERT INTO "DOS" ("Time")
Now, its time to add foreign keys.
SELECT "Time" from raw_data;
ALTER TABLE "DOS"
Adding foreign key Packet_ID that references Packet (Packet_ID)
ADD CONSTRAINT PIK FOREIGN KEY ("Packet_ID") REFERENCES "PACKET"("Packet_ID") MATCH
FULL;
ALTER TABLE "DOS"
Adding foreign key Protocol_ID that references Packet (Protocol_ID)
ADD CONSTRAINT PRIK FOREIGN KEY ("Protocol_ID") REFERENCES "PROTOCOL"("Protocol_ID")
MATCH FULL;
ALTER TABLE "DOS"
Adding foreign key Time that references Timestamp (Time)
ADD CONSTRAINT PRRIK FOREIGN KEY ("Time") REFERENCES "TIMESTAMP"("Time") MATCH FULL;
Selecting all rows from the raw_data
select * from "raw_data";
```

1.7 Views

Views are virtual table that is a result of a SQL query. We are using some of the entities from the table to get more understanding on some aspect of the dataset. In this case particularly, we are trying to find that transmission of packet from source to destination with its information and its length attribute. Views help in seeing data in new light.

We are creating view transmission_mk by selecting Packet_ID, Source, Destination, Info, Length from entities SourceIP, DestinationIP. Then we are connecting DestinationID and SourceID so that it's in sync with each other.

```
Query  Query History
1  CREATE VIEW transmission_mk AS
2  SELECT PACKET.Packet_ID,SOURCEIP.Source,DESTINATIONIP.Destination,PACKET.Info,PACKET.Length
3  FROM SOURCEIP,DESTINATIONIP,PACKET
4  WHERE DESTINATIONIP.Destination_ID=SOURCEIP.Source_ID ;
```

The view generated allows us to understand what packet number is travelling from which source carrying what info towards which destination.

packet_id integer	source character varying	destination character varying	info character varying	length integer
1	192.168.10.2	192.168.11.2	Query: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	78
2	192.168.10.2	192.168.11.2	Response: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	81
3	192.168.10.2	192.168.11.2	58032 > 502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478	66
4	192.168.10.2	192.168.11.2	Query: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	78
5	192.168.10.2	192.168.11.2	Response: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	81
6	192.168.10.2	192.168.11.2	51000 > 502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=1545815710 TSecr=2648963038	66
7	192.168.10.2	192.168.11.2	Query: Trans: 47892; Unit: 1, Func: 3: Read Holding Registers	78
8	192.168.10.2	192.168.11.2	Response: Trans: 47892; Unit: 1, Func: 3: Read Holding Registers	81
9	192.168.10.2	192.168.11.2	58032 > 502 [ACK] Seq=25 Ack=31 Win=229 Len=0 TSval=340602197 TSecr=1065398584	66
10	192.168.10.2	192.168.11.2	Query: Trans: 47891; Unit: 3, Func: 3: Read Holding Registers	78

1.8 Data Manipulation

Data manipulation refers to the insertion, deletion or updation of the data in the database by using operations through queries. This is done to make the data more readable and easy to access.

1.8.1 Data Manipulation: Internal Schema

Three operations used here are insert, delete and update on the model.

1. Using this query, we are inserting into source IP at the 11th index, 192.168.0.0 to add another IP address.

```
Query  Query History
1  INSERT INTO SOURCEIP
2  VALUES(11,'192.168.0.0');
```

2. Using this query, we are deleting from Packet entity whose ID is 3 since it's of not much significance and has dissipate during the DOS transmission.

```
Query  Query History
1  DELETE FROM PACKET
2  WHERE Packet_ID=3;
3
```

3. Using this query, we are updating SourceIP by setting the source to 111.111.11.1 where source_ID is 10 to change the IP .

Query | Query History

```
1  UPDATE SOURCEIP
2  SET SOURCE='111.111.11.1'
3  WHERE Source_ID=10;
```

1.8.2 Data Manipulation: External Schema (View)

1. Upon insertion, the row at 11th position in the table is now filled with new data.

source_id [PK] integer	source character varying
1	192.168.10.2
2	192.168.11.2
3	192.168.12.2
4	192.168.13.2
5	192.168.14.2
6	192.168.15.2
7	192.168.16.2
8	192.168.17.2
9	192.168.18.2
10	192.168.19.2
11	192.168.0.0

2. Upon deletion, the Packet_ID=3 row has been removed as shown.

packet_id [PK] integer	length integer	info character varying
1	78	Query: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers
2	81	Response: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers
4	78	Query: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers
5	81	Response: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers

3. Upon updation, the data row where Source_ID=10 has been updated to 111.111.11.1

source_id [PK] integer	source character varying
1	192.168.10.2
2	192.168.11.2
3	192.168.12.2
4	192.168.13.2
5	192.168.14.2
6	192.168.15.2
7	192.168.16.2
8	192.168.17.2
9	192.168.18.2
10	111.111.11.1

2 Individual Deliverables

This model is based on a real-time cyberattack with basic entities describing the attack environment for a common man to be able to understand the attack conditions. However, a few extensions can technically and logically improvise the database model to be able to identify the presence of a Denial-Of-Service attack without requirement of much knowledge about cybersecurity. The extensions polish the model in such a way that this new database model can be used to develop critically 'safe' softwares.

The following extensions have been added to the model:

Extension No.	*Name of Extension*	*Added By*
Extension 1	Source Port and Destination Port	Kanika Gandhi
Extension 2	Method	Devashish Singh Rajput
Extension 3	SYN and ACK	Karanpartap Singh Aulakh
Extension 4	Firewall	Shrutam Parmar

2.1 Extension 1: Source Port and Destination Port

Briefly, the source port and destination port has been used as an extension to better define the location of initiation and occurrence of the attack.

2.1.1 Business Rules

The extension of this model is based on the following business rules:

- The source port must be in the listening state to be able to allow 'Response' packets to pass through.
- The destination port must be in the listening mode to be able to allow 'Query' and 'Request' packets to be received.
- The source port and the destination port alongside the source IP and destination IP aid in recognizing from WHERE has the attack emerged and WHERE the attack has occurred.
- For the Denial Of Service, the source and destination ports will not affect the attack as even if a port is closed, the attack packet will pass through to the destination IP.
- For the Denial Of Service, the destination ports have to be either 53, 80 or 443, i.e the ports that open by default to compromise the destination.

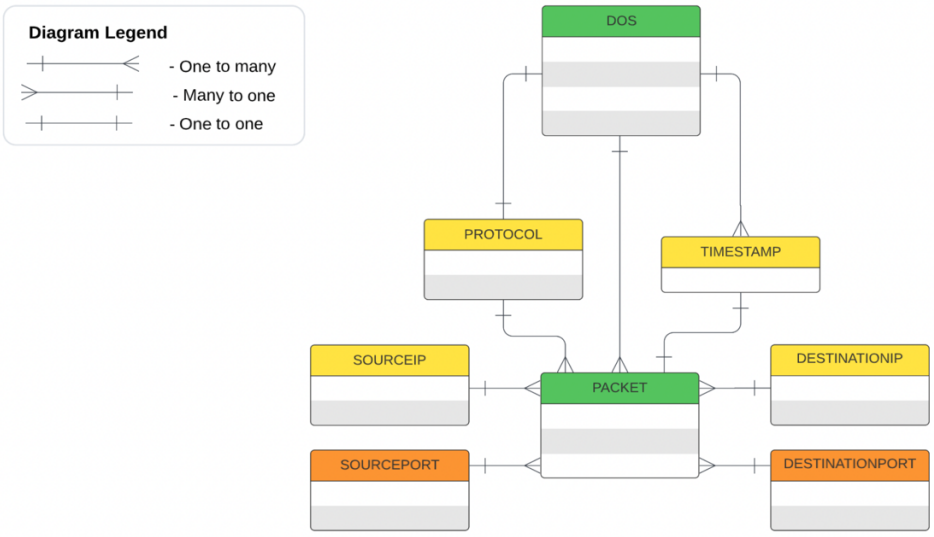
2.1.2 Conceptual Model

The conceptual model is based on the following relations:

- The SOURCEPORT and DESTINATIONPORT do not affect PROTOCOL with which the packets are transmitted, the TIMESTAMP at which the packets are sent, the DOS as the presence of any attack does not directly depend on the ports (with reference to business rule).
- The SOURCEPORT and DESTINATIONPORT are specific to the PACKET only as the 'SOURCEPORT+SOURCEIP' and 'DESTINATIONPORT+DESTINATIONIP' guide the packet to from the source to the destination.
- Since multiple packets will emerge from a single source, the SOURCEPORT has a one-to-many relationship with the PACKET.

- Similarly, multiple packets reach a single destination, the DESTINATIONPORT has a many-to-one relationship with PACKET.
- Lastly, since the SOURCEPORT and DESTINATIONPORT are not dependent on each other, there is no relation between both.

So, the conceptual model has turned out to be as follows:



2.1.3 Logical Model

The logical model is based on the following relations:

- Logical model is an extension of the conceptual model.
- In the logical model, the entities are DOS, Protocol, Timestamp, SourceIP, Packet, DestinationIP, SourcePort and DestinationPort.
- For DOS entity there are 4 attributes associated with it such as ID, Packet_ID, Protocol_ID and Time.
- For Protocol entity, the attributes are Protocol_ID and Protocol.
- For Timestamp, the attribute is Time.
- For SourceIP entity, we have Source_ID and Source for attributes. For DestinationIP entity we have Destination_ID and Destination as the attributes.
- Destinationport_ID and Destination Port are the attributes for the DestinationPort entity. For SourcePort entity, Sourceport_ID and Source Port are the attributes.
- Packet entity contains Packet_ID, Length and Info as the attributes.

2.1.3.1 Normalized Model

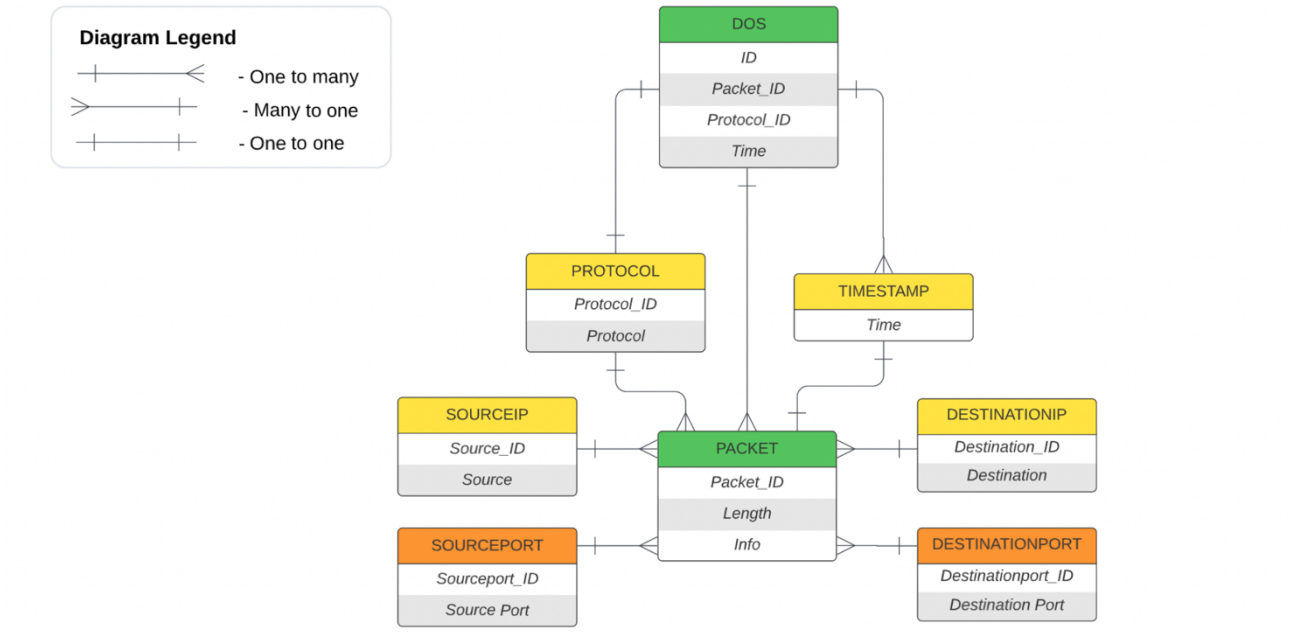
The logical model is already in the normalized form due to the following reasons based on the 3NF:

- 1NF: The first normal form is a description of the table that requires data in each column to be of atomic value only. In this model, the data in each column is atomic except the attribute 'INFO'. The data values in the 'INFO' are all linked together and cannot be segregated into sub parts

because upon separation, the data in 'INFO' holds no significance, and the packet is considered as non-existential.

- 2NF: The second normal form indicates getting rid of any kind of partial dependency. In this model, since they primary key after relation is Protocol_ID+Packet_ID+Time , there is no partial dependency of any other attribute in the model on any part of this primary key, singularly i.e either Protocol_ID or Packet_ID or Time.
- 3NF: The third normal form describes getting rid of any kind of transitive dependency. In this model, no attribute in the whole model is not dependent on any of the non-primary key.

Hence, the logical model is in the normalized form as shown below.

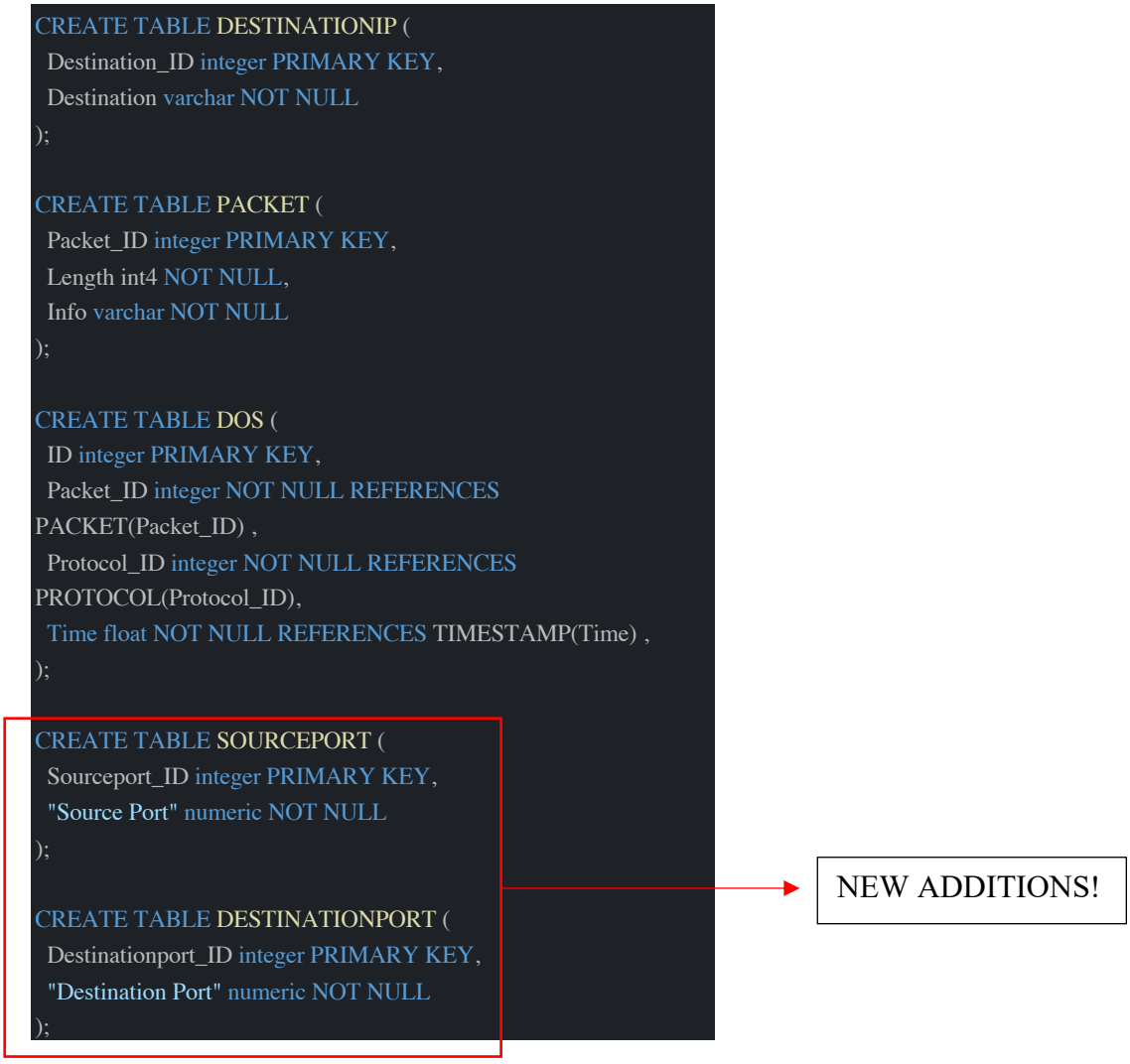


2.1.4 Physical Model: PostgreSQL

```
CREATE TABLE PROTOCOL (
  Protocol_ID integer PRIMARY KEY,
  Protocol varchar(255) NOT NULL
);

CREATE TABLE TIMESTAMP (
  Time float PRIMARY KEY
);

CREATE TABLE SOURCEIP (
  Source_ID integer PRIMARY KEY,
  Source varchar NOT NULL
);
```



The physical model has one main foundation it depicts the physical storage of data in the database.

The two main functions in this schema are:

1. Creation of the entity tables
2. Relation of the Foreign and Primary Keys

- SOURCEIP: This table constitutes of 2 attributes, namely 'Source-ID' and 'Source'. The ID is the primary key as it contains unique values that can be used to identify the IP addresses present in the 'Source' column.
- DESTINATIONIP: Similarly, his table constitutes of 2 attributes, namely 'Destination_ID' and 'Destination'. The ID is the primary key as it contains unique values that can be used to identify the IP addresses present in the 'Destination' column.
- SOURCEPORT: Further, this table also constitutes of 2 attributes, namely 'Sourceport_ID' and 'Source Port'. The ID is the primary key as it contains unique values that can be used to identify the IP addresses present in the 'Source Port' column.

- DESTINATIONPORT: As seen above, this table as well constitutes of 2 attributes, namely 'Destinationport_ID' and 'Destination Port'. The ID is the primary key as it contains unique values that can be used to identify the IP addresses present in the 'Destination Port' column.
- PROTOCOL: Likewise, this table as well constitutes of 2 attributes, namely 'Protocol_ID' and 'Protocol'. The ID is the primary key as it contains unique values that can be used to identify the protocol type present in the 'Protocol' column.
- PACKET: This table constitutes of 3 different attributes, namely, 'Packet_ID','Info' and 'Length'. As 'Packet_ID' is the primary key, it's unique values help to identify the complete packets. A complete packet carries routing information present in the 'Info' column in concatenation with the size of data being sent present in the 'Length' column.
- TIMESTAMP: This table only contains a single attribute, 'Time' ,as the value present in this table are already atomic and unique, hence it does not require a separate primary key. 'Time' contains the value of the microsecond at which a packet was, which can never be redundant in this table.
- DOS: This table constitutes of 4 different attributes, namely, 'ID', 'Packet_ID', 'Protocol_ID' and 'Time'. As 'ID' is the primary key, its unique values help to identify data present in columns. In this table, since the DOS attack directly depends on the PROTOCOL used and the information in PACKET, 'Packet_ID', 'Protocol,_ID', 'Time' are foreign keys that have been referenced from their original tables as they are primary keys in their respective tables.

2.1.5 Data Population: PostgreSQL

```
CREATE TABLE "SOURCEPORT" (
 Sourceport_ID Serial PRIMARY KEY,
  "Source Port" varchar NOT NULL
);
INSERT INTO "SOURCEPORT" ("Source Port")
SELECT "Source Port" FROM raw_data;

CREATE TABLE "DESTINATIONPORT" (
  Destinationport_ID Serial PRIMARY KEY,
  "Destination Port" varchar NOT NULL
);
INSERT INTO "DESTINATIONPORT" ("Destination Port")
SELECT "Destination Port" FROM raw_data;

select * from "raw_data";
```

Create a table called "SOURCEPORT"

Insert values into SOURCEPORT from the raw data table.

Create a table called "DESTINATIONPORT"

Insert values into SOURCEPORT from the raw data table.

Display all values from raw data

2.1.6 Views

In layman language, a view is a subset of a dataset which is a resultant table formed by running a query to combine two or more different entity tables.

Here, VIEW consists of data from the SOURCEIP, DESTINATIONIP, PACKET, SOURCEPORT & DESTINATIONPORT tables to be able to understand the transmission of a packet from a source

to a destination. With the 'Info' from packet, the type of packet travelling through can be identified alongside the size of the load ('Length') being transmitted from the specific source ('Source'+'Source Port') to a specific destination ('Destination'+ 'Destination Port'). The

Destination_ID from DESTINATIONIP has been equated with Source_ID from SOURCEIP to specific the number of rows that should be displayed in the view.

Query Query History

```
1
2  CREATE VIEW INFO AS
3  SELECT PACKET.PACKET_ID,SOURCEIP.SOURCE,SOURCEPORT."Source Port",PACKET.INFO,PACKET.LENGTH,DESTINATIONIP.DESTINATION,DESTINATIONPORT
4  FROM SOURCEIP,DESTINATIONIP,SOURCEPORT,DESTINATIONPORT,PACKET
5  WHERE DESTINATIONIP.DESTINATION_ID=SOURCEIP.SOURCE_ID;
```

packet_id integer	source character varying	Source Port numeric	info character varying	length integer	destination character varying	Destination Port numeric
10	192.168.10.2	54008	Query: Trans: 47891; Unit: 3, Func: 3: Read Holding Registers	66	192.168.11.2	53
9	192.168.10.2	54008	58032 > 502 [ACK] Seq=25 Ack=31 Win=229 Len=0 TSval=340602197 TSecr=1065398584	66	192.168.11.2	53
8	192.168.10.2	54008	Response: Trans: 47892; Unit: 1, Func: 3: Read Holding Registers	81	192.168.11.2	53
7	192.168.10.2	54008	Query: Trans: 47892; Unit: 1, Func: 3: Read Holding Registers	78	192.168.11.2	53
6	192.168.10.2	54008	51000 > 502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=1545815710 TSecr=2648963038	66	192.168.11.2	53
5	192.168.10.2	54008	Response: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	81	192.168.11.2	53
4	192.168.10.2	54008	Query: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	78	192.168.11.2	53
3	192.168.10.2	54008	58032 > 502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478	66	192.168.11.2	53
2	192.168.10.2	54008	Response: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	81	192.168.11.2	53
1	192.168.10.2	54008	Query: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	78	192.168.11.2	53

2.1.7 Data Manipulation

Data Manipulation means altering data in order to update it as per requirement, for ease of understanding, inserting or deleting any data as required. Sometimes insertion operations or updation operations are used to add or modify an entity that the user does not want to discard or wants to ads without manipulating the whole table. Deletion is an operation that is used in order to remove unwanted rows, redundancy or even extra duplicated rows.

2.1.7.1 Data Manipulation: Internal Schema

- INSERT: In this model, for the "DESTINATIONPORT" table, to modify the table by adding another IP address vulnerable to denial of service at the 11th position in the table without disturbing the arrangement of the existing columns, the output given below is obtained.

- DELETE: After insertion, for the "DESTINATIONPORT" table, to modify the table alongside addition, an unwanted IP address with Destinationport_ID=2 was not open for denial of service attack had to be removed from the database as it not being used.

- UPDATE: After deletion, for the "SOURCEPORT" table, to modify the table the port at the 10th position was not in listening mode and another port had been used to perform the DOS attack, so the data at Sourceport_ID=10 had to be updated.

Query Query History

```
1  INSERT INTO DESTINATIONPORT
2  VALUES (11,'11111');
3
4
5  DELETE FROM DESTINATIONPORT
6  WHERE Destinationport_ID=2;
7
8
9  UPDATE SOURCEPORT
10 SET "Source Port"=90
11 WHERE Sourceport_ID=10;
12
```

2.1.7.2 Data Manipulation: External Schema (View)

destinationport_id [PK] integer	Destination Port numeric
1	53
3	53
4	80
5	53
6	80
7	80
8	80
9	80
10	443
11	11111

DELETION

UPDATION

sourceport_id [PK] integer	Source Port numeric
1	54008
2	54110
3	54112
4	54114
5	54116
6	54118
7	54120
8	54122
9	54124
10	90

INSERTION

2.2 Extension 2: Method

2.2.1 Business Rules

When an HTTP client (such as a Web browser) communicates with an HTTP server (such as a Web server), it sends requests of various types, the most common of which are GET and POST. A GET request, which includes images, is used to retrieve a static piece of data, the URL pointing to that piece of data. A GET is performed when you enter a URL in the URL bar.

POST requests are used in combination with forms. A POST request includes parameters, which are typically taken from the same page's input fields.

When flooding, the attacker intends to overwhelm the target server with requests, saturating its computing resources. Flooding is most effective when the server allocates a large number of resources in response to a single request. Because POST requests include parameters, they usually result in relatively complex server processing, which is more expensive for the server than serving a much simpler GET. As a result, POST-based flooding is more effective than GET-based flooding (it takes fewer requests to drown the server if the requests are POST). However, because GET requests are far more common, it is often far easier for the attacker to enlist (unwilling) assistance in his flooding effort when GET-flooding.

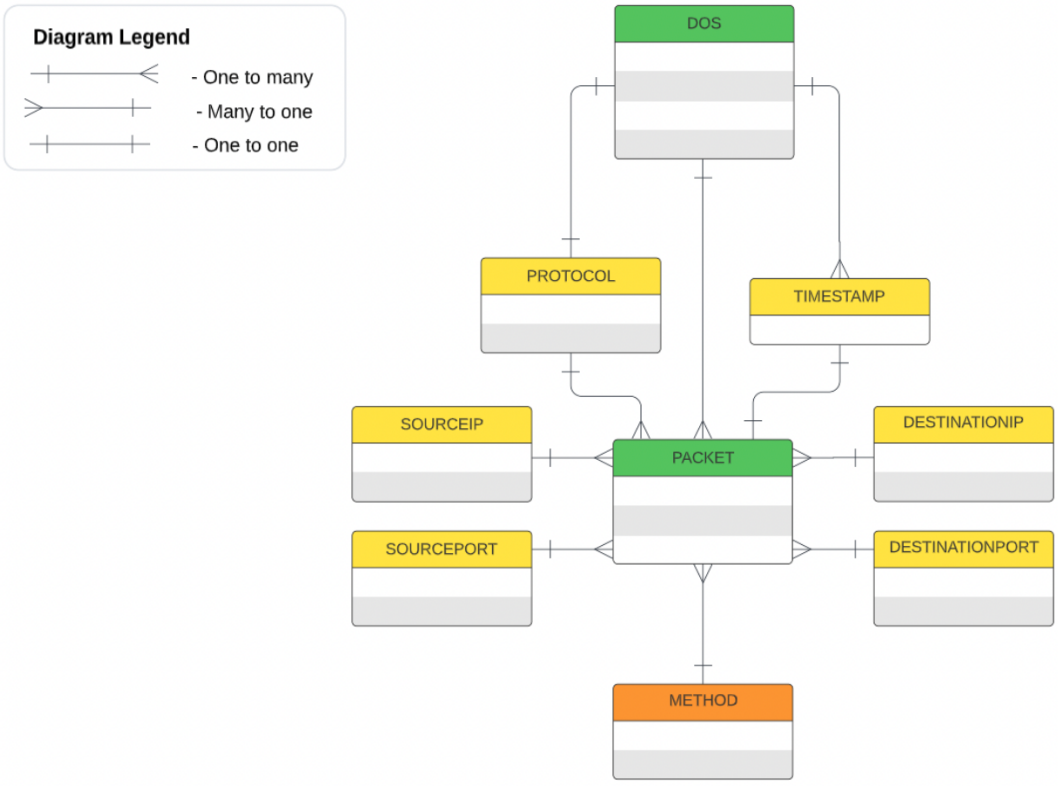
In our case, if the ACK (Acknowledgement) = 0 then it's a POST request and if ACK = 1, then it is a GET request.

2.2.2 Conceptual Model

The conceptual model defines the entities to be represented and the types of relationships that exist between them. It also defines the general rules that must be followed.
The original database tables PROTOCOL, TIMESTAMP, SOURCEIP, DESTINATIONIP, and PACKET are available, as are the extended tables METHOD, DOS, SOURCEPORT, and DESTINATION PORT.

Each table in this case is related to another table or tables. There are three kinds of relationships. one-to-one, many-to-one, and many-to-many When one record in table 1 is related to one or more records in table 2, this is referred to as a one-to-many relationship. In a database, a one-to-one relationship exists when each row in table 1 has only one related row in table 2. When multiple records in one table are related to multiple records in another table, this is referred to as a many-to-many relationship.

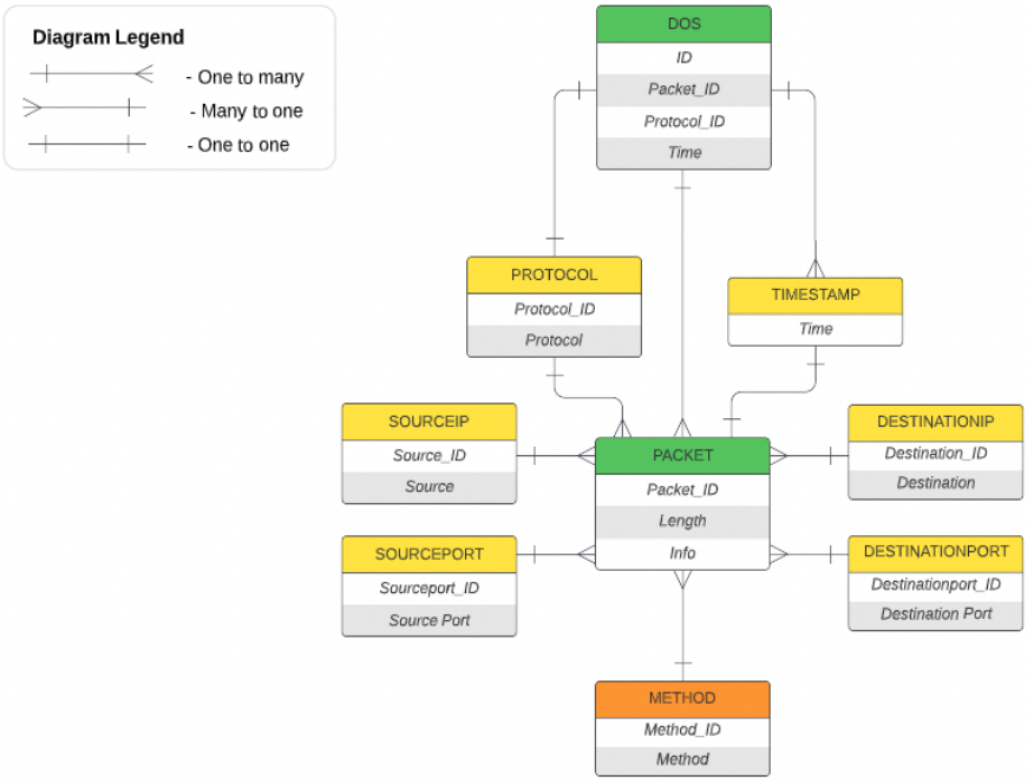
DOS-PROTOCOL and TIMESTAMP-PACKET have a one-to-one relationship in the given model. DOS-TIMESTAMP, DOS-PACKET, PROTOCOL-PACKET, SOUREIP-PACKET, SOURCEPORT-PACKET, METHOD-PACKET, DESTINATIONPORT-PACKET, DESTINATIONIP-PACKET all have a one-to-many relationship.



2.2.3 Logical Model

Logical data modelling is the process of defining the business entities that will eventually become tables, their attributes, and how the entities will interact with one another.
The logical model, as opposed to the conceptual model, displays attributes.
The primary keys in this table are Protocol_ID from the Protocol table, Time from the TIMESTAMP, Source ID from SOURCEIP, Destination_ID from DESTINATIONIP, Packet_ID from the packet, ID from DOS, Sourceport_ID from SOURCEPORT, and Destinationport_ID from DESTINATIONPORT.

Foreign keys defined in the DOS table are also defined. DOS Packet ID refers to PACKET Packet_ID, and Protocol_ID refers to Protocol_ID.
The method table is the only one without a primary or foreign key.



2.2.3.1 Normalized Model

Tables should not contain multiple values in a single cell for the 1NF. In this case, each cell in each table has only one value. As a result, it is in 1st Normal Form. Tables should not have partial dependencies for the 2NF. A non-prime attribute is determined by the appropriate subset of candidate keys in a partial dependency. Non-prime attributes in each table are completely dependent on a single primary key. As a result, the tables contain no partial dependencies. That is why it is written in the second normal form. There should be no transitive dependency for the 3NF. It means that non-prime attributes in a given table should not be dependent on other non-prime attributes. All non-prime attributes in each table are either dependent on the prime attributes or are a single entity. As a result, there is no transitive dependency. As a result, it is in 3rd Normal Form.

2.2.4 Physical Model: PostgreSQL

Tables for Protocol, TIMESTAMP, SOURCEIP, DESTINATION IP, DOS, and PACKET are included in the overall group project. First, a table called Protocol with the columns Protocol_ID and Protocol was created, with Protocol_ID serving as the table's primary key. The TIMESTAMP table is then created using Time as the primary key and column. There are two tables after that: DESTINATIONIP, which has columns like Destination_ID, Destination, and SOURCEIP, which has columns like Source_ID, Source. The DOS table is then produced. Columns like ID, Packet_ID, Protocol_ID, and Time are present. Packet_ID and Protocol_ID are foreign keys in the DOS table, and ID is the primary key (Key creation is done after populating the data). The following table, PACKET, has the columns Packet_ID, Length, and Info.

The METHOD table is an extended table with the columns METHOD_ID and METHOD in it.

```
SET search_path TO 'public';

CREATE TABLE "raw_data" (
 "No." int4,
 "Time" float,
 "source" varchar,
 "Destination" varchar,
 "Protocol" varchar,
 "Length" int4,
 "info" varchar,
 "SYN" int4,
 "ACK" int4,
 "METHOD" varchar,
 "Source Port" int4,
 "Destination Port" int4,
 PRIMARY KEY ("No.")
);
select * from raw_data;

CREATE TABLE "Protocol" (
  "Protocol_ID" Serial,
  "Protocol" varchar,
PRIMARY KEY ("Protocol_ID")
);

CREATE TABLE "TIMESTAMP"(
  Time float PRIMARY KEY
);

CREATE TABLE "SOURCEIP"(
        "Source_ID" Serial PRIMARY KEY,
        "Source" varchar NOT NULL
);

CREATE TABLE "DESTINATIONIP" (
  Destination_ID Serial PRIMARY KEY,
  Destination varchar NOT NULL
);

CREATE TABLE "PACKET" (
        "Packet_ID" Serial PRIMARY KEY,
        "Length" int4 NOT NULL,
        "Info" varchar
);

CREATE TABLE "DOS" (
  "ID" Serial PRIMARY KEY,
```

```
  "Packet_ID" Serial NOT NULL,
  "Protocol_ID" Serial NOT NULL,
  "Time" float NOT NULL
);

CREATE TABLE "METHOD" (
  "METHOD_ID" Serial PRIMARY KEY,
  "METHOD" varchar NOT NULL
);
```

2.2.5 Data Population: PostgreSQL

Query | Query History

```
1  insert into "METHOD" ("METHOD")
2
3  select "METHOD" from raw_data;
```

METHOD_ID [PK] integer	METHOD character varying
1	POST
2	POST
3	GET
4	POST
5	POST
6	GET
7	POST
8	POST
9	GET
10	POST

2.2.6 Views

SQL has a feature called view. It enables the creation of virtual tables based on SQL queries that make use of other database tables. Here, we used "CREATE VIEW" to create a view/virtual table called METHOD DETECTION. We chose one column from each of the four tables we chose from the database to create this. Source from SOURCEIP, destination from DESTINATIONIP, Protocol from Protocol, and METHOD from METHOD are the selected columns, as shown below. These 4 columns are combined into a single table to create the view.

```
CREATE VIEW "METHOD DETECTION" AS
SELECT "SOURCEIP"."Source", "DESTINATIONIP"."destination", "Protocol"."Protocol",
"METHOD"."METHOD"
FROM "SOURCEIP", "DESTINATIONIP", "Protocol", "METHOD"
```

	Source character varying	destination character varying	Protocol character varying	METHOD character varying
1	192.168.10.2	192.168.11.2	Modbus/TCP	POST
2	192.168.10.2	192.168.11.2	Modbus/TCP	POST
3	192.168.10.2	192.168.11.2	Modbus/TCP	GET
4	192.168.10.2	192.168.11.2	Modbus/TCP	POST
5	192.168.10.2	192.168.11.2	Modbus/TCP	POST
6	192.168.10.2	192.168.11.2	Modbus/TCP	GET
7	192.168.10.2	192.168.11.2	Modbus/TCP	POST
8	192.168.10.2	192.168.11.2	Modbus/TCP	POST
9	192.168.10.2	192.168.11.2	Modbus/TCP	GET
10	192.168.10.2	192.168.11.2	Modbus/TCP	POST
11	192.168.10.2	192.168.11.2	Modbus/TCP	POST
12	192.168.10.2	192.168.11.2	Modbus/TCP	GET
13	192.168.10.2	192.168.11.2	Modbus/TCP	POST
14	192.168.10.2	192.168.11.2	Modbus/TCP	POST
15	192.168.10.2	192.168.11.2	Modbus/TCP	GET

2.2.7 Data Manipulation

To query and alter database data, one uses the SQL data manipulation language (DML).
SELECT is a database querying tool.
INSERT: the act of adding data to a table
UPDATE: to modify a table's data
DELETE: to remove information from a table.

2.2.7.1 Data Manipulation: Internal Schema

Query Query History

```
1  INSERT INTO sourceip
2  VALUES(21,'192.169.0.2');
```

Query Query History

```
1  DELETE FROM "METHOD"
2  WHERE "METHOD_ID"= 110;
```

Query Query History

```
1  UPDATE destinationip
2  SET destination = '192.168.13.2'
3  WHERE destination_id = 5;
```

2.2.7.2 Data Manipulation: External Schema (View)

INSERTION

INSERT INTO is used to add any value which we want to add in a table, like in the image below, we inserted the source_id = 21 and source = 192.169.0.2

	source_id [PK] integer	source character varying
1	1	192.168.10.2
2	2	192.168.11.2
3	3	192.168.12.2
4	4	192.168.13.2
5	5	192.168.14.2
6	6	192.168.15.2
7	7	192.168.16.2
8	8	192.168.17.2
9	9	192.168.18.2
10	10	111.111.11.1
11	21	192.169.0.2

DELETION

DELETE is used to remove any value from a table which we want to delete. Here, we have deleted METHOD_ID = 110.

METHOD_ID [PK] integer	METHOD character varying
105	POST
106	POST
107	GET
108	POST
109	POST
111	POST
112	POST
113	GET
114	POST
115	POST
116	GET
117	POST

UPDATION

UPDATE is used to modify any existing record which is present inside out table. Here, we have updated the destination IP address where the “destination_id” = 5.

Before updating the value of “destination_id” = 5

	destination_id [PK] integer	destination character varying
1	1	192.168.11.2
2	2	192.168.10.2
3	3	192.168.11.2
4	4	192.168.12.2
5	5	192.168.10.2
6	6	192.168.12.2
7	7	192.168.11.2
8	8	192.168.10.2
9	9	192.168.12.2
10	10	192.168.11.2

After Updating the value of “destination_id” = 5

	destination_id [PK] integer	destination character varying
1	1	192.168.11.2
2	2	192.168.10.2
3	3	192.168.11.2
4	4	192.168.12.2
5	5	192.168.13.2
6	6	192.168.12.2
7	7	192.168.11.2
8	8	192.168.10.2

2.3 Extension 3: SYN & ACK

Denial of Service attack is a packet and transmission attack which is highly dependent on the packet load and SYN-ACK connection. The SYN-ACK connection is the simple process of a source trying to connect with a destination. A source sends out a SYN request to the destination which the latter upon acceptance accepts and sends out and ACK response in return. In the case of a Denial Of Service attack, multiple sync packets are sent out but due to unmanageable load, the destination is unable to send out the equal amount of ACKS, hence compromising the connection between the two. This extension is a major detection method used to identify and detect at a very early stage, the presence of DOS.

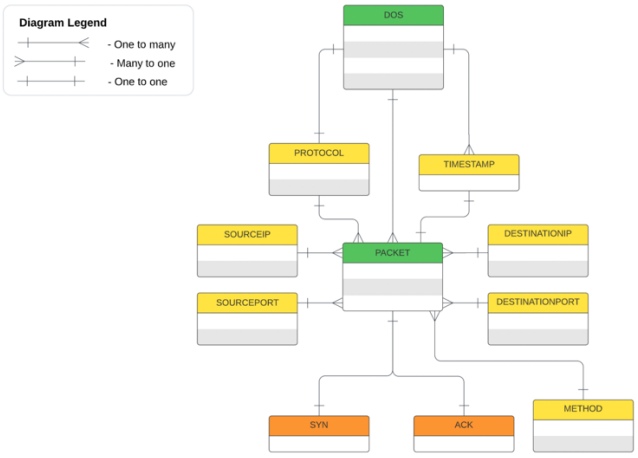
2.3.1 Business Rules

- For a successful TCP connection to be established a successful SYN-ACK transmission must be completed.
- For every Source IP to be able send out an 'ACK' packet a related 'SYN' request packet must be transmitted to the same IP beforehand.
- Source must send a lot of SYN packets to the destination in a short period of time to carry out Denial Of Service.
- Destination must be overwhelmed with SYN requests so that it cannot send out ACK to all SYN requests for an attack to be considered active.

2.3.2 Conceptual Model

The conceptual model is the high-level representation of the system and helps users to understand the system more accurately. Conceptual model has entities and relationships. The model has entities like DOS (Denial Of Service), Protocol, Timestamp, Packet, SourceIP, Sourceport, DestinationIP, Destinationport, SYN, ACK and Method. All the entities are related to each other through relationships which can be one to one, one to many, many to one and many to many.

The extensions that I have used for the conceptual model is SYN and ACK. SYN is the Synchronous packet and the ACK is the acknowledgement packet. SYN and ACK are in relation with the packet with one to one relationship. This can be thought of as packet can either be of type SYN or ACK. The extension entities i.e., SYN and ACK are neither related nor dependent on any other entity in the data hence they are not shown to be in one with any other entity. For example, SYN and ACK packets don't depend on neither SourcePort, DestinationPort nor SourceIP and DestinationIP. SYN and ACK should be present no matter what the protocol is or what the time might be hence there's no dependency with protocol and Timestamp entities either.



2.3.3 Logical Model

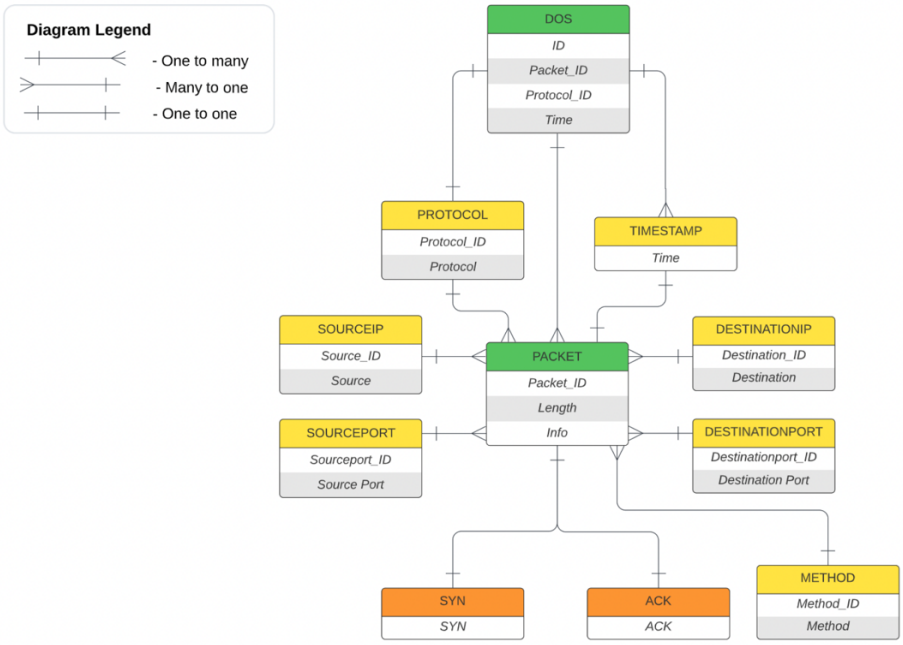
Logical model is the way to describe entities, their attributes, and the relationship between them. In logical model, we define primary and foreign keys. We know from the model that our primary key is a composite key consisting of Packet_ID, Protocol_ID and Time. Protocol ID, Packet ID and Time are Foreign keys in the DOS entity table.

2.3.3.1 Normalized Model

The extensions I have chosen are SYN and ACK. SYN is the synchronous packet and ACK stands for asynchronous packets. The packet can be either of SYN or ACK type hence has a relationship with the Packet entity. After creating the logical model, we must normalize the model in order to make sure that there are no redundant values or attributes in the whole dataset. There are three stages to normalize a model namely 1NF, 2NF and 3NF. In 1NF, we must make sure that the values in all the tables need to be atomic in nature. The extensions I have used are SYN and ACK, since they don't have any columns with non- atomic values hence the whole model is in 1 NF already.

While normalizing the logical model to 2NF, we must eliminate any partial dependency between the tables. Since SYN and ACK have no attribute other than SYN and ACK, there are no partial dependencies to other attributes of the entities in the whole model. Hence, the model has passed the 2NF normalization state also.

Now moving forward to normalize the model to 3NF, we get rid of the transient dependency. No attribute of any entity should be dependent on non-primary key attributes. For example, SYN and ACK is not dependent on any other attribute other than Packet_ID, Protocol_ID and Time (Primary keys) hence no transient dependency could be found. Thus, the logical model is in 3NF.



2.3.4 Physical Model: PostgreSQL

We created a table PROTOCOL which consists of 2 attributes 'Protocol_ID' and 'Protocol' with 'Protocol_ID' as the primary key which is unique for each protocol. Moving on, we created a table named TIMESTAMP which contains only a single attribute 'Time' which also serves as the primary key as it contains unique values against each packet in the dataset.

Next, we created the table SOURCEIP with 'Source' and 'Source_ID' as two attributes and 'Source_ID' serving as the primary key which is unique for each IP address in the dataset. Similarly, we created the table DESTINATIONIP with 2 attributes, namely 'Destination_ID' and 'Destination'. 'Destination_ID' is the primary key as it is unique for each IP address in the dataset.

We created a table PACKET with 3 attributes 'Packet_ID', 'Info' and 'Length'. 'Packet_ID' is the primary key as it is a unique value for each packet. 'Info' contains the routing information and 'Length' contains the size of data being sent.

Next we created a table DOS with 4 different attributes, namely, 'ID', 'Packet_ID', 'Protocol_ID' and 'Time'. 'ID' serves as the primary key that helps to identify data present in columns with its unique values. 'Packet_ID', 'Protocol_ID', 'Time' are foreign keys since DOS attack directly depends on the PROTOCOL used and the information in PACKET. These foreign keys have been referenced from their original tables as they are primary keys in their respective tables.

Lastly, we created the SYN and ACK tables with SYN and ACK as their only attributes respectively.

Query | Query History

```
1  CREATE TABLE PROTOCOL (
2    Protocol_ID integer PRIMARY KEY,
3    Protocol varchar(255) NOT NULL
4  );
5
6  CREATE TABLE TIMESTAMP (
7    Time float PRIMARY KEY
8  );
9
10 CREATE TABLE SOURCEIP (
11   Source_ID integer PRIMARY KEY,
12   Source varchar NOT NULL
13 );
14
15 CREATE TABLE DESTINATIONIP (
16   Destination_ID integer PRIMARY KEY,
17   Destination varchar NOT NULL
18 );
19
20 CREATE TABLE PACKET (
21   Packet_ID integer PRIMARY KEY,
22   Length int4 NOT NULL,
23   Info varchar NOT NULL
24 );
```

Query | Query History

```
  Length int4 NOT NULL,
  Info varchar NOT NULL
);

CREATE TABLE DOS (
  ID integer PRIMARY KEY,
  Packet_ID integer NOT NULL REFERENCES PACKET(Packet_ID) ,
  Protocol_ID integer NOT NULL REFERENCES PROTOCOL(Protocol_ID),
  Time float NOT NULL REFERENCES TIMESTAMP(Time)
);

CREATE TABLE SOURCEPORT (
  Sourceport_ID integer PRIMARY KEY,
  "Source Port" numeric NOT NULL
);

CREATE TABLE DESTINATIONPORT (
  Destinationport_ID integer PRIMARY KEY,
  "Destination Port" numeric NOT NULL
);

CREATE TABLE METHOD(
  Method_ID integer PRIMARY KEY,
  Method varchar NOT NULL
);

CREATE TABLE SYN(
SYN integer PRIMARY KEY
);

CREATE TABLE ACK(
 ACK integer PRIMARY KEY
);
```

2.3.5 Data Population: PostgreSQL

```
CREATE TABLE "SYN"(                    /*Create a Table named SYN with data type int4*/
      "SYN" int4
);
INSERT INTO "SYN" ("SYN")    /*By selecting data from the raw data table, */

SELECT "SYN" FROM raw_Data;  /* Data is pushed into the SYN column and  */
select * from "SYN"                /* all the rows are displayed  */

CREATE TABLE "ACK"(  /*Create a Table named ACK with data type int4*/
      "ACK" int4
);
INSERT INTO "ACK" ("ACK")            /*By selecting data from the raw data table, */
SELECT "ACK" FROM raw_Data;         /* Data is pushed into the ACK column and  */
select * from "ACK";                /* all the rows are displayed  */
select * from "raw_data"
```

syn integer	ack integer
0	0
0	0
1	1
0	0
0	0
1	1
0	0
0	0
1	1
0	0

2.3.6 Views

Views in a database management system is a combination or set of previously existing columns that are put together using a specific query to provide a perspective/view of viewing data in a virtual manner.

Here, for understanding better, the Packet_ID, Source Port, Info, Length, SYN, ACK, Destination Port has been put together using the CREAT VIEW query to understand whether or not a packet with ID=1 is an acknowledge (ACK) packet, in respect to it's source and destination address.

Query | Query History

```
1
2  CREATE VIEW YU AS
3  SELECT PACKET.PACKET_ID,SOURCEPORT."Source Port",PACKET.INFO,PACKET.LENGTH,SYN.SYN,ACK.ACK,DESTINATIONPORT."Destination Port"
4  FROM SOURCEPORT,SYN,ACK,DESTINATIONPORT,PACKET
5  WHERE PACKET_ID=1;
6
```

packet_id integer	Source Port numeric	info character varying	length integer	syn integer	ack integer	Destination Port numeric
1	54008	Query: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	78	0	1	53

2.3.7 Data Manipulation

A combination, subset or singularly, when operations such as insert, delete and update are used on any data in the database, it is knows as data manipulation. There are many reasons as to why a database might be manipulated, a few of them being:

1. Trend Analysis.
2. Analysis of financial, behavioral data.
3. Ease in reading and accessing.

2.3.7.1 Data Manipulation: Internal Schema

For data manipulation, insert operation has been used here to modify the SYN value table to add in another entry about a packet containing information for synchronous connection request.

For data manipulation, delete operation has been used here to modify the SYN value table to remove out all the SYN=1 packets from the table.

For data manipulation, update operation has been used here to modify the ACK value table to change the number of ACK response packets present to 1 by setting ACK=1.

Query | Query History

```
1
2  INSERT INTO SYN
3  VALUES (11,0);
4
5
6  DELETE FROM SYN
7  WHERE SYN=1;
8
9
10 UPDATE ACK
11 SET ACK=1
12 WHERE ACK=0;
13
```

2.3.7.2 Data Manipulation: External Schema (View)

syn integer	ack integer
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1

INSERTION: An insertion has been made in the 11th position where SYN=0

DELETION: Any values having SYN=1 are eliminated

UPDATION: All values in ACK are set to 1

2.4 Extension 4: Firewall

2.4.1 Business Rules

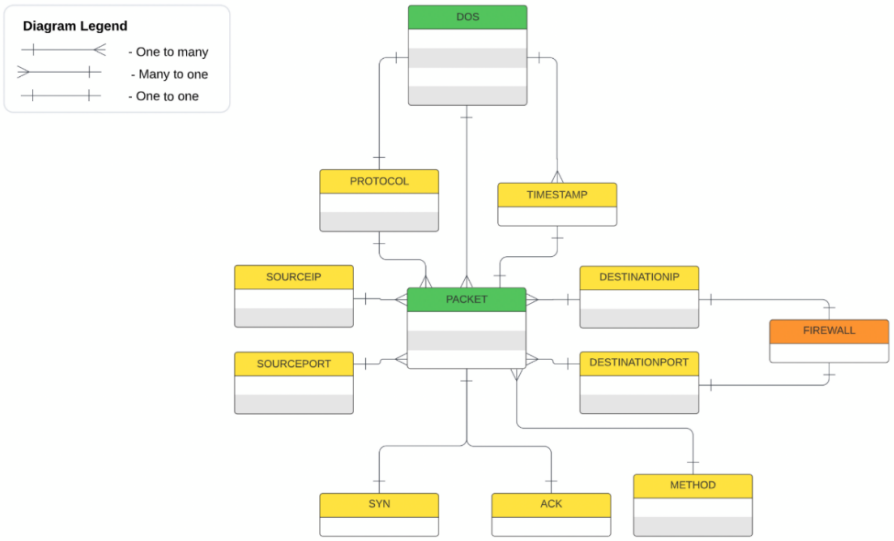
DoS attack protection allow or deny all connection attempts that require crossing an interface on their way to and from the intended destination. With the maximum number of people being online, it has become very crucial for each and every user to implement some kind of cyber security tool on their devices and network to protect against various harmful cyber-attacks. There are few tools like Firewall that can be really helpful in bifurcating legitimate and malicious traffic from the network. When the firewall detects a DoS attack, it applies traffic limits for 10 seconds. After this time, if the DoS attack stops, the firewall removes the traffic limits. If the DoS attack continues, the counter is reset every 10 seconds on a rolling basis, and the firewall keeps the traffic limits. It means that if we get the acknowledgement then the firewall is present and if we don't get the acknowledgement then there is no firewall.

2.4.2 Conceptual Model

The conceptual model sketches out the entities to be represented and determines what kinds of relationships exist between them. It also defines the general rules that need to be considered.

As given below, we have the original database tables PROTOCOL, TIMESTAMP, SOUR CEIP, DESTINATIONIP, PACKET and the extended tables SYN, ACK, FIREWALL, METHOD, DOS, SOURCEPORT, DESTINATION PORT.

Here, each table has relation to other table/tables. Relations are of three types one-to-one, one-to-many, many-to-one. one-to-many relationship occurs when one record in table 1 is related to one or more records in table 2. one-to-one relationship in a database occurs when each row in table 1 has only one related row in table 2. many-to-many relationship occurs when multiple records in one table are related to multiple records in another table.



In the given model, one-to-one relationship can be seen between DOS-PROTOCOL, SYN-PACKET, ACK-PACKET, FIREWALL-DESTINATIONPORT, FIREWALL-DESTINATIONIP, TIMESTAMP-PACKET. one-to-many relationship can be seen between DOS-TIMESTAMP, DOS-PACKET, PROTOCOL-PACKET, SOUREIP-PACKET, SOURCEPORT-PACKET, METHOD-PACKET, DESTINATIONPORT-PACKET, DESTINATIONIP-PACKET.

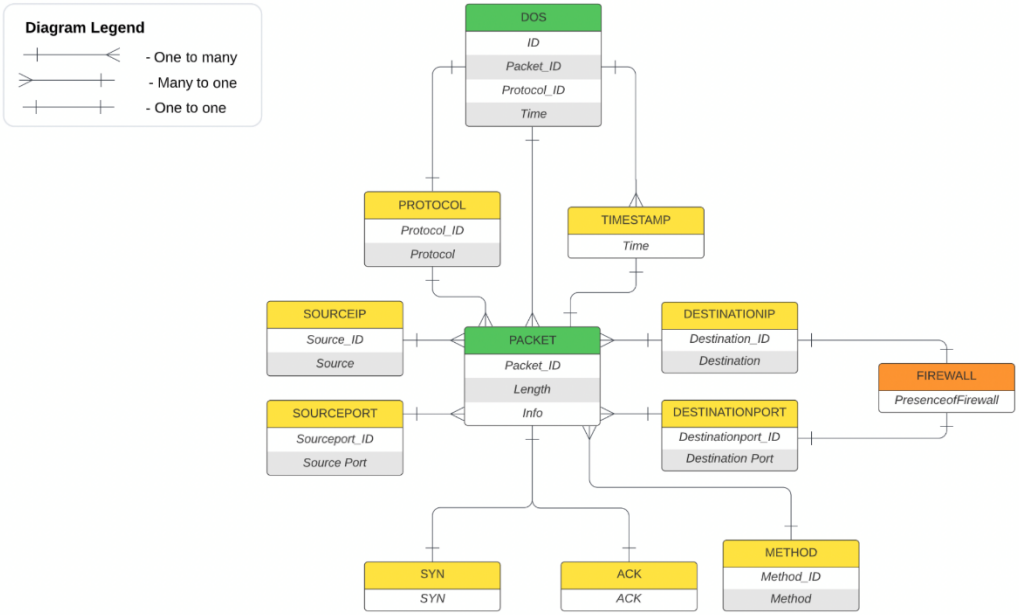
2.4.3 Logical Model

Logical data modeling is where you will define the business entities which will eventually become tables, it's attributes and how the entities inter-relate with each other.

Compared to conceptual model, attributes are displayed in the logical model.

Here, primary keys are Protocol_ID from Protocol table, Time from TIMESTAMP, Source_ID from SOURCEIP, Destination_ID from DESTINATIONIP, Packet_ID from packet, ID from DOS, Sourceport_ID from SOURCEPORT, Destinationport_ID from DESTINATIONPORT.

Foreign keys are also defined which are defined in the DOS table. Packet_Id from DOS is referenced to Packet_Id from PACKET, Protocol_Id is referenced to Protocol_Id from Protocol.



2.4.3.1 Normalized Model

For the 1NF, tables should not contain multiple values in a single cell. Here, all the tables are having only one value in each cell. So, it is in 1st Normal Form. For the 2NF, tables should not have partial dependency. Partial dependency means the proper subset of candidate key determines a non-prime attribute. Here, in each table, non-prime attributes are fully dependent on a single primary key. So, there is no partial dependency in the tables. That is why it is in 2nd Normal Form. For the 3NF, there should be no transitive dependency. It means non-prime attributes should not be dependent on other non-prime

attributes in a given table. Here, in each table, all non-prime attributes are dependent on the prime attributes or they are a single entity. Hence, there is no transitive dependency. So, it is in 3rd Normal Form.

2.4.4 Physical Model: PostgreSQL

The general group work contains tables Protocol, TIMESTAMP, SOURCEIP, DESTINATION IP, DOS and PACKET. First, we created a table called Protocol with Protocol_ID and Protocol columns where Protocol_ID is the primary key. Then Having Time as a column and primary key, the TIMESTAMP table is generated. After that there is a SOURCEIP table which has columns like Source_ID, Source and a DESTINATIONIP table which has columns like Destination_ID, Destination. Then DOS table is created. It is having columns such as ID, Packet_ID, Protocol_ID and Time. In the DOS table ID is the primary key and Packet_ID, Protocol_ID are foreign keys (Key creation is done after populating the data). Then there is a table PACKET having Packet_ID, Length, Info as columns.

The FIREWALL table is the extended table. It is having column called PresenceofFirewall.

```
SET search_path TO 'public';
CREATE TABLE "Protocol" (
  "Protocol_ID" Serial,
  "Protocol" varchar,
PRIMARY KEY ("Protocol_ID")
);

CREATE TABLE "TIMESTAMP"(
  Time float PRIMARY KEY
);

CREATE TABLE "SOURCEIP"(
        "Source_ID" Serial PRIMARY KEY,
        "Source" varchar NOT NULL
);

CREATE TABLE "DESTINATIONIP" (
  Destination_ID Serial PRIMARY KEY,
  Destination varchar NOT NULL
);

CREATE TABLE "DOS" (
  "ID" Serial PRIMARY KEY,
  "Packet_ID" Serial NOT NULL,
  "Protocol_ID" Serial NOT NULL,
  "Time" float NOT NULL
);

CREATE TABLE "PACKET" (
        "Packet_ID" Serial PRIMARY KEY,
        "Length" int4 NOT NULL,
```

```
        "Info" varchar
);

CREATE TABLE "SYN"(
        "SYN" int4
);

CREATE TABLE "ACK"(
        "ACK" int4
);

CREATE TABLE "FIREWALL" (
        "PresenceofFirewall" varchar);
```

2.4.5 Data Population: PostgreSQL

```
CREATE TABLE "raw_data" (
 "No." int4,
 "Time" float,
 "source" varchar,
 "Destination" varchar,
 "Protocol" varchar,
 "Length" int4,
 "info" varchar,
 "SYN" int4,
 "ACK" int4,
 "FIREWALL" varchar,
 "METHOD" varchar,
 "ANOMALY DETECTION" varchar,
 "Source Port" int4,
 "Destination Port" int4,
 PRIMARY KEY ("No.")
);

copy raw_data
from ' C:\Users\ROG\Downloads\Western University\Fall 2022\Data Management and Applications\neww.csv'
DELIMITER ',' CSV HEADER;

INSERT INTO "Protocol"("Protocol")
SELECT "Protocol" FROM raw_data;

INSERT INTO "TIMESTAMP" ("time")
SELECT "Time" FROM raw_data;

INSERT INTO "SOURCEIP" ("Source")
SELECT "source" FROM raw_data;

INSERT INTO "DESTINATIONIP" ("destination")
SELECT "Destination" FROM raw_data;
```

```
INSERT INTO "PACKET" ("Length", "Info")
SELECT "Length", "info" from raw_data;

INSERT INTO "DOS" ("Time")
SELECT "Time" from raw_data;

ALTER TABLE "DOS"
ADD CONSTRAINT PIK FOREIGN KEY ("Packet_ID") REFERENCES
"PACKET"("Packet_ID") MATCH FULL;

ALTER TABLE "DOS"
ADD CONSTRAINT PRIK FOREIGN KEY ("Protocol_ID") REFERENCES
"Protocol"("Protocol_ID") MATCH FULL;

INSERT INTO "SYN" ("SYN")
SELECT "SYN" FROM raw_Data;

INSERT INTO "ACK" ("ACK")
SELECT "ACK" FROM raw_Data;

INSERT INTO "FIREWALL" ("PresenceofFirewall")
SELECT "FIREWALL" FROM raw_Data;
```

2.4.6 Views

View is a feature of SQL. It allows one to create a virtual table based on SQL query referring to other tables in the database. Here, we created a view/virtual table called Prevention using "CREATE VIEW". To create this, we selected 4 tables from the database and 1 column from each table. As given below, selected columns are Source from SOURCEIP, destination from DESTINATIONIP, Protocol from Protocol and PresenceofFirewall from FIREWALL. The view is created by combining these 4 columns into one separate table.

```
    CREATE VIEW "Prevention" AS
SELECT "SOURCEIP"."Source", "DESTINATIONIP"."destination", "Protocol"."Protocol",
"FIREWALL"."PresenceofFirewall"
FROM "SOURCEIP", "DESTINATIONIP", "Protocol", "FIREWALL"
```

	Source character varying	destination character varying	Protocol character varying	PresenceofFirewall character varying
1	192.168.10.2	192.168.11.2	Modbus/TCP	N
2	192.168.10.2	192.168.11.2	Modbus/TCP	N
3	192.168.10.2	192.168.11.2	Modbus/TCP	Y
4	192.168.10.2	192.168.11.2	Modbus/TCP	N
5	192.168.10.2	192.168.11.2	Modbus/TCP	N
6	192.168.10.2	192.168.11.2	Modbus/TCP	Y
7	192.168.10.2	192.168.11.2	Modbus/TCP	N
8	192.168.10.2	192.168.11.2	Modbus/TCP	N
9	192.168.10.2	192.168.11.2	Modbus/TCP	Y
10	192.168.10.2	192.168.11.2	Modbus/TCP	N

2.4.7 Data Manipulation

Data manipulation is used for adding, deleting and modifying data in a database. It is very helpful to organize or arrange the kind of structured data. It can improve the quality of the data. For this there are some SQL commands.

To insert rows into the table we can use “INSERT” command, to delete particular elements we can use “DELETE” command, to update the existing data we can use “UPDATE” command, to select the data to manipulate we can use the “SELECT” command.

2.4.7.1 Data Manipulation: Internal Schema

DELETE

```
DELETE FROM "DESTINATIONIP" WHERE "destination_id"=5
SELECT * FROM "DESTINATIONIP"
```

INSERT

```
INSERT INTO "DESTINATIONIP" VALUES(5, '192.168.10.2')
SELECT * FROM "DESTINATIONIP"
```

UPDATE

```
UPDATE "DESTINATIONIP"
SET "destination"='192.168.220.160'
WHERE "destination_id"=5
SELECT * FROM "DESTINATIONIP"
```

2.4.7.2 Data Manipulation: External Schema (View)

DELETE

Delete is used to remove particular elements from the table or we can say that it removes tuples from a relation. Here, we deleted row with destination_id=5.

	destination_id [PK] integer	destination character varying
1	1	192.168.11.2
2	2	192.168.10.2
3	3	192.168.11.2
4	4	192.168.12.2
5	6	192.168.12.2

INSERT

Insert is used to add one or more rows to the table. Attribute values should be listed in the same order as the attributes were specified in the CREATE TABLE. Here, we added row with destination_id=5.

	destination_id [PK] integer	destination character varying
57919	57920	192.168.10.2
57920	57921	192.168.10.2
57921	57922	192.168.10.2
57922	57923	192.168.10.2
57923	5	192.168.10.2

UPDATE

By using UPDATE, we can change the existing data in a table. Here, we changed the destination value at destination_id=5. Before updating the value was 192.168.10.2 and now it is 192.168.220.160.

	destination_id [PK] integer	destination character varying
57919	57920	192.168.10.2
57920	57921	192.168.10.2
57921	57922	192.168.10.2
57922	57923	192.168.10.2
57923	5	192.168.220.160