CS310 Report for Route Verification Server

Shouju Wang, Steven He

Duke Kunshan University

Abstract. A REST API server for accessing inter-domain routing information

Keywords: Postgresql · Flask · RestAPI.

1 Introduction

We shall produce a REST API server `route_verification_server`, a Python client library `route_verification_client` for convenient requests to the server, and a set of scripts `route_verification_server_loader` to load data into `route_verification_server`'s database. This software is used for networks research, operation, and maintenance. Users shall be able to query for Internet routing policies and policy verification reports based on various conditions they provide.

2 Motivation

In a changing Internet environment, network operators and researchers need tools to monitor both the routing policies they publish in the IRR and the actual routes permitted to propagate. The RPSL used to specify routing policies has highly relational semantics, where ASes, routes, and other objects are tightly coupled. To retrieve the information necessary to identify network regularities and anomalies, A server to model the relationships and query routing information is needed.

3 Database Design

Our DDL can be found in the file of `demo_v1.sql`

3.1 Design Explaination

According to our specific design and need, we designed a database with multiple relations with the ER diagram like below.

And relational model like below.

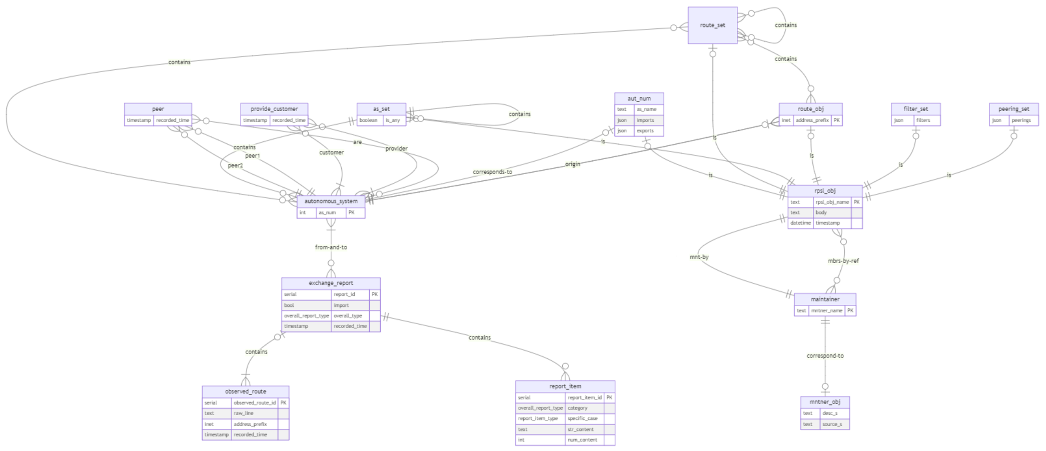


Fig. 1. relational model

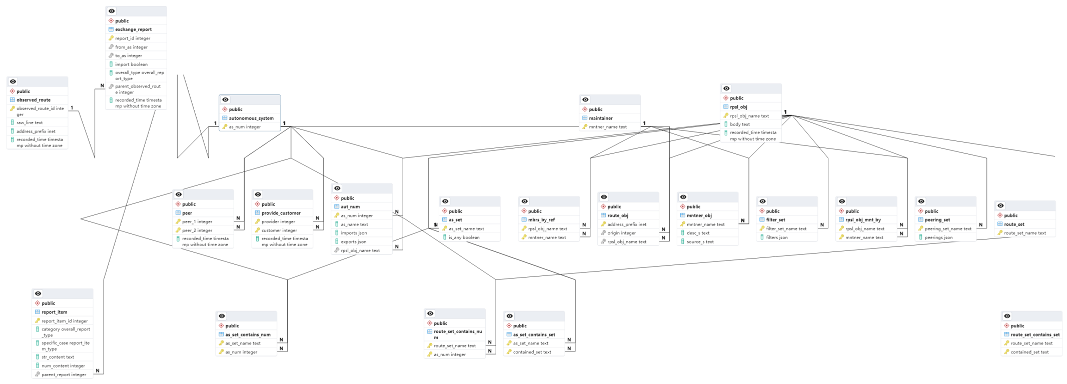
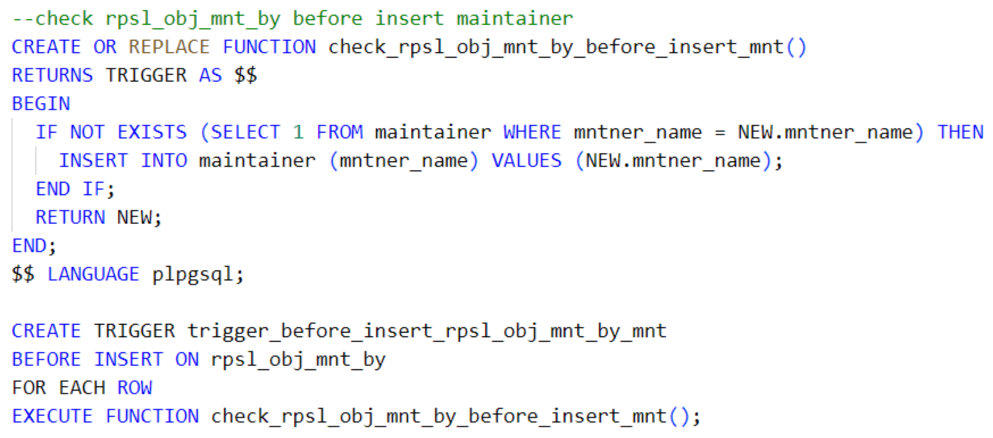


Fig. 2. relational model

3.2 Trigger Design

Our trigger implementation can be found in the file of `trigger_only.sql`

In our project design, `autonomous_system` and `maintainer` are referenced by many other tables through primary-foreign key constraints. Before injecting corresponding data into those tables that reference `autonomous_system` and `maintainer`, we need to check the existence of `as_num` and `mntner_name`. So we designed the trigger SQL to the integrity of this kind of constraint. Like the SQL below, before inserting into `rpsl_obj_mnt_by`, we check if `mnter_name`'s existence.



```
--check rpsl_obj_mnt_by before insert maintainer
CREATE OR REPLACE FUNCTION check_rpsl_obj_mnt_by_before_insert_mnt()
RETURNS TRIGGER AS $$
BEGIN
  IF NOT EXISTS (SELECT 1 FROM maintainer WHERE mntner_name = NEW.mntner_name) THEN
    INSERT INTO maintainer (mntner_name) VALUES (NEW.mntner_name);
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_before_insert_rpsl_obj_mnt_by_mnt
BEFORE INSERT ON rpsl_obj_mnt_by
FOR EACH ROW
EXECUTE FUNCTION check_rpsl_obj_mnt_by_before_insert_mnt();
```

Fig. 3. trigger example

4 Route Verification Server Loader

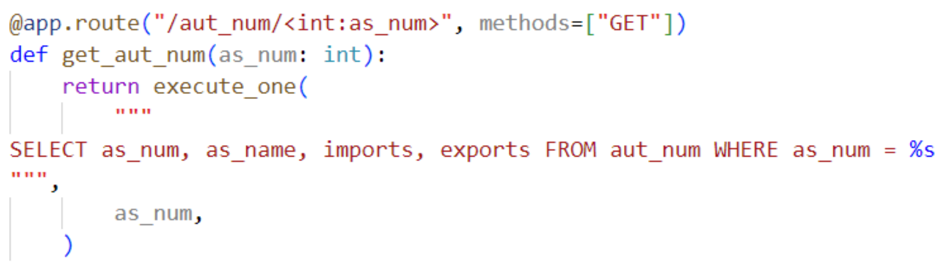
The implement of `Route Verification Server Loader` is the subdirectory of `route_verification_server_loader`

5 Queries For RestAPI Endpoint

We also put the query code in a separate file for the endpoint to use these queries to access data in PostgreSQL.

6 Flask RestAPI

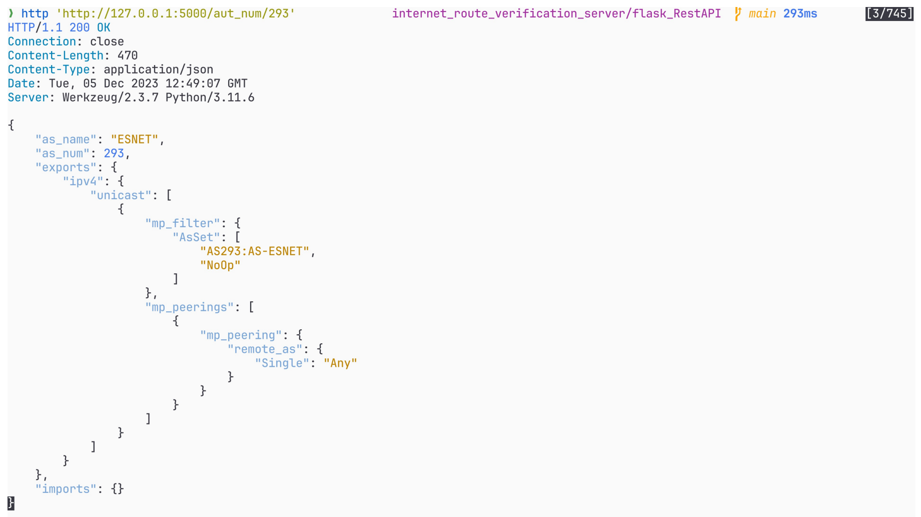
We use flask framework to create a RESTful API. It provides tools for handling HTTP requests, defining routes, and rendering templates. We use RESTful API to interact with the PostgreSQL database to perform CRUD (Create, Read, Update, Delete) operations on data. And we also use JSON to exchange data. Like the example above, we use `@app.route` to define routes, each corre-



```
@app.route("/aut_num/<int:as_num>", methods=["GET"])
def get_aut_num(as_num: int):
    return execute_one(
        """
SELECT as_num, as_name, imports, exports FROM aut_num WHERE as_num = %s
""",
        as_num,
    )
```

Fig. 4. API example

sponding to a specific API endpoint. Then routes handle `HTTP` requests (`GET` in this case) and execute corresponding SQL queries on the PostgreSQL database. The results are returned as JSON responses. Like this result show, we finally



```
❯ http 'http://127.0.0.1:5000/aut_num/293'            internet_route_verification_server/flask_RestAPI  main 293ms       [3/745]
HTTP/1.1 200 OK
Connection: close
Content-Length: 470
Content-Type: application/json
Date: Tue, 05 Dec 2023 12:49:07 GMT
Server: Werkzeug/2.3.7 Python/3.11.6

{
    "as_name": "ESNET",
    "as_num": 293,
    "exports": {
        "ipv4": {
            "unicast": [
                {
                    "mp_filter": {
                        "AsSet": [
                            "AS293:AS-ESNET",
                            "NoOp"
                        ]
                    },
                    "mp_peerings": [
                        {
                            "mp_peering": {
                                "remote_as": {
                                    "Single": "Any"
                                }
                            }
                        }
                    ]
                }
            ]
        }
    },
    "imports": {}
}
```

Fig. 5. result

get `as_num, as_name, imports, exports` from `aut_num` for a given `as_num`, which is the result of the execution of SQL
`SELECT as_num, as_name, imports, exports FROM aut_num WHERE as_num = %s`

7 Division of Labor

Table 1. Division of Labor

Work	worker
Trigger design	Shouju Wang
Data Script and Load	Steven He
SQL for Endpoint	Shouju Wang & Steven He
ER Diagram Design	Steven He
Relational Model Design	Shouju Wang
Flask Implement	Shouju Wang
RestAPI Design	Steven He
Report	Shouju Wang

8 Project Peer Assessment

1. Group1:Good oral introduction for what the system is, but the visuals did not help
2. Group2: The explanation of the problem to solve, retrieving patients' reports is very clear.
3. Group3: Moving the calculations in the reports into SQL is very interesting, and makes the system more suitable to this course.
4. Group4: The intended users, patients, doctors, and researchers, can be emphasized, and their interactions with the database can be specified.