# Jargo:

***Real-time stochastic ridesharing simulator***

`https://github.com/jargors`

Ver. 0.9.0: James J. Pan (pan-j16@mails.tsinghua.edu.cn)

March 25, 2020

# Contents

# Preface

This document is the user manual and the annotated source code for Jargo. I developed Jargo using the noweb[1] literate programming[2] tool. The files in the `src` directory are the source files for this document (`jargo.pdf`) and the Java code (`java/`, `jar/jargors-1.0.0.jar`). With literate programming, the documentation and the code are developed at the same time.

This document is organized into five chapters.

- Chapter 1: is the tutorial. Section 1.1 describes the installation procedure. Section 1.2 explains where to find example clients and traffic functions. Section 1.3 demonstrates how to start a Jargo simulation. Section 1.4 explains how to analyze simulation results.
- Chapter 2 discusses the Jargo model of ridesharing. Developers interested in understanding Jargo's internal model may find this chapter useful. Section 2.1 describes the setting (time and road network). Section 2.2 explains ridesharing users (customers and vehicles). Section 2.3 describes ridesharing service metrics. Section 2.4 presents the SQL schema.
- Chapter 3 presents the simulator components. These chapters serve as a reference for Jargo's classes and methods. Section 3.1 presents an overview of all classes and their public and private methods. Section 3.2 presents methods to administer the simulation. Section 3.3 presents methods to read the simulated ridesharing state. Section 3.4 presents methods to write and update the state. Section 3.5 presents methods for interacting with G-tree. Sections 3.6–3.11 present Jargo classes and class-specific methods.
- Chapter 4 describes the evaluator programs. Section 4.1 presents the command-line evaluator. Sections 4.2–4.3 present the graphical evaluator.
- Chapter 5 list debugging statements and gives troubleshooting suggestions.

## What is Jargo?

Jargo is a Java library that provides real-time ridesharing simulation. It intends to help researchers evaluate the quality of ridesharing algorithms. Jargo offers:

- historical or synthetic real-time customers and vehicles;
- microscopic vehicle routing;
- modular algorithms and traffic conditions;
- various out-of-the-box quality-of-service metrics.

Thanks to these features, it can be used to:

- evaluate the effects of different customer and vehicle configurations, such as customer demand surges and extreme spatial distributions;
- evaluate the effects of algorithm throughput, and observe how throughput changes over time;
- evaluate the effects of traffic;
- perform multi-objective analysis.

Jargo is licensed under the GNU General Public License, Version 3.

[1] https://www.cs.tufts.edu/~nr/noweb/

[2] http://literateprogramming.com/

## Why Literate Programming?

With literate programming, code can be structured in any way and not just in the way imposed by the programming language. For example, suppose you have a `Cat` and `Dog` class, and each have a `speak` method:

```
class Cat {
  public void speak() {
    System.out.println("Meow!");
  }
}
class Dog {
  public void speak() {
    System.out.println("Woof!");
  }
}
```

With literate programming, you could organize the `speak` methods into a single file:

```
<Cat speak>=
public void speak() {
  System.out.println("Meow!");
}
<Dog speak>=
public void speak() {
  System.out.println("Woof!");
}
```

and then add the methods to the classes by referencing them:

```
class Cat {
  <Cat speak>
}
class Dog {
  <Dog speak>
}
```

Putting the `speak` methods together lets you reason about them as a single unit of functionality instead of scattered across various classes. I found that this way of writing code helped me to develop Jargo in terms of reading, writing, and other functionality. The result is a codebase that hopefully is easy to understand, well-reasoned, and correct.

## Reporting Bugs

Report bugs by logging an issue at the official Jargo GitHub repository: https://github.com/jargors/Jargo/issues. By using GitHub, other users can see the existing issues and possible resolutions. You can also write to me directly: pan-j16@mails.tsinghua.edu.cn.

## Contributing

Jargo is an open-source software and contributions are welcome. The recommended way to contribute is to fork the repository (https://github.com/jargors/Jargo), make your changes in your local fork, then create a pull request on GitHub. You can also open an Issue on the GitHub page for any comments or complaints.

If you make your changes directly onto the `*.java` or `*.tex` files, there is a danger of your changes getting overwritten if you accidentally recompile the noweb files. To avoid the danger, remove the entire `src` directory.

James
March 23, 2020

# Chapter 1

# Tutorial

## 1.1 Installation

Jargo exists as a single Java archive (`jar`) file called `jargors-1.0.0.jar`. To "install" it means to compile the Java source files into the `jar`, and then place the `jar` file somewhere on your computer that is accessible by your Java runtime classpath (`-cp`) option. Jargo comes with its own Make-based build system. Type `make` in the Jargo root directory to see a list of build commands (Table 1.1).

If you have an internet connection and are on Linux, Mac, or using Cygwin/MinGW, you might be able to get away with:

```
> make all
```

Otherwise, read on for details.

### 1.1.1 Prerequisites

Before you begin compiling, make sure you have the following prerequisites.

**For compiling Jargo:**

These prerequisites can be automatically obtained by typing `make dep` in the Jargo root directory. The files are downloaded into the `dep/` folder. For GTreeJNI and JavaFX native components, the command gets the x64 Linux versions. If you are on a different platform, you will need to get these native libraries by yourself.

- Java JDK 11.0.1 or above. Latest Java Development Kits licensed under the GPL are available at `https://jdk.java.net`.
- Apache Commons DBCP 2.7.0 package, obtainable from `https://commons.apache.org/proper/commons-dbcp/`.
- Apache Commons Pool 2.7.0 package, needed by DBCP and obtainable from `https://commons.apache.org/proper/commons-pool/`.
- Jargo GTreeJNI 2.0 native library and Java package, obtainable from `https://github.com/jamjpan/GTreeJNI`.
- JavaFX SDK 11 or above, obtainable from `https://openjfx.io/`.
- VisualVM charting components, obtainable from `http://bits.netbeans.org/nexus/content/repositories/visualvm`.

**(Optional) For compiling the documentation (this document):**

- The `texfot` program, included in most distributions of LaTeX.
- The `pdflatex` program, included in most distributions of LaTeX.

**(Optional) For compiling the Java and LaTeX sources:**

- The `notangle` and `noweave` programs, obtainable from `https://www.cs.tufts.edu/~nr/noweb/`.

If you are compiling noweb from source, use `icont` instead of `awk`. If you are on a Debian system, pre-packaged binaries compiled with `icont` should already be available.

### 1.1.2 Building the Documentation

To build the documentation (this file), type `make pdf` in the Jargo root directory.

### 1.1.3 For Users: Building the Library

Follow these steps to build the `jargors-1.0.0.jar` library from the Java sources in the `java/` directory.

1. Verify the `dep/` folder contains the prerequisites listed in Table 1.2. Typing `make dep` will automatically download the prerequisites into the `dep/` folder. Otherwise, the items in *italics* are obtainable from the JavaFX SDK for your platform. The `libgtree.so` native library is obtainable from Jargo GTreeJNI and must be built for your platform beforehand. The remaining `*.jar` files are obtainable from the websites listed above.
2. Type `javac -version` to verify the Java compiler version is at least 11.0.1.
3. Type `make jar`. The compiled library is placed in `jar/jargors-1.0.0.jar`.

### 1.1.4 For Developers: Building the Java and LaTeX Sources

The Jargo Java and LaTeX sources come from noweb files in the `src/` directory. To rebuild the Java and LaTeX sources from these files, type `make src`.

### 1.1.5 Summary

Here is a summary of build targets.

- `make all`: build the library, documentation, and fetch dependencies into the `dep` folder.
- `make jar`: build the `jar/jargors-1.0.0.jar` library only.
- `make pdf`: build the `doc/jargo.pdf` documentation only.
- `make src`: recompile the `java/*/*.java` source files from the `src/*.nw` noweb files.
- `make dep`: download the build prerequisites from the internet.
- `make clean`: delete build objects `jar/`, `com/`, `pdf/`, `build.log`, and `wget.log`
- `make purge`: in addition to `clean`, also delete the Java source files and `doc/body.tex`.
- `make purgedep`: delete the `dep/` folder.

```
---------------------------------------------------------------
Jargo Build System
  Jargo Version: (VERSION) (BUILD_DATE)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Commands:
  make all        build library, documentation, and fetch deps
  make jar        build library only (jar/jargors-VERSION.jar)
  make pdf        build documentation only (pdf/jargo.pdf)
  make src        build Java sources from Noweb files
  make dep        fetch dependencies from the Internet (dep/)
  make clean      delete jar/, com/, pdf/, build.log, wget.log
  make purge      clean + delete Java sources, doc/body.tex
  make purgedep   delete dependencies (dep/)

If you experience any problems, please log an issue at
  https://github.com/jargors/Jargo/issues

Or to be a contributor, you can fork this repository,
make changes in your fork, and submit a pull request
  https://github.com/jargors/Jargo/pulls

Thank you!
===============================================================
```

**Table 1.1:** ***Jargo build commands.***

- commons-dbcp2-2.7.0.jar
- commons-logging-1.2.jar
- commons-pool2-2.7.0.jar
- com-sun-tools-visualvm-charts-RELEASE139.jar
- com-sun-tools-visualvm-uisupport-RELEASE139.jar
- gtree-2.0.jar
- libgtree.so
- *javafx.base.jar*
- *javafx.controls.jar*
- *javafx.fxml.jar*
- *javafx.graphics.jar*
- *javafx.media.jar*
- *javafx.swing.jar*
- *javafx-swt.jar*
- *javafx.web.jar*
- *libavplugin-54.so*
- *libavplugin-56.so*
- *libavplugin-57.so*
- *libavplugin-ffmpeg-56.so*
- *libavplugin-ffmpeg-57.so*
- *libavplugin-ffmpeg-58.so*
- *libdecora_sse.so*
- *libfxplugins.so*
- *libglassgtk2.so*
- *libglassgtk3.so*
- *libglass.so*
- *libgstreamer-lite.so*
- *libjavafx_font_freetype.so*
- *libjavafx_font_pango.so*
- *libjavafx_font.so*
- *libjavafx_iio.so*
- *libjfxmedia.so*
- *libjfxwebkit.so*
- *libprism_common.so*
- *libprism_es2.so*
- *libprism_sw.so*
- org-netbeans-lib-profiler-charts-RELEASE139.jar
- org-netbeans-lib-profiler-ui-RELEASE139.jar
- org-netbeans-modules-profiler-api-RELEASE139.jar
- org-openide-util-lookup-RELEASE139.jar

**Table 1.2:** ***Prerequisites for building Jargo, including JavaFX native libraries.***

## 1.2 Examples

Take a look at *Jargo: Example Clients and Traffic Functions* for some examples, with commentary. You can build this document by going into the `example/` folder and typing `make pdf`.

## 1.3 Starting a Simulation

Start a simulation using an *evaluator*. An evaluator is a standalone Java program that uses the Jargo library to setup and start a simulation. You can write your own or use one of the included evaluators. Jargo includes a command-line evaluator and a graphical evaluator.

### 1.3.1 Setting Up the Runtime Environment

The Java Virtual Machine (JVM) needs to know where to look for the relevant class files and native libraries. Usually these directory locations are passed to the JVM at runtime, as options to the `java` command. All evaluators will need to use the dependencies listed in Table 1.2 in addition to the following dependencies:

- Apache Commons Logging package, needed by DBCP and obtainable from `http://commons.apache.org/proper/commons-logging/`.
- Apache Derby 10.15.1.3 or above, obtainable from `https://db.apache.org/derby/`.

See Table 1.4 for a full list of files.

### 1.3.2 Running the Command-Line Evaluator

If you are on Linux, Mac, or using Cygwin/MinGW, you can use the `launch-cli.sh` script to start the command-line evaluator (Table 1.3). This script sets the classpath to include the `jar/` and `dep/` directories, and sets the native library path to `dep/`. If your dependencies are not in these paths, you will need to modify the `_CLASSPATH` variable and the `-Djava.library.path` flag in `launch-cli.sh` to point to locations on your computer containing the dependencies. Make sure to export `DERBY_HOME` to the location of the Derby root directory before running the script.

```
Jargo, a real-time stochastic ridesharing simulator.
Usage: ./launch-cli [OPTION...] MODE ROAD GTREE PROB CLIENT CLASSNAME

Mandatory arguments:
  MODE       runtime mode, either 'seq' or 'real'
  ROAD       road network *.rnet file
  GTREE      gtree *.gtree file to the road network
  PROB       problem *.instance file (see FORMATS section)
  CLIENT     client *.jar file
  CLASSNAME  client classname

Options:
  -h       show help
  -r       client *.gtree file (default: GTREE)
  -x       traffic *.jar file (default: none)
  -y       traffic classname (default: '')
  -s       start time (see TIME section)
  -e       end time (see TIME section)
```

**Table 1.3:** ***The command-line evaluator.***

### 1.3.3 Running the Graphical Evaluator

Similar to the command-line evaluator, if you are on Linux, Max, or using Cygwin/MinGW, you can use the `launch-gui.sh` script to start the graphical interface (Figure 1.1). This script sets the classpath to include the `jar/` and `dep/` directories, sets the module path to `dep/`, and sets the native library path to `dep/`. You may need to configure these paths for your machine. Again, make sure to export `DERBY_HOME` to the location of the Derby root directory before running the script.

- jargors-1.0.0.jar
- commons-logging-1.2.jar
- *derbyclient.jar*
- *derby.jar*
- *derbyLocale_cs.jar*
- *derbyLocale_de_DE.jar*
- *derbyLocale_es.jar*
- *derbyLocale_fr.jar*
- *derbyLocale_hu.jar*
- *derbyLocale_it.jar*
- *derbyLocale_ja_JP.jar*
- *derbyLocale_ko_KR.jar*
- *derbyLocale_pl.jar*
- *derbyLocale_pt_BR.jar*
- *derbyLocale_ru.jar*
- *derbyLocale_zh_CN.jar*
- *derbyLocale_zh_TW.jar*
- *derbynet.jar*
- *derbyoptionaltools.jar*
- *derbyrun.jar*
- *derbyshared.jar*
- *derbytools.jar*
- *derby.war*

**Table 1.4:** ***Additional prerequisites for running Jargo programs.***

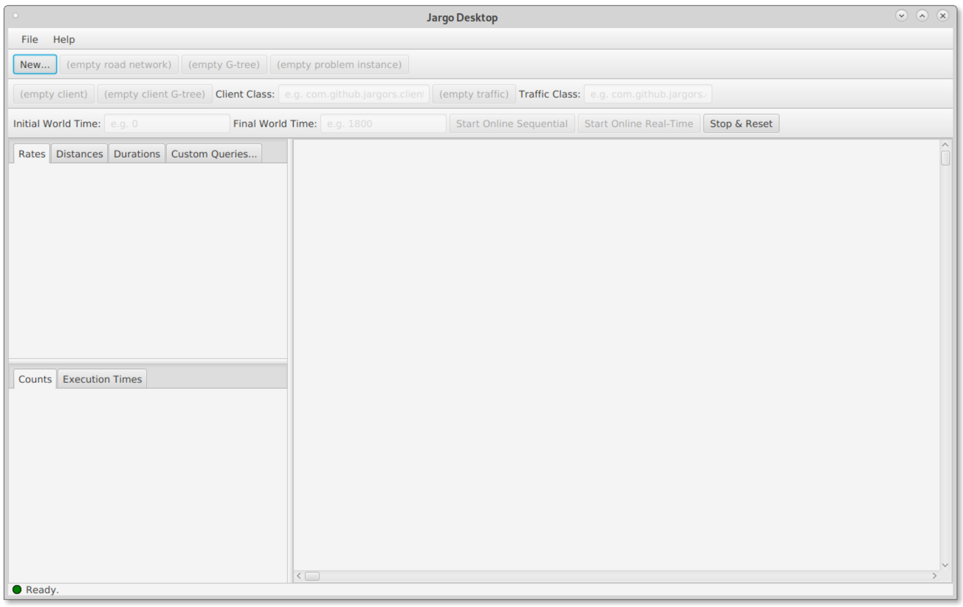

**Figure 1.1:** ***The graphical evaluator.***

## 1.4 Analyzing Results

At the end of each simulation, the command-line and graphical evaluators will export simulation results to disk for offline analysis. The results are stored in Derby database format and can be accessed using any tool that supports the Derby JDBC driver, including the `ij` tool bundled with Derby.

The easiest way to get started is to connect to the database and query Jargo's SQL views (Table 1.5). You can use a JDBC connection string such as `'connect:jdbc:derby:memory:temp;createFrom=jargo'` to connect to the database, replacing `jargo` with the name of the export. This string creates a new in-memory Derby database called `temp` and loads the contents of `jargo` into this database. Use your tool to list the views. In `ij`, the command is `show views`.

```
$ ij
ij version 10.15
ij> connect 'jdbc:derby:memory:temp;createFrom=jargo';
ij> show views;
TABLE_SCHEM         |TABLE_NAME                    |REMARKS
------------------------------------------------------------------------
APP                 |ASSIGNMENTS                   |
APP                 |ASSIGNMENTS_R                 |
APP                 |DIST_BASE                     |
APP                 |DIST_R_BASE                   |
APP                 |DIST_R_DETOUR                 |
APP                 |DIST_R_TRANSIT                |
APP                 |DIST_R_UNASSIGNED             |
APP                 |DIST_S_BASE                   |
APP                 |DIST_S_CRUISING               |
APP                 |DIST_S_SERVICE                |
APP                 |DIST_S_TRAVEL                 |
APP                 |DUR_R_PICKUP                  |
APP                 |DUR_R_TRANSIT                 |
APP                 |DUR_R_TRAVEL                  |
APP                 |DUR_S_SERVICE                 |
APP                 |DUR_S_TRAVEL                  |
APP                 |F_DISTANCE_BLOCKS             |
APP                 |F_STATUS                      |
APP                 |R_SERVER                      |
APP                 |R_USER                        |
APP                 |SERVICE_RATE                  |
APP                 |T_R_ARRIVE                    |
APP                 |T_R_DEPART                    |
APP                 |T_S_ARRIVE                    |
APP                 |T_S_DEPART                    |
APP                 |VIOLATIONS_T_R                |
APP                 |VIOLATIONS_T_S                |

27 rows selected
ij>
```

**Table 1.5:** ***Listing the Jargo views using*** *`ij`*.

### 1.4.1 Assignments

`ASSIGNMENTS`

This view lists all assignments. Each row consists of the assigned vehicle and customer along with the time that the assignments was completed (the customer drop-off time). The `sid` column stores the vehicle identifier and the `rid` column stores the customer identifier. See Table 1.6 for an example. Here are some common queries:

- To get the total number of assignments, use `SELECT COUNT (rid) FROM ASSIGNMENTS`.

- To get assignments per vehicle, use `SELECT sid, COUNT (rid) FROM ASSIGNMENTS GROUP BY sid`.
- To get average number of assignments per vehicle, use `SELECT CAST(COUNT (rid) / COUNT(DISTINCT (sid)) AS FLOAT) FROM ASSIGNMENTS`.

```
ij> SELECT * FROM ASSIGNMENTS FETCH FIRST 5 ROWS ONLY;
T          |SID        |RID
-----------------------------------
34         |791        |240792
78         |396        |240773
80         |477        |240795
85         |629        |240848
88         |503        |240901

5 rows selected
ij>
```

**Table 1.6:** ***The ASSIGNMENTS view.***

`SERVICE_RATE`

This view gives the total "service rate" as a percentage multiplied by $10^4$ (*e.g.* 1.0, or 100%, is written as 10000). The service rate is found by dividing the number of assigned customers over the total number of customers. The total is listed in the `val` column.

### 1.4.2 Distances

`DIST_BASE`

This view lists the total "base" distance for all customers and vehicles, in meters. The base distance for a customer is the shortest travel distance from the customer's pick-up location to the drop-off location, and for a vehicle is the shortest travel distance from the vehicle's starting location to the ending location. The total is listed in the `val` column.

`DIST_R_BASE`

This view lists the total "base" distance for all customers only.

`DIST_R_DETOUR`

This view lists the "detour" distance for each customer, in meters. The detour distance is found by taking the customer's transit distance and then subtracting the customer's base distance. The `rid` column lists the customer identifier and the `val` column lists the detour distance. See Table 1.7 for an example.

`DIST_R_TRANSIT`

This view lists the "transit" distance for each customer, in meters. The transit distance is the distance the customer actually traveled by taking a ridesharing vehicle. The `rid` column lists the customer identifier and the `val` column lists the detour distance. See Table 1.8 for an example.

`DIST_R_UNASSIGNED`

This view lists the total "base" distance for all *unassigned* customers only.

`DIST_S_BASE`

This view lists the total "base" distance for all vehicles only.

```
ij> select * from dist_r_detour fetch first 5 rows only;
RID        |VAL
-----------------------
51300      |1178
51301      |1384
51302      |854
51303      |1685
51304      |1854

5 rows selected
ij>
```

**Table 1.7:** ***The*** *DIST_R_DETOUR* ***view.***

```
ij> select * from dist_r_transit fetch first 5 rows only;
RID        |VAL
-----------------------
51300      |5328
51301      |5242
51302      |1654
51303      |6646
51304      |10356

5 rows selected
ij>
```

**Table 1.8:** ***The*** *DIST_R_TRANSIT* ***view.***

`DIST_S_CRUISING`

This view lists the "cruising" distance for each vehicle, in meters. The cruising distance is the distance the vehicle traveled while empty (no customers onboard).

`DIST_S_SERVICE`

This view lists the "service" distance for each vehicle, in meters. The service distance is the distance the vehicle traveled while having customers onboard.

`DIST_S_TRAVEL`

This view lists the "travel" distance for each vehicle, in meters. The travel distance is the sum of the service and cruising distances.

### 1.4.3 Durations

`DUR_R_PICKUP`

This view lists the "pick-up" duration for each customer, in seconds. The pick-up duration is the difference between the time a customer is picked up and the time the customer first appears on the road network. See Table 1.9 for an example.

`DUR_R_TRANSIT`

This view lists the "transit" duration for each customer, in seconds. The transit duration is the difference between the time a customer is dropped off and the time the customer is picked up, in other words the time a customer spends inside a vehicle. See Table 1.10 for an example.

```
ij> select * from dur_r_pickup fetch first 5 rows only;
RID        |VAL
-----------------------
240764     |36
240765     |107
240766     |31
240767     |45
240768     |33

5 rows selected
ij>
```

**Table 1.9:** ***The*** *DUR_R_PICKUP* ***view.***

```
ij> select * from dur_r_transit fetch first 5 rows only;
RID        |VAL
-----------------------
240764     |139
240765     |163
240766     |242
240767     |1075
240768     |380

5 rows selected
ij>
```

**Table 1.10:** ***The*** *DUR_R_TRANSIT* ***view.***

DUR_R_TRAVEL

This view lists the "travel" duration for each customer, in seconds. The travel duration is the difference between the time a customer is dropped off and the time the customer appears on the road network, in other words the sum of the pick-up and transit durations. See Table 1.11 for an example.

```
ij> select * from dur_r_travel fetch first 5 rows only;
RID        |VAL
-----------------------
240764     |175
240765     |270
240766     |273
240767     |1120
240768     |413

5 rows selected
ij>
```

**Table 1.11:** ***The*** *DUR_R_TRAVEL* ***view.***

DUR_S_SERVICE

This view lists the "service" duration for each vehicle, in seconds. The service duration is the time spent with customers onboard.

`DUR_S_TRAVEL`

This view lists the "travel" duration for each vehicle, in seconds. The travel duration is the total time spent traveling on the road network.

### 1.4.4 Other Views

`F_DISTANCE_BLOCKS`

This view lists the departure load on each vehicle for each location the vehicle visits. It is used to determine service and cruising distances.

`F_STATUS`

This view lists the "status" of each assigned customer after pick-up and after drop-off. It is used to determine the assignments.

`R_SERVER`

This view lists each vehicle location and the "events" that took place on those locations. See Section 2.2.7 for more information.

`R_USER`

This view lists each vehicle and customer along with their properties. See Section **??** for more information.

`T_R_ARRIVE`

This view lists drop-off times for each customer.

`T_R_DEPART`

This view lists pick-up times for each customer.

`T_S_ARRIVE`

This view lists arrival times for each vehicle.

`T_S_DEPART`

This view lists departure times for each vehicle.

`VIOLATIONS_T_R`

This view lists the amount of the "time window violation" for each customer. This amount is found by taking the drop-off time and subtracting the latest acceptable drop-off time for the customer.

`VIOLATIONS_T_S`

This view lists the amount of the "time window violation" for each vehicle.

# Chapter 2

# Ridesharing Model

Physical concepts, such as customers, vehicles, and ridesharing service-related metrics, are defined on Jargo's data tables using relational and set algebra. For a primer on relations and notes on notation used in this document, see Appendix A.

## 2.1 Ridesharing Setting

This section describes Jargo's model for the ridesharing setting.

### 2.1.1 Time

Time is modeled as a positive integer $1 \leq t \leq H$. A time horizon $H$ bounds the system. Time can be operated on. Times cannot be added, but a later (greater) time can subtract an earlier (lesser) time. The difference is called a duration, represented by the symbol $\delta$. Durations can add and subtract each other to get new durations, and times can also add and subtract durations to get new times.

### 2.1.2 Road Network

The road network is modeled as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Vertices in $\mathcal{V}$ represent points along roads in the network. A function $V : \mathcal{V} \to \mathbb{R}^2$ maps vertices to 2-dimensional latitude and longitude coordinates in the real world, and an inverse function map-matches customers and vehicles to vertices. Edges in $\mathcal{E}$ represent road segments. The pair $(a, b) \in \mathcal{V}^2, a \neq b$ exists in $\mathcal{E}$ only if physical traffic flows from $V(a)$ to $V(b)$, and for all $c \in \mathcal{V} \setminus \{a, b\}$ no traffic flows from $V(a)$ to $V(c)$ and from $V(c)$ to $V(b)$. A function $d : \mathcal{E} \to \mathbb{R}_{>0}$ maps edges to positive real weights corresponding to distance along the edge, and the shortest-path distances between the pairs among any three vertices satisfies the triangle inequality. Figure 2.1 shows an example road network, drawn in QGIS, that could be supported by Jargo's model.

### 2.1.3 Paths

A path $p = (p_i)_{i \in 1..n} = p_1..p_n$ is defined as a sequence of $n$ vertices such that any two adjacent vertices are an edge, or $(p_i, p_{i+1}) \in \mathcal{E}$ for $i \in 1..(n-1)$. A vertex or edge can appear multiple times in a path. The path distance is

$$\sum_{i=1}^{n-1} d(p_i, p_{i+1}).$$

Path $p$ is a shortest path only if it minimizes the distance out of all possible paths from $p_1$ to $p_n$. Multiple shortest paths are possible.

### 2.1.4 Waypoints

Waypoints are used to describe points in time as well as space. A waypoint is defined as a tuple $(\mathtt{t}, \mathtt{v})$, with the domain of $\mathtt{t}$ as $1..H$ and the domain of $\mathtt{v}$ as $\mathcal{V}$. Waypoints can be labeled in a way that will be discussed later.

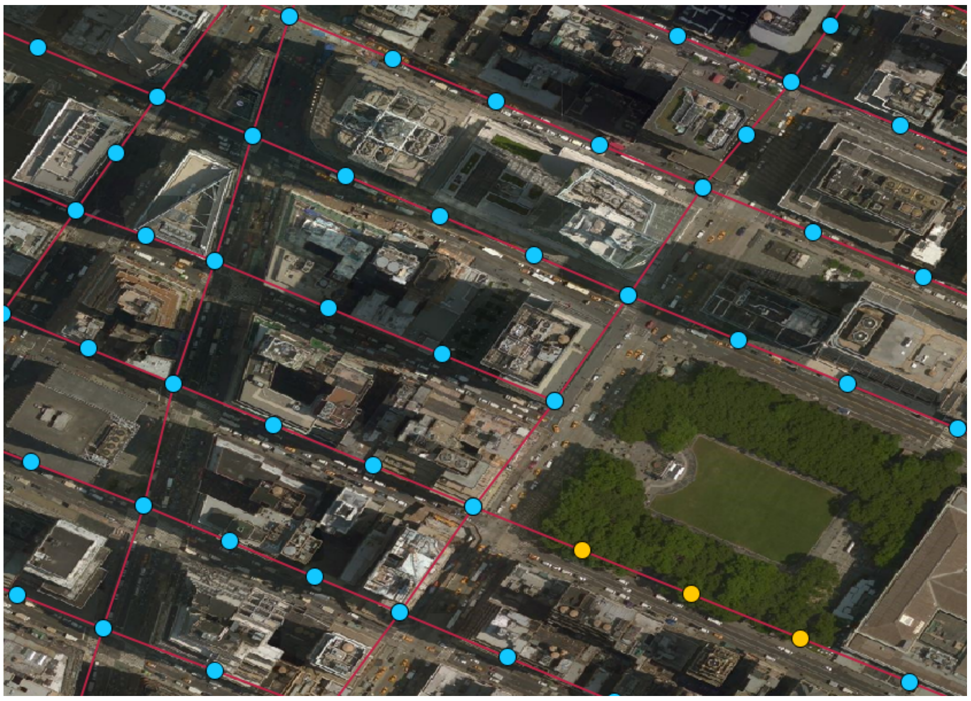

**Figure 2.1:** ***Portion of a road network graph showing edges (red lines) and vertices (blue circles) overlayed on top of Manhattan (QGIS 2.18.16, Bing Aerial). Vertices do not have to be at an intersection (orange circles, lower right).***

### 2.1.5 Routes

Routes are formed by a sequence of waypoints. A route $w = (w_i)_{i \in 1..n} = w_1..w_n = (t_1, v_1)..(t_n, v_n)$ is defined as a sequence of $n$ waypoints such that $t_1..t_n$ is strictly increasing and $v_1..v_n$ is a path. In the spatial dimension, function

$$D(w) = \sum_{i=1}^{n-1} d(\pi_{\mathtt{v}}(w_i), \pi_{\mathtt{v}}(w_{i+1}))$$

gives the route distance, analogous to path distance. In the time dimension, function

$$\delta(w) = \pi_{\mathtt{t}}(w_n) - \pi_{\mathtt{t}}(w_1)$$

gives the route duration. Given a time $t$,

$$w_{\leq t} = \mathrm{sort}(\sigma_{\mathtt{t} \leq t}(w)) \quad \text{and} \quad w_{>t} = \mathrm{sort}(\sigma_{\mathtt{t} > t}(w))$$

give the traveled route denoted $w_{\leq t}$, and the remaining route denoted $w_{>t}$. As the selection operator imposes no ordering on the resulting set, a sort(...) function is introduced to sort a set of waypoints by time in ascending order, returning a sequence. For two adjacent waypoints $w_i$ and $w_{i+1}$, function

$$\nu(w_i, w_{i+1}) = \frac{d(\pi_{\mathtt{v}}(w_i), \pi_{\mathtt{v}}(w_{i+1}))}{\pi_{\mathtt{t}}(w_{i+1}) - \pi_{\mathtt{t}}(w_i)}$$

gives the waypoint rate, or more intuitively the speed. As $d$ only applies to edges, $\nu$ only applies to adjacent waypoints. Speeds can be bounded above by a value $\nu^{\max}(v_i, v_{i+1})$ on each edge, for example to describe road speed limits.

## 2.2 Ridesharing Users

In Jargo, the basic entity representing a ridesharing participant is the *user*. Table 2.1 describes the types of users recognized by Jargo, and Table 2.2 describes their properties. Table 2.3 describes rules governing their behavior. A user is classified as a *request* if it represents a Type 1 or Type 2 customer, or classified as a *server* if it represents a Type 3 or Type 4 vehicle. As only vehicles can move about (P4), only servers are associated with routes in order to describe the motions. Later, schedules describing pick-up and drop-off events are defined on the routes.

| Type | Description |
|---|---|
| Type 1 | Single customer traveling alone |
| Type 2 | Group of customers traveling together |
| Type 3 | Ridesharing vehicle with a predefined final destination |
| Type 4 | Taxi-like vehicle continually serving customers without an explicit destination of its own |

**Table 2.1:** ***Types of ridesharing users.***

| Prop. | Description |
|---|---|
| P1 | **Load.** Each user has a non-zero load, indicating a number of needed seats. For Type 1 users the load is 1, indicating they only need a single seat. For Type 2 users the load exceeds 1. For Type 3 and Type 4 users the load is negative, indicating they have an availability of seats. |
| P2 | **Origin and Destination.** Each user has an origin and a destination, except for Type 4 users that only have an origin. For Type 1 and Type 2, the origin indicates the initial location of the customer and the destination indicates the desired final location. For Type 3, the origin and destination indicate where the vehicle's ridesharing service begins and ends. |
| P3 | **Time Window.** Each user has an early time and a late time, together forming the user's time window. For a Type 1 or Type 2 customer, the time window gives the desired departure time from the origin and the desired arrival time at the destination. For a Type 3 or Type 4 vehicle, the time window gives the time when service begins and the latest time that service can end. The early time precedes the late time. |

**Table 2.2:** ***Ridesharing user properties.***

| Prop. | Description |
|---|---|
| P4 | **Motion.** Users are bound to a network of roads, for example the streets of a city. Only vehicles may directly travel along the roads, whereas customers must be serviced by a vehicle. Both customers and vehicles may enter the system at any time and anywhere. |
| P5 | **Pick-ups and Drop-offs.** For a vehicle to service a customer, it must first travel to the customer's origin to pick up the customer, and then to the customer's destination to drop off the customer, in that order. The customer enters the vehicle during the pick-up and exits the vehicle during the drop-off. These visits must occur within the customer's time window. |
| P6 | **Vehicle Seats.** When a customer enters a vehicle, the customer occupies a number of seats equal to the customer's load. When it exits the vehicle, it relinquishes the seats. At no time can the number of occupied seats exceed the number of available seats in a vehicle. |
| P7 | **User States.** A customer can be in one of three states at any time: waiting for pick-up; in-transit following a pick-up but before the drop-off; or arrived at destination. A vehicle can be either in-service or out-of-service. |

**Table 2.3:** ***Rules governing user behavior.***

### 2.2.1 User Relation

A user $u$ is a 5-tuple defined by $u := (\mathtt{q}, \mathtt{e}, \mathtt{l}, \mathtt{o}, \mathtt{d})$. The $\mathtt{q}$ component corresponds to the user load; the $\mathtt{e}$ and $\mathtt{l}$ components correspond to the user early and late times; the $\mathtt{o}$ and $\mathtt{d}$ components correspond to the user origin and destination. From P1–P4, the domain of $\mathtt{q}$ is the non-zero integers; the domain of $\mathtt{e}$ is $1..(H-1)$ and the domain of $\mathtt{l}$ is $(u_\mathtt{e}+1)..H$; the domains of $\mathtt{o}$ and $\mathtt{d}$ are both $\mathcal{V}$. For a Type 4 vehicle, the destination can be set to a dummy vertex with edge weight equal to 0 to every other vertex in the road network.

The set of all users forms the 5-ary relation $\mathcal{U}$, called the user relation. The set $\mathcal{U}_\mathtt{o} = \pi_\mathtt{o}(\mathcal{U})$ contains all origins and $\mathcal{U}_\mathtt{d} = \pi_\mathtt{d}(\mathcal{U})$ contains all destinations. From P1, a user can be classified as either a request or a server based on its load.

As a convenience, the notation $d_u$ is used to denote the distance of the shortest path from $u_\mathtt{o}$ to $u_\mathtt{d}$ on graph $\mathcal{G}$, and the notation $\delta_u$ is used to denote the shortest travel duration along $d_u$ using the speed limits $\nu^{\max}$ along the shortest-path edges.

### 2.2.2 Requests

A request represents a Type 1 or Type 2 customer. Relation $\mathcal{R} \subseteq \mathcal{U}$,

$$\mathcal{R} = \sigma_{\mathtt{q}>0}(\mathcal{U}),$$

forms the set of all requests by taking users with positive loads. The set $\mathcal{R}_{\mathtt{o}} = \pi_{\mathtt{o}}(\mathcal{R})$ is the set of all request origins and $\mathcal{R}_{\mathtt{d}} = \pi_{\mathtt{d}}(\mathcal{R})$ is the set of all request destinations.

### 2.2.3 Servers

Likewise, a server represents a Type 3 or Type 4 vehicle. Relation $\mathcal{S} = \mathcal{U} \setminus \mathcal{R}$, or

$$\mathcal{S} = \sigma_{\mathtt{q}<0}(\mathcal{U}),$$

forms the set of all servers. The set $\mathcal{S}_{\mathtt{o}} = \pi_{\mathtt{o}}(\mathcal{S})$ is the set of all server origins and $\mathcal{S}_{\mathtt{d}} = \pi_{\mathtt{d}}(\mathcal{S})$ is the set of all server destinations.

### 2.2.4 Routes

To encode vehicle motions, Jargo associates each server $s \in \mathcal{S}$ with a route and a schedule. A server's route is a representation of the corresponding vehicle's motion through the road network while a server's schedule encodes the times and locations of customer pick-ups and drop-offs.

Variable $w$ indicates the route for a server $s$. As time advances, the traveled route $w_{\leq t}$ encodes the server's past motion while the remaining route $w_{>t}$ encodes the future motion. From P2 and P3, Jargo subjects all routes to two rules:

R1. The time component of the first waypoint equals the server's early time, and the time component of the last waypoint is not greater than the server's late time, or $\pi_{\mathtt{t}}(w_1) = s_{\mathtt{e}}$ and $\pi_{\mathtt{t}}(w_{|w|}) \leq s_{\mathtt{l}}$;

R2. The vertex components of the first and last waypoints equal the server's origin and destination respectively, or $\pi_{\mathtt{v}}(w_1) = s_{\mathtt{o}}$ and $\pi_{\mathtt{v}}(w_{|w|}) = s_{\mathtt{d}}$.

### 2.2.5 Schedules

A server's schedule describes the events along the route and not any new motion. It is a subsequence of the server's route $w$

$$b = (b_j)_{j\in 1..m} = (w_{i_j})_{j\in 1..m} = (t_{i_1}, v_{i_1})..(t_{i_m}, v_{i_m}),$$

with $m \leq |w|$ waypoints. Schedules are subjected to a couple rules. First:

R3. The first and last waypoints $b_1$ and $b_m$ equal the first and last waypoints of $w$, or $b_1 = w_1$ and $b_m = w_{|w|}$.

This rule will help later when defining departure and arrival times. Second, from P5:

R4. For each waypoint $b_j$ for $j \in 2..(m-1)$, the vertex component is either a request origin or request destination, or $\pi_{\mathtt{v}}(b_j) \in \mathcal{R}_{\mathtt{o}} \cup \mathcal{R}_{\mathtt{d}}$.

In other words, each entry or exit must occur at a customer origin or destination.

A schedule formalizes the notion of shared travel with other users, as multiple entries and exits can overlap within the same server route. At time $t$, the traveled schedule denoted $b_{\leq t}$ encodes the past entries and exits and is given by $\sigma_{\mathtt{t}\leq t}(b)$. Likewise, the remaining schedule denoted $b_{>t}$ encodes the future entries and exits and is given by $\sigma_{\mathtt{t}>t}(b)$.

### 2.2.6 Schedule Labels

Each waypoint in schedule $b$ has a set of labels in order to identify which customers are entering and exiting the vehicle at the waypoint's time and location. A labeling scheme can be applied to $b$ to determine each of the labels. The set of all possible labels depends on the locations of the waypoints. Let

$$\mathcal{R}' = \sigma_{\mathtt{o} \in \pi_{\mathtt{v}}(b) \vee \mathtt{d} \in \pi_{\mathtt{v}}(b)}(\mathcal{R})$$

give the set of requests whose origin or destination is found in at least one waypoint in $b$. Conceptually, the labeling scheme

$$L : b \to \mathbb{P}(\mathcal{R}' \cup \{s\})$$

maps elements of $b$ to elements of the power set of $\mathcal{R}' \cup \{s\}$. By using the power set $\mathbb{P}$, a waypoint can have multiple labels, representing the case where multiple customers enter or exit the vehicle at the waypoint. The labeling scheme is subjected to the following labeling rules:

R5. No waypoint can be labeled with $r \in \mathcal{R}'$ if a schedule for another server already contains waypoints labeled with $r$;

R6. A waypoint $b_j \in b$ can be labeled with $r$ only if $\pi_{\mathtt{v}}(b_j) = r_{\mathtt{o}}$ or $\pi_{\mathtt{v}}(b_j) = r_{\mathtt{d}}$;

R7. If $b_j$ is to be labeled with $r$ and $\pi_{\mathtt{v}}(b_j) = r_{\mathtt{o}}$, then a second waypoint $b_{j'}$ such that $j' > j$ and $\pi_{\mathtt{v}}(b_{j'}) = r_{\mathtt{d}}$ must also be labeled with $r$;

R8. The time components of $b_j$ and $b_{j'}$ must be within request $r$'s time window, formally $r_{\mathtt{e}} \leq \pi_{\mathtt{t}}(b_j)$ and $\pi_{\mathtt{t}}(b_{j'}) \leq r_{\mathtt{l}}$;

R9. The number of waypoints labeled with $r$ must be exactly 0 or 2;

R10. The first and last waypoints must contain the schedule's server $s$ in their labels, and no other waypoint can be labeled with $s$.

Rules R5–R9 express P5. Rule R10 can be interpreted to mean that a vehicle must "serve itself" at its own origin and destination. This last rule helps to define later concepts.

### 2.2.7 Server Relation

By combining the routes, schedules, and labels into a set of $(\mathtt{s}, \mathtt{t}, \mathtt{v}, \mathtt{L})$ tuples, a 4-ary relation $\mathcal{X}$ can be formed. Jargo calls this relation the server relation. Each tuple associates the waypoint in the `t` and `v` components with the server in the `s` component, along with the labels in the `L` component.

A server's route can be recovered by extracting `t` and `v` components and sorting by time, or formally for a given server $s$, its route is given by

$$W(\mathcal{X}, s) = \mathrm{sort}(\pi_{\mathtt{t},\mathtt{v}}(\sigma_{\mathtt{s}=s}(\mathcal{X}))).$$

Similarly, a server's schedule can be recovered by extracting only those waypoints that are labeled, formally

$$B(\mathcal{X}, s) = \mathrm{sort}(\pi_{\mathtt{t},\mathtt{v}}(\sigma_{\mathtt{s}=s \wedge |\mathtt{L}|>0}(\mathcal{X}))).$$

The server relation can be used to define the remaining physical concepts, P6 and P7.

### 2.2.8 Request Status

Given a request $r$, the function

$$\mathrm{status}(\mathcal{X}, r, t) = |\sigma_{\mathtt{t} \leq t \wedge r \in \mathtt{L}}(\mathcal{X})| \tag{2.1}$$

gives the count of the tuples labeled with $r$ before or on a given time. From the labeling rules, the count can be only 0, 1, or 2. These counts correspond to request waiting, in-transit, and arrived states from P7, respectively.

Given a server $s$, knowing the in-transit requests for $s$ can be useful for pricing and other rider-related metrics. These requests can be found by

$$\mathcal{Q}(\mathcal{X}, s, t) = \{r \in \mathcal{R} \mid \mathrm{status}(\mathcal{X}, r, t) = 1 \wedge \pi_{\mathtt{s}}(\sigma_{r \in \mathtt{L}}(\mathcal{X})) = s\}.$$

### 2.2.9 Load Burden

The load burden on $s$ can be computed using the in-transit requests by

$$Q(\mathcal{X}, s, t) = \sum_{r \in \mathcal{Q}(\mathcal{X}, s, t)} r_{\mathtt{q}}. \tag{2.2}$$

From P6, server routes are subject to the additional rule:

R11. $Q(\mathcal{X}, s, t) \leq -s_{\mathtt{q}}$ must be true for all $s$ and $t$.

## 2.3 Ridesharing Metrics

A variety of metrics can be measured by simple operations on $\mathcal{U}$ and $\mathcal{X}$. This section lists those that have been implemented in Jargo.

### 2.3.1 Assignments

Server $s$ is said to be assigned to request $r$ at time $t$ only if $\text{status}(\mathcal{X}, r, t) = 2$. That is, the request's status is arrived at time $t$. The set of $(s, r)$ pairs where this property is true is called the set of assignments, formally

$$\textit{assignments } A(\mathcal{X}, t) = \{(s, r) \in \mathcal{S} \times \mathcal{R} \mid \text{status}(\mathcal{X}, r, t) = 2\}. \tag{2.3}$$

Using the assignments,

$$\textit{assigned requests } R^{\text{ok}}(\mathcal{X}, t) = \pi_{\mathtt{r}}(A(\mathcal{X}, t)), \text{ and} \tag{2.4}$$

$$\textit{unassigned requests } R^{\text{ko}}(\mathcal{X}, t) = \mathcal{R} \setminus \mathcal{R}^{\text{ok}}(\mathcal{X}, t). \tag{2.5}$$

The server assigned to $r$ can be obtained with

$$S(\mathcal{X}, r, t) = \{s \in \mathcal{S} \mid \text{status}(\mathcal{X}, r, t) = 2\}, \tag{2.6}$$

guaranteed to return only one server due to R5. Likewise, the set of requests assigned to $s$ can be obtained with

$$R(\mathcal{X}, s, t) = \{r \in \mathcal{R} \mid \text{status}(\mathcal{X}, r, t) = 2\}. \tag{2.7}$$

### 2.3.2 Service Rate

The service rate is the number of assigned requests over the number of all requests, or

$$\textit{service rate } \mu(\mathcal{X}, t) = \frac{|R^{\text{ok}}(\mathcal{X}, t)|}{|\mathcal{R}|}. \tag{2.8}$$

### 2.3.3 Distances

The base distance is the sum of the shortest-path distances for all users, or

$$\textit{base distance } D^{\text{base}}(\mathcal{U}) = \sum_{u \in U} d_u. \tag{2.9}$$

The travel distance for one server $s$ is the distance of its route, $D(W(\mathcal{X}, s))$, and the travel duration can be found with $\delta(W(\mathcal{X}, s))$.

For a server with route $w$, travel distance $D(w)$ can be partitioned into cruising distance $D_0(w)$ and service distance $D_1(w)$. The cruising distance sums the distance along portions where the load burden is zero. The service distance sums the distance along portions of $w$ where the load burden is non-zero. Formally, partition $w$ into a set of substrings $\Omega(w)$ such that each waypoint in $w$ is a member of exactly one substring and that for all substrings $\omega \in \Omega$,

$$\text{either } Q(\mathcal{X}, s, t) = 0 \text{ is true for all } t \in \pi_{\mathtt{t}}(\omega), \tag{2.10}$$

$$\text{or } Q(\mathcal{X}, s, t) > 0 \text{ is true for all } t \in \pi_{\mathtt{t}}(\omega). \tag{2.11}$$

The equations can be used to partition $\Omega(w)$ into two subsets,

$$\Omega_0(w) = \{\omega \in \Omega(w) \mid \omega \text{ satisfies Eq. 2.10}\} \text{ and}$$
$$\Omega_1(w) = \{\omega \in \Omega(w) \mid \omega \text{ satisfies Eq. 2.11}\}.$$

The distances of each of the substrings in each subset can be summed to get

$$D_0(w) = \sum_{\omega \in \Omega_0(w)} D(\omega) \quad \text{and} \quad D_1(w) = \sum_{\omega \in \Omega_1(w)} D(\omega).$$

These distances are written as

$$\textit{cruising distance } D^{\text{cruise}}(\mathcal{X}, s) = D_0(W(\mathcal{X}, s)), \text{ and} \tag{2.12}$$

$$\textit{service distance } D^{\text{service}}(\mathcal{X}, s) = D_1(W(\mathcal{X}, s)). \tag{2.13}$$

### 2.3.4 Detours and Delays

In physical terms, the detour route for a customer is the portion of a vehicle's route between when it visits the customer's origin and destination. Formally, let $w = W(\mathcal{X}, S(\mathcal{X}, r, H))$ be the route of the server assigned to $r$. The detour route $\Delta W(\mathcal{X}, r)$ is an $m$-length substring of $w$ given by $\Delta W(\mathcal{X}, r) = w_{1+k}..w_{m+k}$ such that for some $k$,

- $\Delta W(\mathcal{X}, r)$ begins at $r_{\mathtt{o}}$, or $\pi_{\mathtt{v}}(w_{1+k}) = r_{\mathtt{o}}$,
- $\Delta W(\mathcal{X}, r)$ ends at $r_{\mathtt{d}}$, or $\pi_{\mathtt{v}}(w_{m+k}) = r_{\mathtt{d}}$, and
- the first and last waypoints of $\Delta W(\mathcal{X}, r)$ are labeled with $r$, or $r \in \pi_{\mathtt{L}}(w_{1+k}) \cap \pi_{\mathtt{L}}(w_{m+k})$.

Observe that due to the labeling rules, only one value of $k$ can satisfy these conditions. The first and last waypoints $w_{1+k}$ and $w_{m+k}$ can be found by the equations on users,

$$\text{pickup}(\mathcal{X}, u) = \pi_{\mathtt{t},\mathtt{v}}(\sigma_{\mathtt{v}=u_{\mathtt{o}} \wedge u \in \mathtt{L}}(\mathcal{X})), \text{ and} \tag{2.14}$$
$$\text{dropoff}(\mathcal{X}, u) = \pi_{\mathtt{t},\mathtt{v}}(\sigma_{\mathtt{v}=u_{\mathtt{d}} \wedge u \in \mathtt{L}}(\mathcal{X})), \tag{2.15}$$

by substituting $r$ for $u$. Note that if a server is substituted for $u$, these equations give the start and end waypoints of the server's route due to R3 and R10. These two equations can also be used to give two times for any user,

$$\textit{departure time } t^{\text{depart}}(\mathcal{X}, u) = \pi_{\mathtt{t}}(\text{pickup}(\mathcal{X}, u)), \text{ and} \tag{2.16}$$
$$\textit{arrival time } t^{\text{arrive}}(\mathcal{X}, u) = \pi_{\mathtt{t}}(\text{dropoff}(\mathcal{X}, u)). \tag{2.17}$$

In the real world, the time until a vehicle picks up a customer can be of interest. This pick-up delay can be found with

$$\textit{pick-up delay } \delta^{\text{pickup}}(\mathcal{X}, r) = \pi_{\mathtt{t}}(\text{pickup}(\mathcal{X}, r)) - r_{\mathtt{e}}. \tag{2.18}$$

The detour route $\Delta W(\mathcal{X}, r)$ can only apply to assigned requests. If a detour route exists, then the transit distance and duration are

$$\textit{transit distance } D^{\text{transit}}(\mathcal{X}, r) = D(\Delta W(\mathcal{X}, r)), \text{ and} \tag{2.19}$$
$$\textit{transit duration } \delta^{\text{transit}}(\mathcal{X}, r) = \delta(\Delta W(\mathcal{X}, r)). \tag{2.20}$$

Similarly, the detour distance and duration are

$$\textit{detour distance } D^{\text{detour}}(\mathcal{X}, r) = D^{\text{transit}}(\mathcal{X}, r) - d_r, \text{ and} \tag{2.21}$$
$$\textit{detour duration } \delta^{\text{detour}}(\mathcal{X}, r) = \delta^{\text{transit}}(\mathcal{X}, r) - \delta_r. \tag{2.22}$$

Finally, the travel duration is the sum of the pick-up and transit durations,

$$\textit{travel duration } \delta^{\text{travel}}(\mathcal{X}, r) = \delta^{\text{pickup}}(\mathcal{X}, r) + \delta^{\text{transit}}(\mathcal{X}, r) = \pi_{\mathtt{t}}(\text{dropoff}(\mathcal{X}, r)) - r_{\mathtt{e}}. \tag{2.23}$$

### 2.3.5 Utilization

The percentage of servers that are assigned to at least one request is given by

$$\textit{server utilization } \rho^{\text{server}}(\mathcal{X}) = \frac{|\pi_{\mathtt{s}}(\mathcal{A}(\mathcal{X}))|}{|\mathcal{S}|}. \tag{2.24}$$

The distance utilization is

$$\textit{distance utilization } \rho^{\text{distance}}(\mathcal{X}) = \frac{\sum_{s \in \mathcal{S}} D^{\text{service}}(\mathcal{X}, s)}{\sum_{s \in \mathcal{S}} D(\mathcal{X}, s)}. \tag{2.25}$$

## 2.4 SQL Schema

The simple constraints allowed by the SQL standard[1] (`CHECK`, `UNIQUE`, `NOT NULL`, `FOREIGN KEY`) are unable to express the complex ridesharing properties and rules, and consequently a direct "translation" of the ridesharing relations into SQL is not possible without either making code extensions to SQL or reorganizing the relational ridesharing model.

Jargo implements the following schema entirely in standard SQL without any code extensions while staying faithful to the model. In this schema, *tables* capture the descriptive elements of the model and *views* express the analytical measures. Tables are further organized into property, solution, and constraint tables. Property tables store the road network $\mathcal{G}$ and the user relation $\mathcal{U}$. Solution tables store the server relation $\mathcal{X}$. Constraint tables store copies of data from other tables for validation purposes. The views are mostly defined on the constraint tables.

Diagrams of the SQL tables are included in this chapter. In the diagrams, primary keys are indicated in italics. Elsewhere, column names are distinguished by sans serif script. Parentheses are used to logically group together columns. A parent table next to a group of columns indicates foreign key. In SQL, foreign keys must reference their values from the primary key of the parent table. Many of the table diagrams contain duplicate columns (for example, sid shows up three times in Table W). These duplicates are included for illustrating the foreign key relationships, but in practice the duplicates are implemented as single columns participating in multiple foreign keys.

This section also includes Java code chunks. Double-angle brackets enclose the chunk name, used to refer to the chunk in other parts of the document. Anything after the equals sign and before the "at" sign is live code. Noweb is used to compile the code chunks into correct Java source code.

### 2.4.1 Road Network Tables (Tables V and E)

Each vertex $v \in \mathcal{V}$ is stored in Table V along with its coordinates $V(v)$ while each edge $(a, b) \in \mathcal{E}$ is stored in Table E along with its weight $d(a, b)$ and speed limit $\nu^{\max}(a, b)$. Table V thus has three columns, storing $v$ in primary key column v (`P1`) and its coordinates in column lng and lat. Likewise, Table E has four columns, storing $a$ and $b$ in column v1 and v2, $d(a, b)$ in column dd, and $\nu^{\max}(a, b)$ in column nu. The four columns together form the primary key (`P2`) in order to be referenced by later tables. Foreign keys on v1 (`F1`) and v2 (`F2`) referencing Table V validate that $a$ and $b$ are actual vertices.

| Table V (Vertices) | |
|---|---|
| Column | Description |
| *v* | Vertex $v \in \mathcal{V}$ |
| lng<br>lat | Vertex coordinate $V(v)$ |

| Table E (Edges) | | |
|---|---|---|
| Column | Parent | Description |
| *v1* | Table V | Edge $(a, b) \in \mathcal{E}$ |
| *v2* | Table V | |
| *dd* | | Weight $d(a, b)$ |
| *nu* | | Max. speed $\nu^{\max}(a, b)$ |

Jargo considers vertex 0 to be a dummy vertex where any edged formed by 0 has no weight. To implement the dummy vertex, constraint (`C11`) is added that states `dd` must be 0 if either `v1` or `v2` is 0.

Here are the SQL statements to construct the tables.

22 ⟨*Create Table V statement* 22⟩≡ (37c)
```
"CREATE TABLE V ("
  + "v   int  CONSTRAINT P1 PRIMARY KEY,"
  + "lng int  CONSTRAINT C1 NOT NULL,"
  + "lat int  CONSTRAINT C2 NOT NULL,"
  + "CONSTRAINT C3 CHECK (lng BETWEEN -1800000000 AND 1800000000),"
  + "CONSTRAINT C4 CHECK (lat BETWEEN  -900000000 AND  900000000)"
  + ")"
```

[1]ISO/IEC 9075

23a ⟨*Create Table E statement* 23a⟩≡ (37c)
```
"CREATE TABLE E ("
  + "v1  int  CONSTRAINT C5 NOT NULL,"
  + "v2  int  CONSTRAINT C6 NOT NULL,"
  + "dd  int  CONSTRAINT C7 NOT NULL,"
  + "nu  int  CONSTRAINT C8 NOT NULL,"
  + "CONSTRAINT F1 FOREIGN KEY (v1) REFERENCES V (v),"
  + "CONSTRAINT F2 FOREIGN KEY (v2) REFERENCES V (v),"
  + "CONSTRAINT P2 PRIMARY KEY (v1, v2, dd, nu),"
  + "CONSTRAINT C9 CHECK (nu >= 0),"
  + "CONSTRAINT C10 CHECK (v1 <> v2),"
  + "CONSTRAINT C11 CHECK ("
  + "  CASE WHEN v1 = 0 OR v2 = 0"
  + "    THEN dd = 0"
  + "    ELSE dd > 0"
  + "  END"
  + ")"
  + ")"
```

### 2.4.2 User Tables (Table UQ, UE, UL, UO, UD, and UB)

To allow other tables to reference specific user components, the user relation is partitioned into five 2-column tables, UQ, UE, UL, UO, and UD, by taking projections on the respective `q`, `e`, `l`, `o`, and `d` components. Each row is a key-value pair, storing a unique `uid` for user identification as the key alongside the component value, and each row is also its own primary key. A sixth table UB is introduced to store base costs for computing $D^{\text{base}}$ and $\rho^{\text{distance}}$. Table UO and UD can be referenced to Table V to validate against property P2 and rule P4.

| User Tables | | |
|---|---|---|
| Table | Columns | Description |
| UQ | *uid*, *val* | User load $u_{\mathtt{q}}$ |
| UE | *uid*, *val* | User early time $u_{\mathtt{e}}$ |
| UL | *uid*, *val* | User late time $u_{\mathtt{l}}$ |
| UO | *uid*, *val* | User origin $u_{\mathtt{o}}$ |
| UD | *uid*, *val* | User destination $u_{\mathtt{d}}$ |
| UB | *uid*, *val* | User base cost $d_u$ |

23b ⟨*Create Table UQ statement* 23b⟩≡ (37c)
```
"CREATE TABLE UQ ("
  + "uid int  CONSTRAINT C12 NOT NULL,"
  + "uq  int  CONSTRAINT C13 NOT NULL,"
  + "CONSTRAINT C14 UNIQUE (uid),"
  + "CONSTRAINT C15 CHECK (uq != 0),"
  + "CONSTRAINT P3 PRIMARY KEY (uid, uq)"
  + ")"
```

23c ⟨*Create Table UE statement* 23c⟩≡ (37c)
```
"CREATE TABLE UE ("
  + "uid int  CONSTRAINT C16 NOT NULL,"
  + "ue  int  CONSTRAINT C17 NOT NULL,"
  + "CONSTRAINT C18 CHECK (ue BETWEEN 0 AND 86400000),"
  + "CONSTRAINT C19 UNIQUE (uid),"
  + "CONSTRAINT P4 PRIMARY KEY (uid, ue)"
  + ")"
```

23d ⟨*Create Table UL statement* 23d⟩≡ (37c)
```
"CREATE TABLE UL ("
  + "uid int  CONSTRAINT C20 NOT NULL,"
  + "ul  int  CONSTRAINT C21 NOT NULL,"
  + "CONSTRAINT C22 UNIQUE (uid),"
  + "CONSTRAINT C23 CHECK (ul BETWEEN 0 AND 86400000),"
  + "CONSTRAINT P5 PRIMARY KEY (uid, ul)"
  + ")"
```

24a ⟨*Create Table UO statement* 24a⟩≡ (37c)

```
  "CREATE TABLE UO ("
    + "uid int  CONSTRAINT C24 NOT NULL,"
    + "uo  int  CONSTRAINT C25 NOT NULL,"
    + "CONSTRAINT F3 FOREIGN KEY (uo) REFERENCES V (v),"
    + "CONSTRAINT C26 UNIQUE (uid),"
    + "CONSTRAINT P6 PRIMARY KEY (uid, uo)"
    + ")"
```

24b ⟨*Create Table UD statement* 24b⟩≡ (37c)

```
  "CREATE TABLE UD ("
    + "uid int  CONSTRAINT C27 NOT NULL,"
    + "ud  int  CONSTRAINT C28 NOT NULL,"
    + "CONSTRAINT F4 FOREIGN KEY (ud) REFERENCES V (v),"
    + "CONSTRAINT C29 UNIQUE (uid),"
    + "CONSTRAINT P7 PRIMARY KEY (uid, ud)"
    + ")"
```

24c ⟨*Create Table UB statement* 24c⟩≡ (37c)

```
  "CREATE TABLE UB ("
    + "uid int  CONSTRAINT C30 NOT NULL,"
    + "ub  int  CONSTRAINT C31 NOT NULL,"
    + "CONSTRAINT C32 CHECK (ub >= 0),"
    + "CONSTRAINT C33 UNIQUE (uid),"
    + "CONSTRAINT P8 PRIMARY KEY (uid, ub)"
    + ")"
```

### 2.4.3 Routes Table (Table W)

Table W has eight columns, sid, se, t1, v1, t2, v2, dd, and nu. The s, t, and v components of $\mathcal{X}$ are stored in the (sid, t2, v2) columns. By definition, the sequence of vertices in a route must form a path and the speed of adjacent waypoints cannot exceed the limit $\nu^{\max}$. To enforce these rules, the predecessor waypoint is stored in the (sid, t1, v1) columns. The (v1, v2) columns can thus identify an edge. Columns dd and nu are added to store the weight and speed limit on the edge, and (v1, v2, dd, nu) is referenced by foreign key to Table E (F19) to validate the values. A row-level CHECK constraint (C56) validates that the speed dd/(t2 − t1) is not greater than the maximum free-flow speed, nu.

| Table W (Routes) | | |
|---|---|---|
| Col. | Parent | Description |
| *sid* | Table S | Identification for server $s \in \mathcal{S}$ |
| sid<br>se | Table UE | Server early time $s_{\mathsf{e}}$ |
| sid<br>t1<br>v1 | Table W | Predecessor waypoint $w_{i-1}$ |
| *t2*<br>*v2* | | Waypoint $w_i$ |
| v1<br>v2<br>dd<br>nu | Table E | Properties of edge $(\pi_{\mathsf{v}}(w_{i-1}), \pi_{\mathsf{v}}(w_i))$ |

The below items are easily implemented in SQL and establish that each (sid, t1, v1) is indeed the predecessor to (sid, t2, v2) in the same row (refer to the SQL statements below):

1. The predecessor (sid, t1, v1) must reference an existing waypoint (sid, t2, v2) from the table (F20);
2. Out of all rows, (sid, t1) must be unique and (sid, t2) must be unique (C54, C55);
3. Column t2 and v2 cannot be null (C52, C53);
4. Unless t2 is equal to the server's early time, t1 cannot be null and it must be less than t2, otherwise t1, v1, dd, and nu must all be null (C56).

The (sid, t2, v2) columns are the primary key (P11) in order to allow the self-referencing foreign key in the first item. The last item handles the case where the first waypoint in a server's route has no predecessor. Only in this case are t1, v1, dd, and nu are allowed to be null. From rule R1, the first waypoint is detected by checking if t2 is equal to the server's early time, stored in column se. The (sid, se) columns are referenced to UE to validate the early time (F18).

25a ⟨*Create Table W statement* 25a⟩≡ (37c)
```
  "CREATE TABLE W ("
    + "sid int  CONSTRAINT C50 NOT NULL,"
    + "se  int  CONSTRAINT C51 NOT NULL,"
    + "t1  int  ,"
    + "v1  int  ,"
    + "t2  int  CONSTRAINT C52 NOT NULL,"
    + "v2  int  CONSTRAINT C53 NOT NULL,"
    + "dd  int ,"
    + "nu  int ,"
    + "CONSTRAINT P11 PRIMARY KEY (sid, t2, v2),"
    + "CONSTRAINT F17 FOREIGN KEY (sid) REFERENCES S,"
    + "CONSTRAINT F18 FOREIGN KEY (sid, se) REFERENCES UE (uid, ue),"
    + "CONSTRAINT F19 FOREIGN KEY (v1, v2, dd, nu) REFERENCES E,"
    + "CONSTRAINT F20 FOREIGN KEY (sid, t1, v1) REFERENCES W (sid, t2, v2) INITIALLY DEFERRED,"
    + "CONSTRAINT C54 UNIQUE (sid, t1),"
    + "CONSTRAINT C55 UNIQUE (sid, t2),"
    + "CONSTRAINT C56 CHECK ("
    + "  CASE WHEN t1 IS NULL"
    + "    THEN t2 = se AND v1 IS NULL AND dd IS NULL AND nu IS NULL"
    + "    ELSE dd/(t2-t1) <= nu AND t1 < t2"
    + "  END"
    + ")"
    + ")"
```

### 2.4.4 Labels Table (Table PD)

Table PD (for "pick-ups and drop-offs") contains four columns, sid, t2, v2, and rid. The (sid, t2, v2) columns reference Table W (F23), and the rid column indicates the label on that waypoint. Each row is its own primary key (P12) in order to be referenced by the CPD constraint table. A waypoint can have multiple labels simply by listing the waypoint multiple times with different values of rid.

| Table PD (Pick-up and Drop-off Labels) | | |
|---|---|---|
| Col. | Parent | Description |
| *sid* *t2* *v2* | Table W | Waypoint $w_i$ (schedule element $b_j$) |
| *rid* | Table R | Identification for request $r \in \mathcal{R}$ |

25b ⟨*Create Table PD statement* 25b⟩≡ (37c)
```
  "CREATE TABLE PD ("
    + "sid int  CONSTRAINT C57 NOT NULL,"
    + "t2  int  CONSTRAINT C58 NOT NULL,"
    + "v2  int  CONSTRAINT C59 NOT NULL,"
    + "rid int  CONSTRAINT C60 NOT NULL,"
    + "CONSTRAINT P12 PRIMARY KEY (sid, t2, v2, rid),"
    + "CONSTRAINT F21 FOREIGN KEY (sid) REFERENCES S,"
    + "CONSTRAINT F22 FOREIGN KEY (rid) REFERENCES R,"
    + "CONSTRAINT F23 FOREIGN KEY (sid, t2, v2) REFERENCES W INITIALLY DEFERRED"
    + ")"
```

### 2.4.5 User Constraint Tables (Tables S and R)

Table S and Table R enforce the remaining user constraints. Both tables have six columns, one for each of uq, ue, ul, uo, ud, and ub, to store user data. A seventh column stores the user identifier as the primary key. The identifier is stored in the sid column for Table S and the rid column for Table R. Each (sid, column) or (rid, column) pair references the corresponding user property table, for example (sid, uq) references Table UQ.

Properties P1 and P3 that could not be enforced in the user tables are now enforced through simple constraints on S and R. A CHECK constraint validates that uq is less than 0 in Table S (C40), and another CHECK constraint validates it is greater than 0 in Table R (C48), corresponding to servers and requests (property P1). Likewise, a CHECK constraint validates that ue is less than ul (C41, C49) (property P3). None of the columns can be null to prevent incomplete users.

| User Constraint Tables | |
|---|---|
| Table | Columns |
| Table S | *sid*, sq, se, sl, so, sd, sb |
| Table R | *rid*, rq, re, rl, ro, rd, rb |

26a ⟨*Create Table S statement* 26a⟩≡ (37c)
```
  "CREATE TABLE S ("
    + "sid int  CONSTRAINT P9 PRIMARY KEY,"
    + "sq  int  CONSTRAINT C34 NOT NULL,"
    + "se  int  CONSTRAINT C35 NOT NULL,"
    + "sl  int  CONSTRAINT C36 NOT NULL,"
    + "so  int  CONSTRAINT C37 NOT NULL,"
    + "sd  int  CONSTRAINT C38 NOT NULL,"
    + "sb  int  CONSTRAINT C39 NOT NULL,"
    + "CONSTRAINT C40 CHECK (sq < 0),"
    + "CONSTRAINT F5 FOREIGN KEY (sid, sq) REFERENCES UQ (uid, uq),"
    + "CONSTRAINT F6 FOREIGN KEY (sid, se) REFERENCES UE (uid, ue),"
    + "CONSTRAINT F7 FOREIGN KEY (sid, sl) REFERENCES UL (uid, ul),"
    + "CONSTRAINT F8 FOREIGN KEY (sid, so) REFERENCES UO (uid, uo),"
    + "CONSTRAINT F9 FOREIGN KEY (sid, sd) REFERENCES UD (uid, ud),"
    + "CONSTRAINT F10 FOREIGN KEY (sid, sb) REFERENCES UB (uid, ub),"
    + "CONSTRAINT C41 CHECK (se < sl)"
    + ")"
```

26b ⟨*Create Table R statement* 26b⟩≡ (37c)
```
  "CREATE TABLE R ("
    + "rid int  CONSTRAINT P10 PRIMARY KEY,"
    + "rq  int  CONSTRAINT C42 NOT NULL,"
    + "re  int  CONSTRAINT C43 NOT NULL,"
    + "rl  int  CONSTRAINT C44 NOT NULL,"
    + "ro  int  CONSTRAINT C45 NOT NULL,"
    + "rd  int  CONSTRAINT C46 NOT NULL,"
    + "rb  int  CONSTRAINT C47 NOT NULL,"
    + "CONSTRAINT C48 CHECK (rq > 0),"
    + "CONSTRAINT F11 FOREIGN KEY (rid, rq) REFERENCES UQ (uid, uq),"
    + "CONSTRAINT F12 FOREIGN KEY (rid, re) REFERENCES UE (uid, ue),"
    + "CONSTRAINT F13 FOREIGN KEY (rid, rl) REFERENCES UL (uid, ul),"
    + "CONSTRAINT F14 FOREIGN KEY (rid, ro) REFERENCES UO (uid, uo),"
    + "CONSTRAINT F15 FOREIGN KEY (rid, rd) REFERENCES UD (uid, ud),"
    + "CONSTRAINT F16 FOREIGN KEY (rid, rb) REFERENCES UB (uid, ub),"
    + "CONSTRAINT C49 CHECK (re < rl)"
    + ")"
```

### 2.4.6 Route Endpoint Constraints Table (Table CW)

Table CW stores the start and end waypoints of each server route. The table has nine columns, sid, se, sl, so, sd, ts, vs, te, and ve. The start waypoint is stored in (sid, ts, vs) and the end waypoint is stored in (sid, te, ve). Both of these groups reference the (sid, t2, v2) columns in Table W (F29, F30). The sid column is set to be UNIQUE (C70) to prevent a server from being listed multiple times and having "multiple" start and end waypoints. Rule R1 is enforced by adding the server's early and late times into columns se and sl, referencing (sid, se) to UE (F25) and (sid, sl) to UL (F26). A CHECK constraint validates the start time ts equals se (C71) and another one validates the end time te is not beyond sl (C72). Rule 2 is enforced by adding the server's origin and destination into columns so and sd, referencing (sid, so) to UO (F27) and (sid, sd) to UD (F28). Likewise, constraint C71 validates the start location vs equals so and C72 validates the end location ve equals sd.

| Table CW (Route Endpoint Constraints) | | |
|---|---|---|
| Col. | Parent | Description |
| sid<br>se | Table UE | Server early time $s_e$ |
| sid<br>sl | Table UL | Server late time $s_l$ |
| sid<br>so | Table UO | Server origin $s_o$ |
| sid<br>sd | Table UD | Server destination $s_d$ |
| *sid*<br>*ts*<br>vs | Table W | Server pickup$(\mathcal{X}, s)$ |
| sid<br>*te*<br>ve | Table W | Server dropoff$(\mathcal{X}, s)$ |

27 ⟨*Create Table CW statement* 27⟩≡ (37c)

```
  "CREATE TABLE CW ("
    + "sid int  CONSTRAINT C61 NOT NULL,"
    + "se  int  CONSTRAINT C62 NOT NULL,"
    + "sl  int  CONSTRAINT C63 NOT NULL,"
    + "so  int  CONSTRAINT C64 NOT NULL,"
    + "sd  int  CONSTRAINT C65 NOT NULL,"
    + "ts  int  CONSTRAINT C66 NOT NULL,"
    + "vs  int  CONSTRAINT C67 NOT NULL,"
    + "te  int  CONSTRAINT C68 NOT NULL,"
    + "ve  int  CONSTRAINT C69 NOT NULL,"
    + "CONSTRAINT C70 UNIQUE (sid),"
    + "CONSTRAINT P13 PRIMARY KEY (sid, ts, te),"
    + "CONSTRAINT F24 FOREIGN KEY (sid) REFERENCES S,"
    + "CONSTRAINT F25 FOREIGN KEY (sid, se) REFERENCES UE (uid, ue),"
    + "CONSTRAINT F26 FOREIGN KEY (sid, sl) REFERENCES UL (uid, ul),"
    + "CONSTRAINT F27 FOREIGN KEY (sid, so) REFERENCES UO (uid, uo),"
    + "CONSTRAINT F28 FOREIGN KEY (sid, sd) REFERENCES UD (uid, ud),"
    + "CONSTRAINT F29 FOREIGN KEY (sid, ts, vs) REFERENCES W (sid, t2, v2) INITIALLY DEFERRED,"
    + "CONSTRAINT F30 FOREIGN KEY (sid, te, ve) REFERENCES W (sid, t2, v2) INITIALLY DEFERRED,"
    + "CONSTRAINT C71 CHECK (ts = se),"
    + "CONSTRAINT C72 CHECK (vs = so),"
  //+ "CONSTRAINT C73 CHECK (te <= sl),"
    + "CONSTRAINT C74 CHECK (ve = sd),"
    + "CONSTRAINT C75 CHECK (ts < te)"
    + ")"
```

### 2.4.7 Label Constraints Table (Table CPD)

Table CPD enforces the pick-up and drop-off rules R5–R9. It contains twelve columns, sid, ts, te, tp, vp, td, vd, rid, re, rl, ro, and rd. The (sid, tp, vp, rid) and (sid, td, vd, rid) groups reference rows in Table PD (F34, F35) and represent pick-up and drop-off waypoints, respectively. Rules R5 and R9 are enforced by setting rid to UNIQUE (C86), in other words any request identified in rid has only one pick-up and drop-off pair. Rule R6 is enforced by adding columns for the request origin ro and destination rd and validating that pick-up vertex vp equals ro (C89) and drop-off vertex vd equals rd (C90). The (rid, ro) columns are referenced to UO (F38) and (rid, rd) are referenced to UD (F39). Rules R7 and R8 are enforced by simple CHECK constraints. Both tp and td are validated to be between request early time re and late time rl (C89, C90). The (rid, re) and (rid, rl) columns are added and referenced to UE and UL (F36, F37) for this purpose.

| Table CPD (Pick-up and Drop-off Constraints) | | |
|---|---|---|
| Col. | Parent | Description |
| sid<br>ts<br>te | Table CW | Server start and end times<br>$\pi_{\mathtt{t}}(\text{pickup}(\mathcal{X}, s))$,<br>$\pi_{\mathtt{t}}(\text{dropoff}(\mathcal{X}, s))$ |
| *sid*<br>*tp*<br>vp<br>rid | Table PD | Request pickup$(\mathcal{X}, r)$ |
| sid<br>*td*<br>vd<br>*rid* | Table PD | Request dropoff$(\mathcal{X}, r)$ |
| rid<br>re | Table UE | Request early time $r_{\mathtt{e}}$ |
| rid<br>rl | Table UL | Request late time $r_{\mathtt{l}}$ |
| rid<br>ro | Table UO | Request origin $r_{\mathtt{o}}$ |
| rid<br>rd | Table UD | Request destination $r_{\mathtt{d}}$ |

So far, nothing prevents tp and td from falling outside the server's start and end times. These times are thus added into (sid, ts, te) columns, referenced to Table CW (F33). Then, CHECK constraints can validate that tp and td are within the start time ts and the end time te (C87, C88).

28 ⟨*Create Table CPD statement* 28⟩≡ (37c)

```
"CREATE TABLE CPD ("
  + "sid int  CONSTRAINT C76 NOT NULL,"
  + "ts  int  CONSTRAINT C77 NOT NULL,"
  + "te  int  CONSTRAINT C78 NOT NULL,"
  + "tp  int  CONSTRAINT C79 NOT NULL,"
  + "vp  int  CONSTRAINT C80 NOT NULL,"
  + "td  int  CONSTRAINT C81 NOT NULL,"
  + "vd  int  CONSTRAINT C82 NOT NULL,"
  + "rid int  CONSTRAINT C83 NOT NULL,"
  + "re  int  CONSTRAINT C84 NOT NULL,"
  + "rl  int  CONSTRAINT C85 NOT NULL,"
  + "ro  int  CONSTRAINT C86 NOT NULL,"
  + "rd  int  CONSTRAINT C87 NOT NULL,"
  + "CONSTRAINT C88 UNIQUE (rid),"
  + "CONSTRAINT P14 PRIMARY KEY (sid, tp, td, rid),"
  + "CONSTRAINT F31 FOREIGN KEY (sid) REFERENCES S,"
  + "CONSTRAINT F32 FOREIGN KEY (rid) REFERENCES R,"
  + "CONSTRAINT F33 FOREIGN KEY (sid, ts, te) REFERENCES CW (sid, ts, te) "
  + "  INITIALLY DEFERRED,"
  + "CONSTRAINT F34 FOREIGN KEY (sid, tp, vp, rid) REFERENCES PD (sid, t2, v2, rid) "
  + "  INITIALLY DEFERRED,"
  + "CONSTRAINT F35 FOREIGN KEY (sid, td, vd, rid) REFERENCES PD (sid, t2, v2, rid) "
  + "  INITIALLY DEFERRED,"
  + "CONSTRAINT F36 FOREIGN KEY (rid, re) REFERENCES UE (uid, ue),"
  + "CONSTRAINT F37 FOREIGN KEY (rid, rl) REFERENCES UL (uid, ul),"
  + "CONSTRAINT F38 FOREIGN KEY (rid, ro) REFERENCES UO (uid, uo),"
  + "CONSTRAINT F39 FOREIGN KEY (rid, rd) REFERENCES UD (uid, ud),"
  + "CONSTRAINT C89a CHECK (tp >= ts),"
//  + "CONSTRAINT C89b CHECK (td <= te),"
  + "CONSTRAINT C89c CHECK (tp < td),"
  + "CONSTRAINT C91 CHECK (tp >= re),"
  + "CONSTRAINT C92 CHECK (vp  = ro),"
//+ "CONSTRAINT C93 CHECK (td <= rl)",
  + "CONSTRAINT C94 CHECK (vd  = rd)"
  + ")"
```

| Table CQ (Load Constraints) | | |
|---|---|---|
| Col. | Parent | Description |
| sid sq | Table UQ | Server load $s_q$ |
| sid se | Table UE | Server early time $s_e$ |
| sid t1 q1 o1 | Table CQ | Load burden $\mathcal{Q}(\mathcal{X}, s, \mathrm{t1})$ up to order o1 |
| *sid t2 q2 o2* | | Load burden $\mathcal{Q}(\mathcal{X}, s, \mathrm{t2})$ up to order o2 |
| sid t2 v2 rid | Table PD | Request pick-up or delivery waypoint |
| sid tp td rid | Table CPD | Request pick-up and delivery times $\pi_{\mathrm{t}}(\mathrm{pickup}(\mathcal{X}, r))$, $\pi_{\mathrm{t}}(\mathrm{dropoff}(\mathcal{X}, r))$ |
| rid rq | Table UQ | Request load $r_q$ |

### 2.4.8 Load Constraints Table (Table CQ)

Table CQ enforces the load rule R11. It has fourteen columns, sid, sq, se, t1, t2, v2, q1, q2, rid, rq, tp, td, o1, and o2. From Eq. 2.2, the load burden only changes at the times of waypoints labeled with a request. It increases when a waypoint corresponds to a customer pick-up and decreases when the waypoint corresponds to a customer drop-off. Each load-changing waypoint is stored in (sid, t2, v2, rid) and referenced to PD (F46). To determine if the waypoint is a customer pick-up or drop-off, the pick-up and drop-off times for rid are stored in (sid, tp, td, rid) and referenced to CPD (F47). If t2 = tp, then the waypoint represents a pick-up, otherwise it represents a drop-off. The load of the server and request are stored in (sid, sq) and (rid, rq), referenced to UQ (F44, F45).

To validate if the load burden is always within a server's capacity, CQ must keep track of every load change. It does so by storing the predecessor load in columns (sid, t1, q1, o1) next to the current load in columns (sid, t2, q2, o2). If the waypoint in the row is a pick-up, CQ validates that q1+rq = q2, otherwise that q1 − rq = q2 (C98). As repetitive load changes can occur at a single waypoint due to multiple pick-ups and drop-offs, the o1 and o2 columns are introduced to store a unique order number. This number increments by 1 for each pick-up or drop-off per server and can be handled by the application. Similar rules for establishing predecessor waypoints in Table W can be used to establish predecessor loads in CQ. Subsequently, (sid, t2, q2, o2) is set to be the primary key (P15) in order to allow a self-referencing foreign key on (sid, t1, q1, o1) (F42), and the server early time is stored in (sid, se) and referenced to UE (F43) in order to detect the first load change.

29 ⟨*Create Table CQ statement* 29⟩≡ (37c)
```
  "CREATE TABLE CQ ("
    + "sid int  CONSTRAINT C95 NOT NULL,"
    + "sq  int  CONSTRAINT C96 NOT NULL,"
    + "se  int  CONSTRAINT C97 NOT NULL,"
    + "t1  int  ,"
    + "t2  int  CONSTRAINT C98 NOT NULL,"
    + "v2  int  ,"
    + "q1  int  ,"
    + "q2  int  CONSTRAINT C99 NOT NULL,"
    + "rid int  ,"
    + "rq  int  ,"
    + "tp  int  ,"
    + "td  int  ,"
    + "o1  int  ,"
    + "o2  int  CONSTRAINT C100 NOT NULL,"
    + "CONSTRAINT C101 CHECK (o2 > 0),"
    + "CONSTRAINT P15 PRIMARY KEY (sid, t2, q2, o2),"
    + "CONSTRAINT F40 FOREIGN KEY (sid) REFERENCES S,"
    + "CONSTRAINT F41 FOREIGN KEY (rid) REFERENCES R,"
    + "CONSTRAINT F42 FOREIGN KEY (sid, t1, q1, o1) REFERENCES CQ (sid, t2, q2, o2)"
    + "  INITIALLY DEFERRED,"
    + "CONSTRAINT F43 FOREIGN KEY (sid, se) REFERENCES UE (uid, ue),"
    + "CONSTRAINT F44 FOREIGN KEY (sid, sq) REFERENCES UQ (uid, uq),"
    + "CONSTRAINT F45 FOREIGN KEY (rid, rq) REFERENCES UQ (uid, uq),"
    + "CONSTRAINT F46 FOREIGN KEY (sid, t2, v2, rid) REFERENCES PD INITIALLY DEFERRED,"
    + "CONSTRAINT F47 FOREIGN KEY (sid, tp, td, rid) REFERENCES CPD INITIALLY DEFERRED,"
    + "CONSTRAINT C102a CHECK (CASE WHEN t1 IS NULL THEN t2 = se END),"
    + "CONSTRAINT C102b CHECK (CASE WHEN t1 IS NULL THEN q2 = sq END),"
    + "CONSTRAINT C102c CHECK (CASE WHEN t1 IS NULL THEN o2 = 1 END),"
    + "CONSTRAINT C102d CHECK (CASE WHEN t1 IS NULL THEN q1 IS NULL END),"
    + "CONSTRAINT C102e CHECK (CASE WHEN t1 IS NULL THEN o1 IS NULL END),"
    + "CONSTRAINT C102f CHECK (CASE WHEN t1 IS NULL THEN rid IS NULL END),"
    + "CONSTRAINT C102g CHECK (CASE WHEN t1 IS NULL THEN rq IS NULL END),"
    + "CONSTRAINT C102h CHECK (CASE WHEN t1 IS NULL THEN tp IS NULL END),"
    + "CONSTRAINT C102i CHECK (CASE WHEN t1 IS NULL THEN td IS NULL END),"
    + "CONSTRAINT C102j CHECK (CASE WHEN t1 IS NOT NULL THEN q2 <= 0 END),"
    + "CONSTRAINT C102k CHECK (CASE WHEN t1 IS NOT NULL THEN o2 = o1 + 1 END),"
    + "CONSTRAINT C103 CHECK (CASE WHEN t2 = tp THEN q2 = q1 + rq END) INITIALLY DEFERRED,"
    + "CONSTRAINT C104 CHECK (CASE WHEN t2 = td THEN q2 = q1 - rq END) INITIALLY DEFERRED,"
    + "CONSTRAINT C105 UNIQUE (t2, v2, rid)"
    + ")"
```

### 2.4.9 Views

The user relation $\mathcal{U}$ can be formed by a union of Table S and R.

30a ⟨*Create View r_user statement* 30a⟩≡ (37c)

```
"CREATE VIEW r_user (uid, uq, ue, ul, uo, ud, ub) AS "
  + "SELECT * from S UNION SELECT * from R"
```

The server relation $\mathcal{X}$ can be constructed by joining the routes in Table W with the labels in CW and PD.

30b ⟨*Create View r_server statement* 30b⟩≡ (37c)

```
"CREATE VIEW r_server (sid, t, v, Ls, Lr) AS "
  + "SELECT W.sid, W.t2, W.v2, CW.sid, PD.rid "
  + "FROM W LEFT OUTER JOIN CW ON W.sid = CW.sid AND (W.t2 = CW.ts OR W.t2 = CW.te) "
  + "  LEFT OUTER JOIN PD ON W.sid = PD.sid AND W.t2 = PD.t2"
```

The cruising and service distances $D^{\text{cruise}}$ and $D^{\text{service}}$ require an auxilliary view. This view joins Table W with CQ in such a way that the distances in column dd of W can be aggregated based on whether there is load burden at the time of the waypoint.

30c ⟨*Create View f_distance_blocks statement* 30c⟩≡ (37c)

```
"CREATE VIEW f_distance_blocks (sid, wt1, wt2, wdd, cqsq, cqt1, cqt2, cqq1, cqq2) "
  + "AS SELECT W.sid, W.t1, W.t2, W.dd, CQ.sq, CQ.t1, CQ.t2, CQ.q1, CQ.q2 "
  + "FROM W LEFT OUTER JOIN CQ ON W.sid = CQ.sid and W.t2 > CQ.t1 and W.t2 <= CQ.t2 "
  + "WHERE W.dd IS NOT NULL"
```

Request status can also be obtained using an auxilliary view. This view lists the count of occurrences of a request in column rid of CQ, corresponding to the request status. Table CQ is used to get the counts over time. If the count is 0, it will not appear in the aggregation and the status for the request is "waiting".

30d ⟨*Create View f_status statement (Eq. 2.1)* 30d⟩≡ (37c)

```
"CREATE VIEW f_status (t, sid, rid, val) AS "
  + "SELECT a.t2, a.sid, a.rid, COUNT (b.rid) "
  + "FROM CQ AS a INNER JOIN CQ AS b ON a.t2 >= b.t2 "
  + "WHERE a.rid IS NOT NULL AND b.rid IS NOT NULL AND a.rid = b.rid "
  + "GROUP BY a.t2, a.sid, a.rid"
```

To list all assignments $\mathcal{A}$:

31a ⟨*Create View assignments statement (Eq. 2.3)* 31a⟩≡ (37c)
```
"CREATE VIEW assignments (t, sid, rid) AS "
  + "SELECT t, sid, rid FROM f_status WHERE val = 2 ORDER BY t ASC"
```

To list assigned requests $\mathcal{R}^{\text{ok}}$:

31b ⟨*Create View assignments_r statement (Eq. 2.4)* 31b⟩≡ (37c)
```
"CREATE VIEW assignments_r (t, rid) AS "
  + "SELECT t, rid FROM assignments"
```

To list service rate $\mu$:

31c ⟨*Create View service_rate statement (Eq. 2.8)* 31c⟩≡ (37c)
```
"CREATE VIEW service_rate (val) AS "
  + "SELECT CAST(CAST(A.NUM AS FLOAT) / CAST(A.DENOM AS FLOAT) * 10000 as INT)"
  + "FROM ( "
  + "SELECT (SELECT COUNT(*) FROM assignments_r) AS NUM, "
  + "       (SELECT COUNT(*) FROM R) AS DENOM "
  + "       FROM R FETCH FIRST ROW ONLY "
  + ") A"
```

To list base distance $D^{\text{base}}$:

31d ⟨*Create View dist_base statement (Eq. 2.9)* 31d⟩≡ (37c)
```
"CREATE VIEW dist_base (val) AS "
  + "SELECT SUM (ub) FROM UB"
```

To list travel distances $D$ of all servers:

31e ⟨*Create View dist_s_travel statement* 31e⟩≡ (37c)
```
"CREATE VIEW dist_s_travel (sid, val) AS "
  + "SELECT W.sid, SUM (COALESCE (dd, 0)) "
  + "FROM W JOIN CW ON w.sid = cw.sid AND (t2 BETWEEN ts AND te) "
  + "GROUP BY W.sid"
```

To list cruising distances $D^{\text{cruise}}$ of all servers:

31f ⟨*Create View dist_s_cruising statement (Eq. 2.12)* 31f⟩≡ (37c)
```
"CREATE VIEW dist_s_cruising (sid, val) AS "
  + "SELECT sid, SUM (wdd) FROM f_distance_blocks "
  + "WHERE cqq1 = cqsq OR cqq1 IS NULL GROUP BY sid"
```

To list service distances $D^{\text{service}}$ of all servers:

31g ⟨*Create View dist_s_service statement (Eq. 2.13)* 31g⟩≡ (37c)
```
"CREATE VIEW dist_s_service (sid, val) AS "
  + "SELECT sid, SUM (wdd) FROM f_distance_blocks "
  + "WHERE cqq1 > cqsq GROUP BY sid"
```

To list base distances $d$ of all servers:

31h ⟨*Create View dist_s_base statement* 31h⟩≡ (37c)
```
"CREATE VIEW dist_s_base (val) AS "
  + "SELECT SUM (sb) FROM S"
```

To list base distances $d$ of all requests:

31i ⟨*Create View dist_r_base statement* 31i⟩≡ (37c)
```
"CREATE VIEW dist_r_base (val) AS "
  + "SELECT SUM (rb) FROM R"
```

To list base distances $d$ of all unassigned requests:

31j ⟨*Create View dist_r_unassigned statement* 31j⟩≡ (37c)
```
"CREATE VIEW dist_r_unassigned (val) AS "
  + "SELECT SUM (rb) FROM R LEFT JOIN assignments_r "
  + "  ON R.rid = assignments_r.rid "
  + "WHERE assignments_r.rid IS NULL"
```

To list detour distances $D^{\text{detour}}$ of all requests:

31k ⟨*Create View dist_r_detour statement (Eq. 2.21)* 31k⟩≡ (37c)
```
"CREATE VIEW dist_r_detour (rid, val) AS "
  + "SELECT rid, val-ub FROM UB JOIN dist_r_transit ON uid = rid"
```

To list transit distances $D^{\text{transit}}$ of all requests:

32a ⟨*Create View dist_r_transit statement (Eq. 2.19)* 32a⟩≡ (37c)
```
"CREATE VIEW dist_r_transit (rid, val) AS "
  + "SELECT rid, SUM (COALESCE (dd, 0)) "
  + "FROM CPD JOIN W ON CPD.sid = W.sid AND CPD.tp < W.t2 AND W.t2 <= CPD.td "
  + "GROUP BY rid"
```

To list travel duration $\delta$ of all servers:

32b ⟨*Create View dur_s_travel statement* 32b⟩≡ (37c)
```
"CREATE VIEW dur_s_travel (sid, val) AS "
  + "SELECT sid, te - ts FROM CW"
```

To list service duration of all servers:

32c ⟨*Create View dur_s_service statement* 32c⟩≡ (37c)
```
"CREATE VIEW dur_s_service (sid, val) AS "
  + "SELECT sid, sum (t2 - t1) FROM CQ WHERE Q1 <> SQ GROUP BY sid"
```

To list pick-up delay $\delta^{\text{pickup}}$ of all requests:

32d ⟨*Create View dur_r_pickup statement (Eq. 2.18)* 32d⟩≡ (37c)
```
"CREATE VIEW dur_r_pickup (rid, val) AS "
  + "SELECT rid, tp - re FROM CPD"
```

To list transit durations $\delta^{\text{transit}}$ of all requests:

32e ⟨*Create View dur_r_transit statement (Eq. 2.20)* 32e⟩≡ (37c)
```
"CREATE VIEW dur_r_transit (rid, val) AS "
  + "SELECT rid, td - tp FROM CPD"
```

To list travel durations $\delta^{\text{travel}}$ of all requests:

32f ⟨*Create View dur_r_travel statement (Eq. 2.23)* 32f⟩≡ (37c)
```
"CREATE VIEW dur_r_travel (rid, val) AS "
  + "SELECT rid, td - re FROM CPD"
```

To list departure times $t^{\text{depart}}$ of all requests:

32g ⟨*Create View t_r_depart statement (Eq. 2.16)* 32g⟩≡ (37c)
```
"CREATE VIEW t_r_depart (rid, val) AS "
  + "SELECT rid, tp FROM CPD"
```

To list departure times $t^{\text{depart}}$ of all servers:

32h ⟨*Create View t_s_depart statement (Eq. 2.16)* 32h⟩≡ (37c)
```
"CREATE VIEW t_s_depart (sid, val) AS "
  + "SELECT sid, ts FROM CW"
```

To list arrival times $t^{\text{arrive}}$ of all requests:

32i ⟨*Create View t_r_arrive statement (Eq. 2.17)* 32i⟩≡ (37c)
```
"CREATE VIEW t_r_arrive (rid, val) AS "
  + "SELECT rid, td FROM CPD"
```

To list arrival times $t^{\text{arrive}}$ of all servers:

32j ⟨*Create View t_s_arrive statement (Eq. 2.17)* 32j⟩≡ (37c)
```
"CREATE VIEW t_s_arrive (sid, val) AS "
  + "SELECT sid, te FROM CW"
```

To list time window violations of all servers:

32k ⟨*Create View violations_t_s* 32k⟩≡ (37c)
```
"CREATE VIEW violations_t_s (sid, val) AS "
  + "SELECT sid, te - sl FROM CW WHERE te - sl > 0"
```

To list time window violations of all requests:

32l ⟨*Create View violations_t_r* 32l⟩≡ (37c)
```
"CREATE VIEW violations_t_r (rid, val) AS "
  + "SELECT rid, td - rl FROM CPD WHERE td - rl > 0"
```

# Chapter 3

# Jargo Simulator

This chapter contains the code for the Jargo library. As in the previous section, double-angle brackets indicate a chunk of live Java code, used in other parts of the document. Noweb is used to compile the code chunks into correct Java source code.

## 3.1 Overview

This section presents an overview of all of Jargo's library methods, organized by function and class. Tables 3.1 and 3.2 describe the function and class groupings. The number in parentheses next to each method name indicates the number of parameters. Go to the page number next to a method to jump to the summary and source code.

| Ch. | Functional Group | Description |
|---|---|---|
| Ch. 3.2 | Administration | Manage classes and the simulation lifecycle |
| Ch. 3.3 | Read Methods | Retrieve direct or derived values from the simulation state |
| | Cached Read Methods | Retrieve values from cache instead of from Derby (overloads of normal read methods) |
| Ch. 3.4 | Write Methods | Push new values into the simulation state |
| Ch. 3.5 | G-tree Methods | Interact with G-tree spatial index |
| | Special Methods | Specific to a class (see individual chapters on classes) |

**Table 3.1:** ***Method Functional Groupings***

| Ch. | Class | Description |
|---|---|---|
| Ch. 3.6 | Storage | Provides direct access to Jargo's underlying Derby database containing the ridesharing simulation state |
| Ch. 3.7 | Controller | Manages the simulation lifecycle |
| Ch. 3.8 | Communicator | Provides a client-facing subset of Storage functionality |
| Ch. 3.9 | Client | Provides overrideable functionality for serving ridesharing requests |
| Ch. 3.10 | Traffic | Provides overrideable functionality for returning speeds in the road network |
| Ch. 3.11 | Tools | Provides convenience methods |

**Table 3.2:** ***Class Descriptions***

## 3.2 Administration

These methods manage the classes, the simulation, and the database connections.

### 3.2.1 Packages

The Jargo library has two packages:

- The `sim` package is for core classes.

35a ⟨*Package:* `sim` 35a⟩≡ (63–65 129a 134a 148a 150a 153 155a 157a)
```
package com.github.jargors.sim;
```

- The `ui` package is for the evaluators.

35b ⟨*Package:* `ui` 35b⟩≡ (165a 170a 172a)
```
package com.github.jargors.ui;
```

### 3.2.2 Chunks

These chunks perform some common “atomic” tasks used in multiple methods.

#### Open `conn`

35c ⟨*Open* `conn` 35c⟩≡ (35–37 40a 67–70 72–98 100–112 120–24)
```
Connection conn = DriverManager.getConnection(CONNECTIONS_POOL_URL)
```

#### Set statement values

35d ⟨*Set statement values* 35d⟩≡ (55b 56a)
```
for (int i = 0; i < values.length; i++) {
  if (values[i] == null) {
    p.setNull((i + 1), java.sql.Types.INTEGER);
  } else {
    p.setInt ((i + 1), values[i]);
  }
}
```

#### Cache server distance

35e ⟨*Cache server distance* 35e⟩≡ (43d 125a)
```
try (⟨Open conn 35c⟩) {
  this.distance_servers.put(sid, this.PSQuery(conn, "S104", 1, sid)[0]);
} catch (SQLException e) {
  throw e;
}
```
Uses `PSQuery` 56a and `S104` 50j.

## Cache cruising distance and duration

36a ⟨*Cache cruising distance and duration* 36a⟩≡ (43d 125a)

```
  try (⟨Open conn 35c⟩) {
    int sum = 0;
    int dur = 0;
    int[] output = this.PSQuery(conn, "S153", 2, sid);
    if (output.length > 0) {
      int ts = output[0];
      for (int i = 0; i < output.length - 1; i++) {
        final int t1 = output[(i + 0)];
        final int t2 = output[(i + 1)];
        sum += this.PSQuery(conn, "S154", 1, sid, t1, t2)[0];
        dur += (t2 - t1);
      }
      final int tt = this.PSQuery(conn, "S155", 1, sid)[0];
      final int te = this.PSQuery(conn, "S145", 2, sid)[0];
      sum += this.DBQueryServerDistanceRemaining(sid, tt)[0];
      dur += (te - tt);
      this.distance_servers_cruising.put(sid, sum);
      this.duration_servers_cruising.put(sid, dur);
      this.duration_servers.put(sid, (te - ts));
    } else {
      final int te = this.PSQuery(conn, "S145", 2, sid)[0];
      final int ts = this.PSQuery(conn, "S156", 2, sid)[0];
      sum = this.DBQueryServerDistanceRemaining(sid, ts)[0];
      this.distance_servers_cruising.put(sid, sum);
      this.duration_servers_cruising.put(sid, (te - ts));
      this.duration_servers.put(sid, (te - ts));
    }
  } catch (SQLException e) {
    throw e;
  }
```

Uses `DBQueryServerDistanceRemaining` 87c, `PSQuery` 56a, `S145` 53f, `S153` 53m, `S154` 53n, `S155` 54a, and `S156` 54b.

## Cache request transit distance and duration

36b ⟨*Cache request transit distance and duration* 36b⟩≡ (43d 125a)

```
  try (⟨Open conn 35c⟩) {
    this.distance_requests_transit.put(r, this.DBQueryRequestDistanceTransit(r, false)[0]);
    this.duration_requests_transit.put(r, this.DBQueryRequestDurationTransit(r, false)[0]);
    this.duration_requests_travel .put(r, this.DBQueryRequestDurationTravel (r, false)[0]);
    this.duration_requests_pickup .put(r, this.DBQueryRequestDurationPickup (r, false)[0]);
  } catch (SQLException e) {
    throw e;
  }
```

Uses `DBQueryRequestDistanceTransit` 76a, `DBQueryRequestDurationPickup` 76b, `DBQueryRequestDurationTransit` 77a, and `DBQueryRequestDurationTravel` 77b.

## Construct exception

36c ⟨*Construct empty exception* 36c⟩≡ (63–65)

```
  () { }
```

36d ⟨*Construct message exception* 36d⟩≡ (63–65)

```
  (String message) { super(message); }
```

36e ⟨*Construct cause exception* 36e⟩≡ (63–65)

```
  (Throwable cause) { super(cause); }
```

36f ⟨*Construct message and cause exception* 36f⟩≡ (63–65)

```
  (String message, Throwable cause) { super(message, cause); }
```

### 3.2.3 Methods: Administration

**JargoInstanceNew(0)**

Creates a new database instance.

**Parameters:** None.
**Returns:** Nothing.
**Side Effects:** Assigns new objects to `connection_factory`, `poolableconnection_factory`, `pool`, `driver`.
**Throws:**

- `SQLException`: if database failure is encountered
- `ClassNotFoundException`: if Derby driver cannot be loaded

**Wrappers:**

- `instanceNew`(0)

37a ⟨*Admin: JargoInstanceNew(0)* 37a⟩≡ (133b)

```
  void JargoInstanceNew() throws SQLException {
    try {
      this.JargoSetupDriver();
    } catch (SQLException e) {
      throw e;
    } catch (ClassNotFoundException e) {
      System.err.println("Fatal exception");
      e.printStackTrace();
      System.exit(1);
    }
  }
```

Defines:
  `JargoInstanceNew`, used in chunk 37.
Uses `JargoSetupDriver` 44b.

37b ⟨*Admin: instanceNew(0)* 37b⟩≡ (141c)

```
  void instanceNew() throws SQLException {
    this.storage.JargoInstanceNew();
  }
```

Defines:
  `instanceNew`, used in chunks 167b and 192e.
Uses `JargoInstanceNew` 37a.

□

**JargoInstanceInitialize(0)**

Creates the Jargo schema into the existing instance.

**Parameters:** None.
**Returns:** Nothing.
**Side Effects:** Modifies the database.
**Throws:**

- `SQLException`: if database failure is encountered

**Wrappers:**

- `instanceInitialize`(0)

37c ⟨*Admin: JargoInstanceInitialize(0)* 37c⟩≡ (133b)

```
  void JargoInstanceInitialize() {
    try (⟨Open conn 35c⟩) {
      Statement stmt = conn.createStatement();
      stmt.clearBatch();
      stmt.addBatch(⟨Create Table V statement 22⟩);
      stmt.addBatch(⟨Create Table E statement 23a⟩);
      stmt.addBatch(⟨Create Table UQ statement 23b⟩);
      stmt.addBatch(⟨Create Table UE statement 23c⟩);
      stmt.addBatch(⟨Create Table UL statement 23d⟩);
      stmt.addBatch(⟨Create Table UO statement 24a⟩);
      stmt.addBatch(⟨Create Table UD statement 24b⟩);
      stmt.addBatch(⟨Create Table UB statement 24c⟩);
      stmt.addBatch(⟨Create Table S statement 26a⟩);
      stmt.addBatch(⟨Create Table R statement 26b⟩);
      stmt.addBatch(⟨Create Table W statement 25a⟩);
      stmt.addBatch(⟨Create Table PD statement 25b⟩);
      stmt.addBatch(⟨Create Table CW statement 27⟩);
      stmt.addBatch(⟨Create Table CPD statement 28⟩);
      stmt.addBatch(⟨Create Table CQ statement 29⟩);
      stmt.addBatch(⟨Create View r_user statement 30a⟩);
      stmt.addBatch(⟨Create View r_server statement 30b⟩);
      stmt.addBatch(⟨Create View f_distance_blocks statement 30c⟩);
      stmt.addBatch(⟨Create View f_status statement (Eq. 2.1) 30d⟩);
      stmt.addBatch(⟨Create View assignments statement (Eq. 2.3) 31a⟩);
      stmt.addBatch(⟨Create View assignments_r statement (Eq. 2.4) 31b⟩);
      stmt.addBatch(⟨Create View service_rate statement (Eq. 2.8) 31c⟩);
      stmt.addBatch(⟨Create View dist_base statement (Eq. 2.9) 31d⟩);
      stmt.addBatch(⟨Create View dist_s_travel statement 31e⟩);
      stmt.addBatch(⟨Create View dist_s_cruising statement (Eq. 2.12) 31f⟩);
      stmt.addBatch(⟨Create View dist_s_service statement (Eq. 2.13) 31g⟩);
      stmt.addBatch(⟨Create View dist_s_base statement 31h⟩);
      stmt.addBatch(⟨Create View dist_r_base statement 31i⟩);
      stmt.addBatch(⟨Create View dist_r_unassigned statement 31j⟩);
      stmt.addBatch(⟨Create View dist_r_transit statement (Eq. 2.19) 32a⟩);
      stmt.addBatch(⟨Create View dist_r_detour statement (Eq. 2.21) 31k⟩);
      stmt.addBatch(⟨Create View dur_s_travel statement 32b⟩);
      stmt.addBatch(⟨Create View dur_s_service statement 32c⟩);
      stmt.addBatch(⟨Create View dur_r_pickup statement (Eq. 2.18) 32d⟩);
      stmt.addBatch(⟨Create View dur_r_transit statement (Eq. 2.20) 32e⟩);
      stmt.addBatch(⟨Create View dur_r_travel statement (Eq. 2.23) 32f⟩);
      stmt.addBatch(⟨Create View t_r_depart statement (Eq. 2.16) 32g⟩);
      stmt.addBatch(⟨Create View t_s_depart statement (Eq. 2.16) 32h⟩);
      stmt.addBatch(⟨Create View t_r_arrive statement (Eq. 2.17) 32i⟩);
      stmt.addBatch(⟨Create View t_s_arrive statement (Eq. 2.17) 32j⟩);
      stmt.addBatch(⟨Create View violations_t_s 32k⟩);
      stmt.addBatch(⟨Create View violations_t_r 32l⟩);
      stmt.addBatch("CREATE INDEX R_re ON R (re)");
      stmt.addBatch("CREATE INDEX W_sid_t1 ON W (sid, t1)");
      stmt.addBatch("CREATE INDEX W_sid_t2 ON W (sid, t2)");
      stmt.addBatch("CREATE INDEX W_sid_v2 ON W (sid, v2)");
      stmt.addBatch("CREATE INDEX W_sid_t1_t2 ON W (sid, t1, t2)");
      stmt.addBatch("CREATE INDEX CQ_sid_t2_o2 ON CQ (sid, t2, o2)");
      stmt.addBatch("CREATE INDEX CQ_sid_t2_q2 ON CQ (sid, t2 DESC, q2 DESC)");
      stmt.executeBatch();
      conn.commit();
    } catch (SQLException e) {
      System.err.println("Fatal error.");
      if (e.getErrorCode() == 0) {
        System.err.println("(did you forget to call Storage.JargoInstanceNew()?)");
      } else if (e.getErrorCode() == 20000) {
        System.err.println("(data model already exists from Storage.JargoInstanceLoad()?)");
      }
      e.printStackTrace(System.err);
      System.exit(1);
    }
  }
```

Defines:
  `JargoInstanceInitialize`, used in chunk 39a.
Uses `JargoInstanceLoad` 39b and `JargoInstanceNew` 37a.

39a ⟨*Admin: instanceInitialize(0)* 39a⟩≡ (141c)
```
  void instanceInitialize() {
    this.storage.JargoInstanceInitialize();
  }
```
Defines:
  `instanceInitialize`, used in chunks 167b and 192e.
Uses `JargoInstanceInitialize` 37c.

□

`JargoInstanceLoad`**(1)**

Loads an on-disk database into memory.

**Parameters:**
- `String p`: Path to database.

**Returns:** Nothing.
**Side Effects:** Modifies the in-memory database.
**Throws:**
- `SQLException`: if database failure is encountered
- `ClassNotFoundException`: if Derby driver cannot be loaded

**Wrappers:**
- `instanceLoad`(1)

39b ⟨*Admin: JargoInstanceLoad(1)* 39b⟩≡ (133b)
```
  void JargoInstanceLoad(final String p) throws SQLException {
    this.CONNECTIONS_URL = "jdbc:derby:memory:jargo;createFrom="+p;
    try {
      this.JargoSetupDriver();
    } catch (ClassNotFoundException e) {
      System.out.println("Fatal error.");
      e.printStackTrace();
      System.exit(1);
    }
  }
```
Defines:
  `JargoInstanceLoad`, used in chunks 37c and 39c.
Uses `JargoSetupDriver` 44b.

39c ⟨*Admin: instanceLoad(1)* 39c⟩≡ (141c)
```
  void instanceLoad(final String p) throws SQLException {
    this.storage.JargoInstanceLoad(p);
  }
```
Defines:
  `instanceLoad`, used in chunk 193.
Uses `JargoInstanceLoad` 39b.

## JargoInstanceExport(1)

Exports the in-memory database to disk.

**Parameters:**

- `String p`: path to the export.

**Returns:** Nothing.
**Side Effects:** Modifies the disk.
**Throws:**

- `SQLException`: if database failure is encountered

**Wrappers:**

- `instanceExport`(1)

40a ⟨*Admin: JargoInstanceExport(1)* 40a⟩≡ (133b)
```
void JargoInstanceExport(final String p) throws SQLException {
  try (⟨Open conn 35c⟩) {
    CallableStatement cs = conn.prepareCall("CALL SYSCS_UTIL.SYSCS_BACKUP_DATABASE('"+p+"')");
    cs.execute();
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
`JargoInstanceExport`, used in chunk 40b.

40b ⟨*Admin: instanceExport(1)* 40b⟩≡ (141c)
```
void instanceExport(final String p) throws SQLException {
  this.storage.JargoInstanceExport(p);
}
```
Defines:
`instanceExport`, used in chunks 137, 138, 165d, and 202.
Uses `JargoInstanceExport` 40a.

## JargoInstanceClose(0)

Closes the database instance.

**Parameters:** None.
**Returns:** Nothing.
**Side Effects:** Drops the database connection and clears many data structures (see source).
**Throws:**

- `SQLException`: code 45000 on success, some other error code otherwise

**Wrappers:**

- `instanceClose`(0)

40c ⟨*Admin: JargoInstanceClose(0)* 40c⟩≡ (133b)
```
void JargoInstanceClose() throws SQLException {
  try {
    DriverManager.getConnection("jdbc:derby:memory:jargo;drop=true");
  } catch (SQLException e) {
    if (e.getErrorCode() != 45000) {
      throw e;
    } else {
      this.lu_rstatus.clear();
      this.lu_vertices.clear();
      this.lu_edges.clear();
      this.lu_users.clear();
      this.lu_lvt.clear();
```

```
        this.count_requests = 0;
        this.count_assigned = 0;
        this.sum_distance_unassigned = 0;
        this.sum_distance_base_requests = 0;
        this.sum_distance_base_servers = 0;
        this.distance_servers.clear();
        this.distance_servers_cruising.clear();
        this.distance_requests_transit.clear();
        this.duration_servers.clear();
        this.duration_servers_cruising.clear();
        this.duration_requests_transit.clear();
        this.duration_requests_travel.clear();
        this.duration_requests_pickup.clear();
        this.connection_factory = null;
        this.poolableconnection_factory = null;
        this.pool = null;
        this.driver = null;
      }
    }
  }
```

Defines:
`JargoInstanceClose`, used in chunk 41a.

41a ⟨*Admin: instanceClose(0)* 41a⟩≡ (141c)

```
  void instanceClose() throws SQLException {
    this.storage.JargoInstanceClose();
  }
```

Defines:
`instanceClose`, used in chunk 203.
Uses `JargoInstanceClose` 40c.

## `JargoCacheRoadNetworkFromDB`(0)

Loads in-memory vertex and edge caches from the V and E database tables.

| | |
|---|---|
| **Parameters:** | None. |
| **Returns:** | Nothing. |
| **Side Effects:** | Modifies `lu_vertices` and `lu_edges`. |
| **Throws:** | • `SQLException`: if database failure is encountered |
| **Wrappers:** | • `cacheRoadNetworkFromDB`(0) |

41b ⟨*Admin: JargoCacheRoadNetworkFromDB(0)* 41b⟩≡ (133b) 41c ▷

```
  void JargoCacheRoadNetworkFromDB() throws SQLException {
```

Defines:
`JargoCacheRoadNetworkFromDB`, used in chunk 42d.

Our approach is to create two temporary maps on the heap, populate the temporary maps, then assign `lu_vertices` and `lu_edges` to reference the temporary maps if all succeeds. This way we don't corrupt `lu_vertices` and `lu_edges` in case of failure. (The approach might be overly cautious as it's hard to imagine why this method would ever be called if the caches are already populated.)

41c ⟨*Admin: JargoCacheRoadNetworkFromDB(0)* 41b⟩+≡ (133b) ◁ 41b 42a ▷

```
    ConcurrentHashMap<Integer, int[]>    lu1 =
        new ConcurrentHashMap<Integer, int[]>();
    ConcurrentHashMap<Integer,
      ConcurrentHashMap<Integer, int[]>> lu2 =
        new ConcurrentHashMap<Integer, ConcurrentHashMap<Integer, int[]>>();
```

We start by querying the vertices.

42a ⟨*Admin: JargoCacheRoadNetworkFromDB(0)* 41b⟩+≡ (133b) ◁41c 42b▷
```
    try {
      final int[] output = this.DBQueryVertices();
      for (int i = 0; i < (output.length - 2); i += 3) {
        final int   v = output[(i + 0)];
        final int lng = output[(i + 1)];
        final int lat = output[(i + 2)];
        lu1.put(v, new int[] { lng, lat });
      }
    } catch (SQLException e) {
      throw e;
    }
```
Uses DBQueryVertices 70b.

Then we go on to query the edges.

42b ⟨*Admin: JargoCacheRoadNetworkFromDB(0)* 41b⟩+≡ (133b) ◁42a 42c▷
```
    try {
      final int[] output = this.DBQueryEdges();
      for (int i = 0; i < (output.length - 3); i += 4) {
        final int v1 = output[(i + 0)];
        final int v2 = output[(i + 1)];
        final int dd = output[(i + 2)];
        final int nu = output[(i + 3)];
        if (!lu2.containsKey(v1)) {
          lu2.put(v1, new ConcurrentHashMap<Integer, int[]>());
        }
        lu2.get(v1).put(v2, new int[] { dd, nu });
      }
    } catch (SQLException e) {
      throw e;
    }
```
Uses DBQueryEdges 72a.

Finally we do the assignment.

42c ⟨*Admin: JargoCacheRoadNetworkFromDB(0)* 41b⟩+≡ (133b) ◁42b
```
    this.lu_vertices = lu1;
    this.lu_edges    = lu2;
  }
```

42d ⟨*Admin: cacheRoadNetworkFromDB(0)* 42d⟩≡ (141c)
```
  void cacheRoadNetworkFromDB() throws SQLException {
    this.storage.JargoCacheRoadNetworkFromDB();
  }
```
Defines:
  cacheRoadNetworkFromDB, used in chunk 193.
Uses JargoCacheRoadNetworkFromDB 41b.

## JargoCacheUsersFromDB(0)

Load in-memory user caches from the S and R database tables.

| | |
|---|---|
| **Parameters:** | None. |
| **Returns:** | Nothing. |
| **Side Effects:** | Modifies lu_users and lu_rstatus. |
| **Throws:** | • SQLException: if database failure is encountered |
| **Wrappers:** | • cacheUsersFromDB(0) |

42e ⟨*Admin: JargoCacheUsersFromDB(0)* 42e⟩≡ (133b) 43a▷
```
  void JargoCacheUsersFromDB() throws SQLException {
```

Defines:
`JargoCacheUsersFromDB`, used in chunk 44a.

Our approach follows the approach for `JargoCacheRoadNetworkFromDB`(0). We start by creating two temporary maps on the heap.

43a ⟨*Admin: JargoCacheUsersFromDB(0)* 42e⟩+≡ (133b) ◁42e 43b▷

```
ConcurrentHashMap<Integer, int[]> lu1 = new ConcurrentHashMap<Integer, int[]>();
Map<Integer, Boolean>             lu2 = new HashMap<Integer, Boolean>();
Map<Integer, Integer>             lu3 = new HashMap<Integer, Integer>();
```

Then we query the users.

43b ⟨*Admin: JargoCacheUsersFromDB(0)* 42e⟩+≡ (133b) ◁43a 43c▷

```
try {
  final int[] output = this.DBQueryUsers();
  for (int i = 0; i < (output.length - 6); i += 7) {
    final int uid = output[(i + 0)];
    final int  uq = output[(i + 1)];
    final int  ue = output[(i + 2)];
    final int  ul = output[(i + 3)];
    final int  uo = output[(i + 4)];
    final int  ud = output[(i + 5)];
    final int  ub = output[(i + 6)];
    lu1.put(uid, new int[] { uid, uq, ue, ul, uo, ud, ub });
```

Uses `DBQueryUsers` 74b.

If the user is a request, in other words the user load is positive, we query the request's assignment status. Else, we initialize the last-visitation time.

43c ⟨*Admin: JargoCacheUsersFromDB(0)* 42e⟩+≡ (133b) ◁43b 43d▷

```
    if (uq > 0) {
      lu2.put(uid, (this.DBQueryRequestIsAssigned(uid, false).length > 0 ? true : false));
    } else {
      lu3.put(uid, 0);
    }
  }
} catch (SQLException e) {
  throw e;
}
```

Uses `DBQueryRequestIsAssigned` 75a.

Finally we do the assignment and populate some counters.

43d ⟨*Admin: JargoCacheUsersFromDB(0)* 42e⟩+≡ (133b) ◁43c

```
this.lu_users   = lu1;
this.lu_rstatus = lu2;
this.lu_lvt     = lu3;
for (Integer uid : this.lu_users.keySet()) {
  final int[] u = this.lu_users.get(uid);
  if (u[1] > 0) {
    this.count_requests++;
    this.sum_distance_unassigned += u[6];
    this.sum_distance_base_requests += u[6];
  } else {
    final int sid = uid;
    ⟨Cache server distance 35e⟩
    ⟨Cache cruising distance and duration 36a⟩
    this.sum_distance_base_servers += u[6];
  }
}
for (Integer rid : this.lu_rstatus.keySet()) {
  final boolean flag = this.lu_rstatus.get(rid);
  if (flag == true) {
    this.count_assigned++;
    this.sum_distance_unassigned -= this.lu_users.get(rid)[6];
    int r = rid;
    try {
      int sid = this.DBQueryRequestIsAssigned(rid, false)[0];
```

```
          ⟨Cache request transit distance and duration 36b⟩
        } catch (SQLException e) {
          throw e;
        }
      }
    }
  }
```

Uses DBQueryRequestIsAssigned 75a.

44a ⟨*Admin: cacheUsersFromDB(0)* 44a⟩≡ (141c)

```
  void cacheUsersFromDB() throws SQLException {
    this.storage.JargoCacheUsersFromDB();
  }
```

Defines:
  cacheUsersFromDB, used in chunk 193.
Uses JargoCacheUsersFromDB 42e.

## JargoSetupDriver(0)

Sets up a new database connection.

| | |
|---|---|
| **Parameters:** | None. |
| **Returns:** | Nothing. |
| **Side Effects:** | Assigns new objects to connection_factory, poolableconnection_factory, pool, driver. |
| **Throws:** | • SQLException: if database failure is encountered<br>• ClassNotFoundException: if Derby driver cannot be loaded |
| **Wrappers:** | None. |

44b ⟨*Admin: JargoSetupDriver(0)* 44b⟩≡ (133b)

```
  void JargoSetupDriver() throws SQLException, ClassNotFoundException {
    connection_factory = new DriverManagerConnectionFactory(CONNECTIONS_URL);
    poolableconnection_factory = new PoolableConnectionFactory(connection_factory, null);
    poolableconnection_factory.setPoolStatements(true);
    poolableconnection_factory.setDefaultAutoCommit(false);
    poolableconnection_factory.setMaxOpenPreparedStatements(STATEMENTS_MAX_COUNT);
    poolableconnection_factory.setDefaultTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);
    GenericObjectPoolConfig<PoolableConnection> cfg = new GenericObjectPoolConfig<PoolableConnection>();
    cfg.setMinIdle(100000);
    cfg.setMaxIdle(100000);
    cfg.setMaxTotal(100000);
    pool = new GenericObjectPool<PoolableConnection>(poolableconnection_factory, cfg);
    poolableconnection_factory.setPool(pool);
    Class.forName("org.apache.commons.dbcp2.PoolingDriver");
    driver = (PoolingDriver) DriverManager.getDriver(CONNECTIONS_DRIVER_URL);
    driver.registerPool(CONNECTIONS_POOL_NAME, pool);
  }
```

Defines:
  JargoSetupDriver, used in chunks 37a and 39b.

## JargoSetupPreparedStatements(0)

Initialize the statement cache.

| | |
|---|---|
| **Parameters:** | None. |
| **Returns:** | Nothing. |
| **Side Effects:** | Modifies lu_pstr. |
| **Throws:** | • SQLException: if database failure is encountered |
| **Wrappers:** | None. |

45 ⟨*Admin: JargoSetupPreparedStatements(0)* 45⟩≡ (133b)

```
void JargoSetupPreparedStatements() {
  final String INS = "INSERT INTO ";
  final String UPD = "UPDATE ";
  final String DEL = "DELETE FROM ";
  final String SEL = "SELECT ";
  final String q2  = "(?,?)";
  final String q3  = "(?,?,?)";
  final String q4  = "(?,?,?,?)";
  final String q7  = "(?,?,?,?,?,?,?)";
  final String q8  = "(?,?,?,?,?,?,?,?)";
  final String q9  = "(?,?,?,?,?,?,?,?,?)";
  final String q12 = "(?,?,?,?,?,?,?,?,?,?,?,?)";
  final String q14 = "(?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
  ⟨S0 46a⟩
  ⟨S1 46b⟩
  ⟨S2 46c⟩
  ⟨S3 47a⟩
  ⟨S4 47b⟩
  ⟨S5 47c⟩
  ⟨S6 47d⟩
  ⟨S7 47e⟩
  ⟨S8 47f⟩
  ⟨S9 47g⟩
  ⟨S10 47h⟩
  ⟨S11 47j⟩
  ⟨S12 47k⟩
  ⟨S13 47l⟩
  ⟨S14 47m⟩
  ⟨S15 47n⟩
  ⟨S42 48g⟩
  ⟨S43 48h⟩
  ⟨S51 49e⟩
  ⟨S59 49g⟩
  ⟨S60 49i⟩
  ⟨S61 49k⟩
  ⟨S62 48j⟩
  ⟨S63 48l⟩
  ⟨S64 48k⟩
  ⟨S65 48m⟩
  ⟨S66 49c⟩
  ⟨S67 49f⟩
  ⟨S73 50d⟩
  ⟨S76 48f⟩
  ⟨S77 48b⟩
  ⟨S80 48i⟩
  ⟨S82 48d⟩
  ⟨S83 48e⟩
  ⟨S84 48c⟩
  ⟨S86 50c⟩
  ⟨S87 50e⟩
  ⟨S100 50f⟩
  ⟨S101 50g⟩
  ⟨S102 50h⟩
  ⟨S103 50i⟩
  ⟨S104 50j⟩
  ⟨S105 50k⟩
  ⟨S106 50l⟩
  ⟨S107 51a⟩
  ⟨S108 51b⟩
  ⟨S109 51c⟩
  ⟨S110 51d⟩
  ⟨S111 51e⟩
  ⟨S112 51f⟩
  ⟨S113 51g⟩
  ⟨S114 51h⟩
  ⟨S115 51i⟩
  ⟨S116 51j⟩
  ⟨S117 51k⟩
  ⟨S118 51l⟩
  ⟨S119 51m⟩
  ⟨S120 51n⟩
  ⟨S121 52a⟩
  ⟨S122 52b⟩
  ⟨S123 52c⟩
  ⟨S124 52d⟩
  ⟨S125 52e⟩
  ⟨S127 52g⟩
  ⟨S128 49h⟩
  ⟨S129 49j⟩
  ⟨S131 48a⟩
  ⟨S133 52h⟩
  ⟨S134 52i⟩
  ⟨S135 52j⟩
  ⟨S136 52k⟩
  ⟨S137 52l⟩
  ⟨S138 52m⟩
  ⟨S139 52n⟩
  ⟨S140 53a⟩
  ⟨S141 53b⟩
  ⟨S142 53c⟩
  ⟨S143 53d⟩
  ⟨S144 53e⟩
  ⟨S145 53f⟩
  ⟨S147 53g⟩
  ⟨S148 53h⟩
  ⟨S150 53j⟩
  ⟨S151 53k⟩
  ⟨S152 53l⟩
  ⟨S153 53m⟩
  ⟨S154 53n⟩
  ⟨S155 54a⟩
  ⟨S156 54b⟩
  ⟨S157 54c⟩
  ⟨S158 54d⟩
  ⟨S160 54f⟩
  ⟨S161 54g⟩
  ⟨S162 54h⟩
  ⟨S163 54i⟩
}
```

Defines:
`JargoSetupPreparedStatements`, used in chunk 131b.

46a ⟨*S0* 46a⟩≡ (45)

```
this.lu_pstr.put("S0", INS+"V VALUES "+q3);
```

Defines:
`S0`, used in chunk 120c.

46b ⟨*S1* 46b⟩≡ (45)

```
this.lu_pstr.put("S1", INS+"E VALUES "+q4);
```

Defines:
`S1`, used in chunk 121.

46c ⟨*S2* 46c⟩≡ (45)

```
this.lu_pstr.put("S2", INS+"UQ VALUES "+q2);
```

Defines:
`S2`, used in chunk 116a.

47a ⟨*S3* 47a⟩≡ (45)
```
  this.lu_pstr.put("S3", INS+"UE VALUES "+q2);
```
Defines:
  `S3`, used in chunk 116a.

47b ⟨*S4* 47b⟩≡ (45)
```
  this.lu_pstr.put("S4", INS+"UL VALUES "+q2);
```
Defines:
  `S4`, used in chunk 116a.

47c ⟨*S5* 47c⟩≡ (45)
```
  this.lu_pstr.put("S5", INS+"UO VALUES "+q2);
```
Defines:
  `S5`, used in chunk 116a.

47d ⟨*S6* 47d⟩≡ (45)
```
  this.lu_pstr.put("S6", INS+"UD VALUES "+q2);
```
Defines:
  `S6`, used in chunk 116a.

47e ⟨*S7* 47e⟩≡ (45)
```
  this.lu_pstr.put("S7", INS+"UB VALUES "+q2);
```
Defines:
  `S7`, used in chunk 116a.

47f ⟨*S8* 47f⟩≡ (45)
```
  this.lu_pstr.put("S8", INS+"S VALUES "+q7);
```
Defines:
  `S8`, used in chunk 116c.

47g ⟨*S9* 47g⟩≡ (45)
```
  this.lu_pstr.put("S9", INS+"R VALUES "+q7);
```
Defines:
  `S9`, used in chunk 116b.

47h ⟨*S10* 47h⟩≡ (45)
```
  this.lu_pstr.put("S10", INS+"W VALUES "+q8);
```
Defines:
  `S10`, used in chunks 116d and 118a.

47i ⟨*S70* 47i⟩≡
```
  this.lu_pstr.put("S70", SEL+"sid, sq, se, sl, so, sd, sb FROM S WHERE sid=?");
```
Defines:
  `S70`, never used.

47j ⟨*S11* 47j⟩≡ (45)
```
  this.lu_pstr.put("S11", INS+"CW VALUES "+q9);
```
Defines:
  `S11`, used in chunk 116f.

47k ⟨*S12* 47k⟩≡ (45)
```
  this.lu_pstr.put("S12", INS+"PD VALUES "+q4);
```
Defines:
  `S12`, used in chunk 117c.

47l ⟨*S13* 47l⟩≡ (45)
```
  this.lu_pstr.put("S13", INS+"CPD VALUES "+q12);
```
Defines:
  `S13`, used in chunk 117c.

47m ⟨*S14* 47m⟩≡ (45)
```
  this.lu_pstr.put("S14", INS+"CQ VALUES "+q14);
```
Defines:
  `S14`, used in chunk 117.

47n ⟨*S15* 47n⟩≡ (45)
```
  this.lu_pstr.put("S15", UPD+"E SET nu=? WHERE v1=? AND v2=?");
```
Defines:
  `S15`, used in chunk 122a.

48a ⟨*S131* 48a⟩≡ (45)
```
  this.lu_pstr.put("S131", UPD+"W SET nu=? WHERE v1=? AND v2=?");
```
Defines:
  `S131`, used in chunk 122a.

48b ⟨*S77* 48b⟩≡ (45)
```
  this.lu_pstr.put("S77", UPD+"CW SET te=?, ve=? WHERE sid=?");
```
Defines:
  `S77`, used in chunk 119b.

48c ⟨*S84* 48c⟩≡ (45)
```
  this.lu_pstr.put("S84", UPD+"PD SET t2=? WHERE v2=? AND rid=?");
```
Defines:
  `S84`, used in chunk 119c.

48d ⟨*S82* 48d⟩≡ (45)
```
  this.lu_pstr.put("S82", UPD+"CPD SET tp=? WHERE vp=? AND rid=?");
```
Defines:
  `S82`, used in chunk 119c.

48e ⟨*S83* 48e⟩≡ (45)
```
  this.lu_pstr.put("S83", UPD+"CPD SET td=? WHERE vd=? AND rid=?");
```
Defines:
  `S83`, used in chunk 119c.

48f ⟨*S76* 48f⟩≡ (45)
```
  this.lu_pstr.put("S76", DEL+"W WHERE sid=? AND t2>?");
```
Defines:
  `S76`, used in chunk 115b.

48g ⟨*S42* 48g⟩≡ (45)
```
  this.lu_pstr.put("S42", DEL+"PD WHERE rid=?");
```
Defines:
  `S42`, used in chunk 115c.

48h ⟨*S43* 48h⟩≡ (45)
```
  this.lu_pstr.put("S43", DEL+"CPD WHERE rid=?");
```
Defines:
  `S43`, used in chunk 115c.

48i ⟨*S80* 48i⟩≡ (45)
```
  this.lu_pstr.put("S80", DEL+"CQ WHERE sid=? AND t2>?");
```
Defines:
  `S80`, used in chunk 115d.

48j ⟨*S62* 48j⟩≡ (45)
```
  this.lu_pstr.put("S62", SEL+"COUNT (*) FROM V WHERE v<>0");
```
Defines:
  `S62`, used in chunk 70d.

48k ⟨*S64* 48k⟩≡ (45)
```
  this.lu_pstr.put("S64", SEL+"MIN (lng), MAX (lng), MIN (lat), MAX (lat) "
        + "FROM V WHERE v<>0");
```
Defines:
  `S64`, used in chunk 69a.

48l ⟨*S63* 48l⟩≡ (45)
```
  this.lu_pstr.put("S63", SEL+"COUNT (*) FROM E WHERE v1<>0 AND v2<>0");
```
Defines:
  `S63`, used in chunk 72c.

48m ⟨*S65* 48m⟩≡ (45)
```
  this.lu_pstr.put("S65", SEL+"MIN (dd), MAX (dd), SUM (dd) / COUNT (dd), "
        + "MIN (nu), MAX (nu), SUM (nu) / COUNT (nu) "
        + "FROM E WHERE v1<>0 AND v2<>0");
```
Defines:
  `S65`, used in chunk 73a.

48n ⟨*S46* 48n⟩≡
```
  this.lu_pstr.put("S46", SEL+"dd, nu FROM E WHERE v1=? AND v2=?");
```
Defines:
  `S46`, never used.

49a ⟨*S130* 49a⟩≡
```
  this.lu_pstr.put("S130", SEL+"lng, lat FROM V WHERE v=?");
```
Defines:
S130, never used.

49b ⟨*S48* 49b⟩≡
```
  this.lu_pstr.put("S48", SEL+"sq, se FROM S WHERE sid=?");
```
Defines:
S48, never used.

49c ⟨*S66* 49c⟩≡ (45)
```
  this.lu_pstr.put("S66", SEL+"COUNT (*) FROM S");
```
Defines:
S66, used in chunk 94a.

49d ⟨*S75* 49d⟩≡
```
  this.lu_pstr.put("S75", SEL+"rid, rq, re, rl, ro, rd, rb FROM R WHERE rid=?");
```
Defines:
S75, never used.

49e ⟨*S51* 49e⟩≡ (45)
```
  this.lu_pstr.put("S51", SEL+"rq, re, rl, ro, rd FROM R WHERE rid=?");
```
Defines:
S51, used in chunk 117c.

49f ⟨*S67* 49f⟩≡ (45)
```
  this.lu_pstr.put("S67", SEL+"COUNT (*) FROM R");
```
Defines:
S67, used in chunk 79b.

49g ⟨*S59* 49g⟩≡ (45)
```
  this.lu_pstr.put("S59", SEL+"a.sid, a.t2, a.v2 FROM W AS a INNER JOIN ("
        + "SELECT sid, MIN(ABS(t2-?)) as tdiff FROM W WHERE t2<=? AND v2<>0 "
        + "GROUP BY sid"
        + ") as b ON a.sid=b.sid AND ABS(a.t2-?)=b.tdiff AND a.t2<=?");
```
Defines:
S59, used in chunk 95c.

49h ⟨*S128* 49h⟩≡ (45)
```
  this.lu_pstr.put("S128", SEL+"a.sid, a.t2, a.v2 FROM W AS a INNER JOIN ("
        + "SELECT sid FROM CW WHERE te>? OR (ve=0 AND sl>?)"
        + ") as b ON a.sid=b.sid INNER JOIN ("
        + "SELECT sid, MIN(ABS(t2-?)) as tdiff FROM W WHERE t2<=? AND v2<>0 "
        + "GROUP BY sid"
        + ") as c ON a.sid=c.sid AND ABS(a.t2-?)=c.tdiff AND a.t2<=?");
```
Defines:
S128, never used.

49i ⟨*S60* 49i⟩≡ (45)
```
  this.lu_pstr.put("S60", SEL+"DISTINCT t, v FROM r_server WHERE sid=? ORDER BY t ASC");
```
Defines:
S60, used in chunk 82c.

49j ⟨*S129* 49j⟩≡ (45)
```
  this.lu_pstr.put("S129", SEL+"t, v FROM r_server WHERE sid=? AND t>? ORDER BY t ASC");
```
Defines:
S129, used in chunk 83a.

49k ⟨*S61* 49k⟩≡ (45)
```
  this.lu_pstr.put("S61", SEL+"t, v, Ls, Lr FROM r_server "
        + "LEFT JOIN CQ ON t=t2 and lr=rid WHERE r_server.sid=?"
        + "AND (Ls IS NOT NULL OR Lr IS NOT NULL) ORDER BY t, o2 ASC");
```
Defines:
S61, used in chunk 84a.

Need to join CQ in order to sort by order number o2. This query has worse performance after W (r_server), CQ, PD grow.

50a ⟨*S69 50a*⟩≡
```
  this.lu_pstr.put("S69", SEL+"t, v, Ls, Lr "
        + "FROM r_server LEFT JOIN CQ ON t=t2 and v=v2 and Lr=rid "
        + "WHERE r_server.sid=?"
        + "   AND (t>? OR v=0)"
        + "   AND (Ls IS NOT NULL OR Lr IS NOT NULL)"
        + "ORDER BY t ASC, o2 ASC");
```
Defines:
  S69, never used.

50b ⟨*S85 50b*⟩≡
```
  this.lu_pstr.put("S85", SEL+"uq FROM UQ WHERE uid=?");
```
Defines:
  S85, never used.

50c ⟨*S86 50c*⟩≡ (45)
```
  this.lu_pstr.put("S86", SEL+"tp, td FROM CPD WHERE rid=?");
```
Defines:
  S86, used in chunk 118d.

50d ⟨*S73 50d*⟩≡ (45)
```
  this.lu_pstr.put("S73", SEL+"q2 FROM CQ WHERE sid=? AND t2<=? "
        + "ORDER BY t2 DESC, q2 DESC FETCH FIRST ROW ONLY");
```
Defines:
  S73, used in chunk 85b.

50e ⟨*S87 50e*⟩≡ (45)
```
  this.lu_pstr.put("S87", SEL+"t2, q2, o2 FROM CQ WHERE sid=? AND t2<=? "
        + "ORDER BY t2 DESC, o2 DESC FETCH FIRST ROW ONLY");
```
Defines:
  S87, used in chunk 119a.

50f ⟨*S100 50f*⟩≡ (45)
```
  this.lu_pstr.put("S100", SEL+"rid FROM assignments WHERE t>? AND sid=?");
```
Defines:
  S100, used in chunk 92c.

50g ⟨*S101 50g*⟩≡ (45)
```
  this.lu_pstr.put("S101", SEL+"rid FROM assignments WHERE t<=? AND sid=?");
```
Defines:
  S101, used in chunk 93a.

50h ⟨*S102 50h*⟩≡ (45)
```
  this.lu_pstr.put("S102", SEL+"* FROM service_rate");
```
Defines:
  S102, used in chunk 97b.

50i ⟨*S103 50i*⟩≡ (45)
```
  this.lu_pstr.put("S103", SEL+"* FROM dist_base");
```
Defines:
  S103, used in chunk 98e.

50j ⟨*S104 50j*⟩≡ (45)
```
  this.lu_pstr.put("S104", SEL+"val FROM dist_s_travel WHERE sid=?");
```
Defines:
  S104, used in chunks 35e and 87a.

50k ⟨*S105 50k*⟩≡ (45)
```
  this.lu_pstr.put("S105", SEL+"SUM (val) FROM dist_s_travel");
```
Defines:
  S105, used in chunk 100a.

50l ⟨*S106 50l*⟩≡ (45)
```
  this.lu_pstr.put("S106", SEL+"val FROM dist_s_cruising WHERE sid=?");
```
Defines:
  S106, used in chunk 88b.

51a ⟨*S107* 51a⟩≡ (45)
```
  this.lu_pstr.put("S107", SEL+"SUM (val) FROM dist_s_cruising");
```
Defines:
  S107, used in chunk 102a.

51b ⟨*S108* 51b⟩≡ (45)
```
  this.lu_pstr.put("S108", SEL+"val FROM dist_s_service WHERE sid=?");
```
Defines:
  S108, used in chunk 88c.

51c ⟨*S109* 51c⟩≡ (45)
```
  this.lu_pstr.put("S109", SEL+"SUM (val) FROM dist_s_service");
```
Defines:
  S109, used in chunk 102e.

51d ⟨*S110* 51d⟩≡ (45)
```
  this.lu_pstr.put("S110", SEL+"val FROM dist_s_base");
```
Defines:
  S110, used in chunk 101b.

51e ⟨*S111* 51e⟩≡ (45)
```
  this.lu_pstr.put("S111", SEL+"val FROM dist_r_base");
```
Defines:
  S111, used in chunk 106c.

51f ⟨*S112* 51f⟩≡ (45)
```
  this.lu_pstr.put("S112", SEL+"val FROM dist_r_detour WHERE rid=?");
```
Defines:
  S112, used in chunk 75b.

51g ⟨*S113* 51g⟩≡ (45)
```
  this.lu_pstr.put("S113", SEL+"SUM (val) FROM dist_r_detour");
```
Defines:
  S113, used in chunk 108b.

51h ⟨*S114* 51h⟩≡ (45)
```
  this.lu_pstr.put("S114", SEL+"val FROM dist_r_transit WHERE rid=?");
```
Defines:
  S114, used in chunk 76a.

51i ⟨*S115* 51i⟩≡ (45)
```
  this.lu_pstr.put("S115", SEL+"SUM (val) FROM dist_r_transit");
```
Defines:
  S115, used in chunk 109b.

51j ⟨*S116* 51j⟩≡ (45)
```
  this.lu_pstr.put("S116", SEL+"val FROM dur_s_travel WHERE sid=?");
```
Defines:
  S116, used in chunk 90a.

51k ⟨*S117* 51k⟩≡ (45)
```
  this.lu_pstr.put("S117", SEL+"SUM (val) FROM dur_s_travel");
```
Defines:
  S117, used in chunk 103d.

51l ⟨*S118* 51l⟩≡ (45)
```
  this.lu_pstr.put("S118", SEL+"val FROM dur_r_pickup WHERE rid=?");
```
Defines:
  S118, used in chunk 76b.

51m ⟨*S119* 51m⟩≡ (45)
```
  this.lu_pstr.put("S119", SEL+"SUM (val) FROM dur_r_pickup");
```
Defines:
  S119, used in chunk 110b.

51n ⟨*S120* 51n⟩≡ (45)
```
  this.lu_pstr.put("S120", SEL+"val FROM dur_r_transit WHERE rid=?");
```
Defines:
  S120, used in chunk 77a.

52a ⟨*S121* 52a⟩≡ (45)
```
this.lu_pstr.put("S121", SEL+"SUM (val) FROM dur_r_transit");
```
Defines:
`S121`, used in chunk 111a.

52b ⟨*S122* 52b⟩≡ (45)
```
this.lu_pstr.put("S122", SEL+"val FROM dur_r_travel WHERE rid=?");
```
Defines:
`S122`, used in chunk 77b.

52c ⟨*S123* 52c⟩≡ (45)
```
this.lu_pstr.put("S123", SEL+"SUM (val) FROM dur_r_travel");
```
Defines:
`S123`, used in chunk 112a.

52d ⟨*S124* 52d⟩≡ (45)
```
this.lu_pstr.put("S124", SEL+"val FROM t_r_depart WHERE rid=?");
```
Defines:
`S124`, used in chunk 78a.

52e ⟨*S125* 52e⟩≡ (45)
```
this.lu_pstr.put("S125", SEL+"val FROM t_s_depart WHERE sid=?");
```
Defines:
`S125`, used in chunk 91a.

52f ⟨*S126* 52f⟩≡
```
this.lu_pstr.put("S126", SEL+"val FROM t_r_arrive WHERE rid=?");
```
Defines:
`S126`, used in chunk 78c.

52g ⟨*S127* 52g⟩≡ (45)
```
this.lu_pstr.put("S127", SEL+"te FROM CW WHERE sid=?");
```
Defines:
`S127`, used in chunks 89a and 92a.

52h ⟨*S133* 52h⟩≡ (45)
```
this.lu_pstr.put("S133", SEL+"val FROM f_status WHERE rid=? AND t<=? "
    + "ORDER BY t DESC FETCH FIRST ROW ONLY");
```
Defines:
`S133`, used in chunk 74c.

52i ⟨*S134* 52i⟩≡ (45)
```
this.lu_pstr.put("S134", SEL+"sid, te FROM CW WHERE se<=? AND (?<te OR (ve=0 AND sl>?))");
```
Defines:
`S134`, used in chunks 93b, 94c, and 96b.

52j ⟨*S135* 52j⟩≡ (45)
```
this.lu_pstr.put("S135", SEL+"t2, v2 FROM W WHERE sid=? AND t2=("
    + "SELECT t1 FROM W WHERE sid=? AND ? <= t1 AND t1 <= ? AND ? < t2)");
```
Defines:
`S135`, used in chunk 96b.

52k ⟨*S136* 52k⟩≡ (45)
```
this.lu_pstr.put("S136", SEL+"* FROM V");
```
Defines:
`S136`, used in chunk 70b.

52l ⟨*S137* 52l⟩≡ (45)
```
this.lu_pstr.put("S137", SEL+"* FROM E");
```
Defines:
`S137`, used in chunk 72a.

52m ⟨*S138* 52m⟩≡ (45)
```
this.lu_pstr.put("S138", SEL+"val FROM dist_r_unassigned");
```
Defines:
`S138`, used in chunk 107a.

52n ⟨*S139* 52n⟩≡ (45)
```
this.lu_pstr.put("S139", UPD+"CPD SET te=? WHERE sid=?");
```
Defines:
`S139`, used in chunk 119b.

53a ⟨*S140* 53a⟩≡ (45)
```
  this.lu_pstr.put("S140", UPD+"CQ SET tp=?, td=? WHERE rid=?");
```
Defines:
  S140, used in chunk 118d.

53b ⟨*S141* 53b⟩≡ (45)
```
  this.lu_pstr.put("S141", SEL+"* FROM r_user");
```
Defines:
  S141, used in chunk 74b.

53c ⟨*S142* 53c⟩≡ (45)
```
  this.lu_pstr.put("S142", SEL+"SUM (dd) FROM W WHERE sid=? AND t2>?");
```
Defines:
  S142, used in chunk 87c.

53d ⟨*S143* 53d⟩≡ (45)
```
  this.lu_pstr.put("S143", SEL+"* FROM R WHERE re<=? AND ?<=re+?");
```
Defines:
  S143, used in chunk 81c.

53e ⟨*S144* 53e⟩≡ (45)
```
  this.lu_pstr.put("S144", SEL+"t2, v2, rid FROM CQ WHERE sid=? AND t2>? ORDER BY o2 ASC");
```
Defines:
  S144, used in chunk 84c.

53f ⟨*S145* 53f⟩≡ (45)
```
  this.lu_pstr.put("S145", SEL+"te, ve FROM CW WHERE sid=?");
```
Defines:
  S145, used in chunks 36a and 84c.

53g ⟨*S147* 53g⟩≡ (45)
```
  this.lu_pstr.put("S147", SEL+"t2, v2 FROM W WHERE sid=? AND t2=("
      + "SELECT t1 FROM W WHERE sid=? AND v2=0)");
```
Defines:
  S147, used in chunk 96b.

53h ⟨*S148* 53h⟩≡ (45)
```
  this.lu_pstr.put("S148", SEL+"sid FROM assignments WHERE rid=?");
```
Defines:
  S148, used in chunk 75a.

53i ⟨*S149* 53i⟩≡
```
  this.lu_pstr.put("S149", SEL+"t2, v2 FROM W WHERE sid=? ORDER BY t2 ASC");
```
Defines:
  S149, never used.

53j ⟨*S150* 53j⟩≡ (45)
```
  this.lu_pstr.put("S150", SEL+"COUNT (*) FROM violations_t_s");
```
Defines:
  S150, used in chunk 106a.

53k ⟨*S151* 53k⟩≡ (45)
```
  this.lu_pstr.put("S151", SEL+"COUNT (*) FROM violations_t_r");
```
Defines:
  S151, used in chunk 112e.

53l ⟨*S152* 53l⟩≡ (45)
```
  this.lu_pstr.put("S152", SEL+"t2, v2 FROM W WHERE sid=? AND t2>=? "
      + "ORDER BY t2 ASC FETCH FIRST ? ROWS ONLY");
```
Defines:
  S152, used in chunk 83c.

53m ⟨*S153* 53m⟩≡ (45)
```
  this.lu_pstr.put("S153", SEL+"t1, t2 FROM CQ WHERE sid=? AND q1=sq");
```
Defines:
  S153, used in chunk 36a.

53n ⟨*S154* 53n⟩≡ (45)
```
  this.lu_pstr.put("S154", SEL+"SUM (dd) FROM W WHERE sid=? AND t2 > ? AND t2 <= ?");
```
Defines:
  S154, used in chunk 36a.

54a ⟨*S155* 54a⟩≡ (45)
```
  this.lu_pstr.put("S155", SEL+"MAX (td) FROM CPD WHERE sid = ?");
```
Defines:
  S155, used in chunk 36a.

54b ⟨*S156* 54b⟩≡ (45)
```
  this.lu_pstr.put("S156", SEL+"ts, vs FROM CW WHERE sid=?");
```
Defines:
  S156, used in chunk 36a.

54c ⟨*S157* 54c⟩≡ (45)
```
  this.lu_pstr.put("S157", SEL+"val FROM dur_s_service WHERE sid=?");
```
Defines:
  S157, used in chunk 90d.

54d ⟨*S158* 54d⟩≡ (45)
```
  this.lu_pstr.put("S158", SEL+"SUM (val) FROM dur_s_service");
```
Defines:
  S158, used in chunks 90c and 105c.

54e ⟨*S159* 54e⟩≡
```
  this.lu_pstr.put("S159", SEL+"(a.val - b.val) FROM "
      + "(SELECT val FROM dur_s_travel WHERE sid=?) as a,"
      + "(SELECT val FROM dur_s_service WHERE sid=?) as b");
```
Defines:
  S159, never used.

54f ⟨*S160* 54f⟩≡ (45)
```
  this.lu_pstr.put("S160", SEL+"(a.val - b.val) FROM "
      + "(SELECT SUM (val) as val FROM dur_s_travel) as a,"
      + "(SELECT SUM (val) as val FROM dur_s_service) as b");
```
Defines:
  S160, used in chunk 104d.

54g ⟨*S161* 54g⟩≡ (45)
```
  this.lu_pstr.put("S161", SEL+"(a.val - b.val) FROM "
      + "(SELECT COUNT (*) as val FROM R WHERE ?>= re AND ? < rl) as a,"
      + "(SELECT COUNT (*) as val FROM assignments_r WHERE t <= ?) as b");
```
Defines:
  S161, used in chunks 79d and 80f.

54h ⟨*S162* 54h⟩≡ (45)
```
  this.lu_pstr.put("S162", SEL+"r.rid, r.ro FROM R LEFT JOIN CPD ON R.rid = CPD.rid "
      + "WHERE R.re <= ? AND ? < (R.re + ?) AND (? < CPD.tp OR CPD.tp IS NULL)");
```
Defines:
  S162, used in chunk 82a.

54i ⟨*S163* 54i⟩≡ (45)
```
  this.lu_pstr.put("S163", SEL+"COUNT (*) FROM CQ WHERE sid=? AND q1+? > 0"
      + "AND ( (t1 < ? AND t2 > ?) OR (? < t2 AND t2 <= ?) )");
```
Defines:
  S163, used in chunk 86b.

## PSCreate(2)

Create a prepared statement for the given statement string.

**Parameters:**

- `Connection conn`: the database connection for creating the prepared statement
- `String k`: the statement identifier (see `JargoSetupPreparedStatements`(0))

**Returns:** A `PreparedStatement`.

**Side Effects:** None.

**Throws:**

- `SQLException`: if database failure is encountered

**Wrappers:** None.

55a ⟨*Admin: PSCreate(2)* 55a⟩≡ (133b)
```
PreparedStatement PSCreate(final Connection conn, final String k) throws SQLException {
  PreparedStatement p = null;
  try {
    p = conn.prepareStatement(lu_pstr.get(k),
      ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
    p.clearBatch();
    p.clearParameters();
  } catch (SQLException e) {
    throw e;
  }
  return p;
}
```
Defines:
`PSCreate`, used in chunks 56a and 115–22.

### `PSAdd`(2..)

Add a prepared statement to the batch of statements to be executed.

**Parameters:**
- `PreparedStatement p`: the prepared statement to add
- `Integer.. values`: the values to use in the statement, if any

**Returns:** Nothing.
**Side Effects:** Modifies p.
**Throws:**
- `SQLException`: if database failure is encountered

**Wrappers:** None.

55b ⟨*Admin: PSAdd(2..)* 55b⟩≡ (133b)
```
void PSAdd(PreparedStatement p, final Integer... values) throws SQLException {
  p.clearParameters();
  ⟨Set statement values 35d⟩
  try {
    p.addBatch();
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
`PSAdd`, used in chunks 115–22.

### `PSSubmit`(1..)

Executes a batch of prepared statements in the order that they are given.

**Parameters:**
- `PreparedStatement.. statements`: prepared statements to execute

**Returns:** Nothing.
**Side Effects:** May modified the database.
**Throws:**
- `SQLException`: if database failure is encountered

**Wrappers:** None.

55c ⟨*Admin: PSSubmit(1..)* 55c⟩≡ (133b)
```
void PSSubmit(PreparedStatement... statements) throws SQLException {
    try {
      for (PreparedStatement p : statements) {
        p.executeBatch();
        p.close();
      }
    } catch (SQLException e) {
      throw e;
    }
  }
```

Defines:
  PSSubmit, used in chunks 115–22.

### PSQuery(3..)

Execute a predefined SELECT statement and return the results in a flattened array.

**Parameters:**

- Connection conn: the database connection
- String k: the statement identifier (see JargoSetupPreparedStatements)
- Integer ncols: the number of columns in the selection

**Returns:** results of the query flattened into an integer array, or null if no results. Element $in + j$ contains the value of column $j$ at row $i$ of the result set, for $n$ columns.

**Side Effects:** None.

**Throws:**

- SQLException: if database failure is encountered

**Wrappers:** None.

56a ⟨*Admin: PSQuery(3..)* 56a⟩≡ (133b)

```
  int[] PSQuery(final Connection conn, final String k, final int ncols, final Integer... values)
  throws SQLException {
    int[] output = new int[] { };
    try {
      PreparedStatement p = PSCreate(conn, k);
      ⟨Set statement values 35d⟩
      ResultSet res = p.executeQuery();
      if (res.last() == true) {
        ⟨Flatten results 67a⟩
      }
      res.close();
      p.close();
    } catch (SQLException e) {
      throw e;
    }
    return output;
  }
```

Defines:
  PSQuery, used in chunks 35e, 36a, 69, 70, 72–98, 100–112, and 117–19.
Uses PSCreate 55a.

## 3.2.4 Methods: Getters

### getClock(0)

56b ⟨*Admin: getClock(0)* 56b⟩≡ (141c)

```
  int getClock() {
    return this.simClock;
  }
```

Defines:
  getClock, used in chunks 58f, 178, and 181–83.

### getClockStart(0)

57a ⟨*Admin: getClockStart(0)* 57a⟩≡ (141c)
```
int getClockStart() {
  return this.CLOCK_START;
}
```
Defines:
`getClockStart`, never used.

### getClockReference(0)

57b ⟨*Admin: getClockReference(0)* 57b⟩≡ (141c)
```
String getClockReference() {
  return this.refTimeStr;
}
```
Defines:
`getClockReference`, never used.

### getClockReferenceMs(0)

57c ⟨*Admin: getClockReferenceMs(0)* 57c⟩≡ (141c)
```
long getClockReferenceMs() {
  return this.refTimeMs;
}
```
Defines:
`getClockReferenceMs`, used in chunks 114, 178, and 183.

### getRefCacheEdges(0)

Method `getRefCacheEdges`(0) returns a read-only reference to `lu_edges`.

**Parameters:** none.
**Returns:** a read-only reference to `lu_edges`.
**Side Effects:** none.
**Throws:** nothing.

57d ⟨*Admin: getRefCacheEdges(0)* 57d⟩≡ (133b)
```
final ConcurrentHashMap<Integer, ConcurrentHashMap<Integer, int[]>> getRefCacheEdges() {
  return this.lu_edges;
}
```
Defines:
`getRefCacheEdges`, used in chunks 59b, 62a, and 144c.

### getRefCacheUsers(0)

Method `getRefCacheUsers`(0) returns a read-only reference to `lu_users`.

**Parameters:** none.
**Returns:** a read-only reference to `lu_users`.
**Side Effects:** none.
**Throws:** nothing.

57e ⟨*Admin: getRefCacheUsers(0)* 57e⟩≡ (133b)
```
final ConcurrentHashMap<Integer, int[]> getRefCacheUsers() {
  return this.lu_users;
}
```
Defines:
`getRefCacheUsers`, used in chunk 59c.

### getRefCacheVertices(0)

Method getRefCacheVertices(0) returns a read-only reference to lu_vertices.

**Parameters:** none.
**Returns:** a read-only reference to lu_vertices.
**Side Effects:** none.
**Throws:** nothing.

58a ⟨*Admin: getRefCacheVertices(0)* 58a⟩≡ (133b)
```
final ConcurrentHashMap<Integer, int[]> getRefCacheVertices() {
  return this.lu_vertices;
}
```
Defines:
`getRefCacheVertices`, used in chunks 59a, 62a, and 144c.

### getRefCommunicator(0)

58b ⟨*Admin: getRefCommunicator(0)* 58b⟩≡ (141c)
```
Communicator getRefCommunicator() {
  return this.communicator;
}
```
Defines:
`getRefCommunicator`, used in chunks 167b and 190g.

### getRefStorage(0)

58c ⟨*Admin: getRefStorage(0)* 58c⟩≡ (141c)
```
Storage getRefStorage() {
  return this.storage;
}
```
Defines:
`getRefStorage`, never used.

### retrieveQueueSize(0)

58d ⟨*Admin: retrieveQueueSize(0)* 58d⟩≡ (141c)
```
int retrieveQueueSize() {
  return this.client.getQueueSize();
}
```
Defines:
`retrieveQueueSize`, used in chunk 189g.
Uses `getQueueSize` 152f.

### retrieveHandleRequestDur(0)

58e ⟨*Admin: retrieveHandleRequestDur(0)* 58e⟩≡ (141c)
```
long retrieveHandleRequestDur() {
  return this.client.getHandleRequestDur();
}
```
Defines:
`retrieveHandleRequestDur`, used in chunk 190e.

### retrieveClock(0)

58f ⟨*Admin: retrieveClock(0)* 58f⟩≡ (149c)
```
int retrieveClock() {
  return this.controller.getClock();
}
```
Defines:
`retrieveClock`, used in chunk 125b.
Uses `getClock` 56b.

#### retrieveRefCacheVertices(0)

59a ⟨*Admin: retrieveRefCacheVertices(0) 59a*⟩≡ (141c 149c)
```
final ConcurrentHashMap<Integer, int[]> retrieveRefCacheVertices() {
  return this.storage.getRefCacheVertices();
}
```
Defines:
  retrieveRefCacheVertices, used in chunks 167b, 190g, and 191a.
Uses getRefCacheVertices 58a.

#### retrieveRefCacheEdges(0)

59b ⟨*Admin: retrieveRefCacheEdges(0) 59b*⟩≡ (141c 149c)
```
final ConcurrentHashMap<Integer, ConcurrentHashMap<Integer, int[]>> retrieveRefCacheEdges() {
  return this.storage.getRefCacheEdges();
}
```
Defines:
  retrieveRefCacheEdges, used in chunks 167b, 190g, and 191a.
Uses getRefCacheEdges 57d.

#### retrieveRefCacheUsers(0)

59c ⟨*Admin: retrieveRefCacheUsers(0) 59c*⟩≡ (141c 149c)
```
final ConcurrentHashMap<Integer, int[]> retrieveRefCacheUsers() {
  return this.storage.getRefCacheUsers();
}
```
Defines:
  retrieveRefCacheUsers, used in chunks 167b and 190g.
Uses getRefCacheUsers 57e.

### 3.2.5 Methods: Setters

#### setClockReference(1)

59d ⟨*Admin: setClockReference(1) 59d*⟩≡ (141c)
```
void setClockReference(final String clock_reference) throws IllegalArgumentException {
  int hour = Integer.parseInt(clock_reference.substring(0, 2));
  if (!(0 <= hour && hour <= 23)) {
    throw new IllegalArgumentException("Invalid clock reference (hour got "+hour+"; must be between [00, 23])
  }
  int minute = Integer.parseInt(clock_reference.substring(2, 4));
  if (!(0 <= minute && minute <= 59)) {
    throw new IllegalArgumentException("Invalid clock reference (minute got "+minute+"; must be between [00,
  }
  this.refTimeStr = clock_reference;
  try {
    this.refTimeMs = this.tools.parseClockReference(clock_reference);
  } catch (Exception pe) {
    throw new IllegalArgumentException("Invalid clock reference (parse failed)");
  }
  this.simClockReferenceHour= hour;
  this.simClockReferenceMinute = minute;
  if (DEBUG) {
    System.out.printf("refHr=%d, refMn=%d, refMs=%d\n",
      hour, minute, this.refTimeMs);
  }
}
```
Defines:
  setClockReference, used in chunk 140b.

## setClockStart(1)

60a ⟨*Admin: setClockStart(1)* 60a⟩≡ (141c)
```
  void setClockStart(final int clock_start) {
    this.CLOCK_START = clock_start;
    this.simClockReferenceSecond += clock_start;
    this.simClockReferenceSecond %= 60;
    this.simClockReferenceMinute += (clock_start / 60);
    this.simClockReferenceMinute %= 60;
    this.simClockReferenceHour += (clock_start / 3600);
    this.simClockReferenceHour %= 24;
    this.simClockReferenceDay += (clock_start / 86400);
    if (DEBUG) {
      System.out.printf("clock_start=%d\n", clock_start);
      System.out.printf("clock day %d %02d:%02d:%02d\n",
          this.simClockReferenceDay,
          this.simClockReferenceHour,
          this.simClockReferenceMinute,
          this.simClockReferenceSecond);
    }
  }
```
Defines:
  `setClockStart`, used in chunks 167b and 202.

## setClockEnd(1)

60b ⟨*Admin: setClockEnd(1)* 60b⟩≡ (141c)
```
  void setClockEnd(final int clock_end) {
    this.CLOCK_END = clock_end;
  }
```
Defines:
  `setClockEnd`, used in chunks 167b and 202.

## setQueueTimeout(1)

60c ⟨*Admin: setQueueTimeout(1)* 60c⟩≡ (141c)
```
  void setQueueTimeout(final int queue_timeout) {
    this.QUEUE_TIMEOUT = queue_timeout;
  }
```
Defines:
  `setQueueTimeout`, never used.

## setRequestTimeout(1)

60d ⟨*Admin: setRequestTimeout(1)* 60d⟩≡ (133b)
```
  void setRequestTimeout(final int request_timeout) {
    this.REQUEST_TIMEOUT = request_timeout;
  }
```
Defines:
  `setRequestTimeout`, used in chunks 145c and 146a.

## setRefCacheVertices(1)

60e ⟨*Admin: setRefCacheVertices(1)* 60e⟩≡ (158d)
```
  void setRefCacheVertices(final ConcurrentHashMap<Integer, int[]> lu_vertices) {
    this.lu_vertices = lu_vertices;
  }
```
Defines:
  `setRefCacheVertices`, used in chunks 62f and 144c.

## setRefCacheEdges(1)

61a ⟨*Admin: setRefCacheEdges(1)* 61a⟩≡ (158d)
```
void setRefCacheEdges(final ConcurrentHashMap<Integer, ConcurrentHashMap<Integer, int[]>> lu_edges) {
  this.lu_edges = lu_edges;
}
```
Defines:
`setRefCacheEdges`, used in chunks 62d and 144c.

## setRefCacheUsers(1)

61b ⟨*Admin: setRefCacheUsers(1)* 61b⟩≡ (158d)
```
void setRefCacheUsers(final ConcurrentHashMap<Integer, int[]> lu_users) {
  this.lu_users = lu_users;
}
```
Defines:
`setRefCacheUsers`, used in chunk 62e.

## setRefClient(1)

61c ⟨*Admin: setRefClient(1)* 61c⟩≡ (141c)
```
void setRefClient(final Client client) {
  this.client = client;
}
```
Defines:
`setRefClient`, used in chunks 167b and 190g.

## setRefCommunicator(1)

61d ⟨*Admin: setRefCommunicator(1)* 61d⟩≡ (151b)
```
void setRefCommunicator(final Communicator communicator) {
  this.communicator = communicator;
}
```
Defines:
`setRefCommunicator`, used in chunk 62c.

## setRefController(1)

61e ⟨*Admin: setRefController(1)* 61e⟩≡ (149c)
```
void setRefController(final Controller controller) {
  this.controller = controller;
}
```
Defines:
`setRefController`, used in chunk 140a.

## setRefStorage(1)

61f ⟨*Admin: setRefStorage(1)* 61f⟩≡ (149c)
```
void setRefStorage(final Storage storage) {
  this.storage = storage;
}
```
Defines:
`setRefStorage`, used in chunk 140a.

### setRefTraffic(1)

62a ⟨*Admin: setRefTraffic(1) 62a*⟩≡ (149c)
```
  void setRefTraffic (final Traffic traffic) {
    this.traffic = traffic;
    this.traffic.forwardRefCacheVertices(this.storage.getRefCacheVertices());
    this.traffic.forwardRefCacheEdges(this.storage.getRefCacheEdges());
  }
```
Defines:
`setRefTraffic`, used in chunk 62b.
Uses `forwardRefCacheEdges` 62d, `forwardRefCacheVertices` 62f, `getRefCacheEdges` 57d, and `getRefCacheVertices` 58a.

### forwardRefTraffic(1)

62b ⟨*Admin: forwardRefTraffic(1) 62b*⟩≡ (141c)
```
  void forwardRefTraffic(final Traffic traffic) {
    this.communicator.setRefTraffic(traffic);
  }
```
Defines:
`forwardRefTraffic`, used in chunks 167b and 191a.
Uses `setRefTraffic` 62a.

### forwardRefCommunicator(1)

62c ⟨*Admin: forwardRefCommunicator(1) 62c*⟩≡ (141c)
```
  void forwardRefCommunicator(final Communicator communicator) {
    this.client.setRefCommunicator(communicator);
  }
```
Defines:
`forwardRefCommunicator`, used in chunks 167b and 190g.
Uses `setRefCommunicator` 61d.

### forwardRefCacheEdges(1)

62d ⟨*Admin: forwardRefCacheEdges(1) 62d*⟩≡ (151b 155e)
```
  void forwardRefCacheEdges(final ConcurrentHashMap<Integer, ConcurrentHashMap<Integer, int[]>> lu_edges) {
    this.tools.setRefCacheEdges(lu_edges);
  }
```
Defines:
`forwardRefCacheEdges`, used in chunks 62a, 167b, 190g, and 191a.
Uses `setRefCacheEdges` 61a.

### forwardRefCacheUsers(1)

62e ⟨*Admin: forwardRefCacheUsers(1) 62e*⟩≡ (151b)
```
  void forwardRefCacheUsers(final ConcurrentHashMap<Integer, int[]> lu_users) {
    this.tools.setRefCacheUsers(lu_users);
  }
```
Defines:
`forwardRefCacheUsers`, used in chunks 167b and 190g.
Uses `setRefCacheUsers` 61b.

### forwardRefCacheVertices(1)

62f ⟨*Admin: forwardRefCacheVertices(1) 62f*⟩≡ (151b 155e)
```
  void forwardRefCacheVertices(final ConcurrentHashMap<Integer, int[]> lu_vertices) {
    this.tools.setRefCacheVertices(lu_vertices);
  }
```
Defines:
`forwardRefCacheVertices`, used in chunks 62a, 167b, 190g, and 191a.
Uses `setRefCacheVertices` 60e.

### 3.2.6 Exceptions

`DuplicateEdgeException`

63a ⟨*DuplicateEdgeException.java* 63a⟩≡

```
  ⟨Package: sim 35a⟩
  public class DuplicateEdgeException extends Exception {
    public DuplicateEdgeException ⟨Construct empty exception 36c⟩
    public DuplicateEdgeException ⟨Construct message exception 36d⟩
    public DuplicateEdgeException ⟨Construct cause exception 36e⟩
    public DuplicateEdgeException ⟨Construct message and cause exception 36f⟩
  }
```

Defines:
`DuplicateEdgeException`, used in chunks 121, 129b, 134b, 144b, 148b, and 166.

`DuplicateUserException`

63b ⟨*DuplicateUserException.java* 63b⟩≡

```
  ⟨Package: sim 35a⟩
  public class DuplicateUserException extends Exception {
    public DuplicateUserException ⟨Construct empty exception 36c⟩
    public DuplicateUserException ⟨Construct message exception 36d⟩
    public DuplicateUserException ⟨Construct cause exception 36e⟩
    public DuplicateUserException ⟨Construct message and cause exception 36f⟩
  }
```

Defines:
`DuplicateUserException`, used in chunks 122–24, 129b, 134b, 144d, and 148b.

`DuplicateVertexException`

63c ⟨*DuplicateVertexException.java* 63c⟩≡

```
  ⟨Package: sim 35a⟩
  public class DuplicateVertexException extends Exception {
    public DuplicateVertexException ⟨Construct empty exception 36c⟩
    public DuplicateVertexException ⟨Construct message exception 36d⟩
    public DuplicateVertexException ⟨Construct cause exception 36e⟩
    public DuplicateVertexException ⟨Construct message and cause exception 36f⟩
  }
```

Defines:
`DuplicateVertexException`, used in chunks 120, 129b, 134b, 143f, and 148b.

`EdgeNotFoundException`

63d ⟨*EdgeNotFoundException.java* 63d⟩≡

```
  ⟨Package: sim 35a⟩
  public class EdgeNotFoundException extends Exception {
    public EdgeNotFoundException ⟨Construct empty exception 36c⟩
    public EdgeNotFoundException ⟨Construct message exception 36d⟩
    public EdgeNotFoundException ⟨Construct cause exception 36e⟩
    public EdgeNotFoundException ⟨Construct message and cause exception 36f⟩
  }
```

Defines:
`EdgeNotFoundException`, used in chunks 71, 118a, 122–25, 129b, 134b, 144d, 148b, and 157b.

`GtreeIllegalSourceException`

63e ⟨*GtreeIllegalSourceException.java* 63e⟩≡

```
  ⟨Package: sim 35a⟩
  public class GtreeIllegalSourceException extends Exception {
    public GtreeIllegalSourceException ⟨Construct empty exception 36c⟩
    public GtreeIllegalSourceException ⟨Construct message exception 36d⟩
    public GtreeIllegalSourceException ⟨Construct cause exception 36e⟩
    public GtreeIllegalSourceException ⟨Construct message and cause exception 36f⟩
  }
```
Defines:
  `GtreeIllegalSourceException`, used in chunks 124b, 134b, 144d, 157b, 160, and 161.

## GtreeIllegalTargetException

```
64a  ⟨GtreeIllegalTargetException.java 64a⟩≡
       ⟨Package: sim 35a⟩
       public class GtreeIllegalTargetException extends Exception {
         public GtreeIllegalTargetException ⟨Construct empty exception 36c⟩
         public GtreeIllegalTargetException ⟨Construct message exception 36d⟩
         public GtreeIllegalTargetException ⟨Construct cause exception 36e⟩
         public GtreeIllegalTargetException ⟨Construct message and cause exception 36f⟩
       }
```
Defines:
  `GtreeIllegalTargetException`, used in chunks 124b, 134b, 144d, 157b, and 161b.

## GtreeNotLoadedException

```
64b  ⟨GtreeNotLoadedException.java 64b⟩≡
       ⟨Package: sim 35a⟩
       public class GtreeNotLoadedException extends Exception {
         public GtreeNotLoadedException ⟨Construct empty exception 36c⟩
         public GtreeNotLoadedException ⟨Construct message exception 36d⟩
         public GtreeNotLoadedException ⟨Construct cause exception 36e⟩
         public GtreeNotLoadedException ⟨Construct message and cause exception 36f⟩
       }
```
Defines:
  `GtreeNotLoadedException`, used in chunks 124b, 134b, 144d, 157b, 160, and 161.

## RouteIllegalOverwriteException

```
64c  ⟨RouteIllegalOverwriteException.java 64c⟩≡
       ⟨Package: sim 35a⟩
       public class RouteIllegalOverwriteException extends Exception {
         public RouteIllegalOverwriteException ⟨Construct empty exception 36c⟩
         public RouteIllegalOverwriteException ⟨Construct message exception 36d⟩
         public RouteIllegalOverwriteException ⟨Construct cause exception 36e⟩
         public RouteIllegalOverwriteException ⟨Construct message and cause exception 36f⟩
       }
```
Defines:
  `RouteIllegalOverwriteException`, used in chunks 125b and 148b.

## TimeWindowException

```
64d  ⟨TimeWindowException.java 64d⟩≡
       ⟨Package: sim 35a⟩
       public class TimeWindowException extends Exception {
         public TimeWindowException ⟨Construct empty exception 36c⟩
         public TimeWindowException ⟨Construct message exception 36d⟩
         public TimeWindowException ⟨Construct cause exception 36e⟩
         public TimeWindowException ⟨Construct message and cause exception 36f⟩
       }
```
Defines:
  `TimeWindowException`, used in chunks 115a, 125b, 129b, and 148b.

## UserNotFoundException

65a ⟨*UserNotFoundException.java* 65a⟩≡
```
  ⟨Package: sim 35a⟩
  public class UserNotFoundException extends Exception {
    public UserNotFoundException ⟨Construct empty exception 36c⟩
    public UserNotFoundException ⟨Construct message exception 36d⟩
    public UserNotFoundException ⟨Construct cause exception 36e⟩
    public UserNotFoundException ⟨Construct message and cause exception 36f⟩
  }
```
Defines:
`UserNotFoundException`, used in chunks 73, 74, 96b, 99e, 108a, 117, 118d, 124c, 125b, 129b, 134b, and 148b.

## VertexNotFoundException

65b ⟨*VertexNotFoundException.java* 65b⟩≡
```
  ⟨Package: sim 35a⟩
  public class VertexNotFoundException extends Exception {
    public VertexNotFoundException ⟨Construct empty exception 36c⟩
    public VertexNotFoundException ⟨Construct message exception 36d⟩
    public VertexNotFoundException ⟨Construct cause exception 36e⟩
    public VertexNotFoundException ⟨Construct message and cause exception 36f⟩
  }
```
Defines:
`VertexNotFoundException`, used in chunks 69c, 70a, 129b, 134b, 148b, 157b, 159d, 162a, 178, 181, and 182.

## 3.3 Read Operations

### 3.3.1 Chunks

#### Flatten results

67a ⟨*Flatten results* 67a⟩≡ (56a 67b 68a)

```
output = new int[(ncols*res.getRow())];
res.first();
int i = 0;
do {
  final int idx = i*ncols;
  for (int j = 1; j <= ncols; j++) {
    output[(idx + (j - 1))] = res.getInt(j);
  }
  i++;
} while (res.next());
```

### 3.3.2 Methods: General

#### DBQuery(2)

Method DBQuery(2) executes an arbitrary SELECT query against the Jargo database instance. A SQLException is thrown in case of database failure.

**Parameters:**

String sql (param. 1): SELECT statement to execute.
Integer ncols (param. 2): number of columns $n$ in the selection.

**Returns:** results of the query flattened into an integer array, or null if no results.

$\cdots \boxed{in + j : \text{value at column } j\text{, row } i \text{ of the result set}} \cdots$

where $i$, $j$ start from 0.
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

67b ⟨*Read: DBQuery(2)* 67b⟩≡ (131c)

```
int[] DBQuery(final String sql, final int ncols) throws SQLException {
  int[] output = new int[] { };
```

```
    try (⟨Open conn 35c⟩) {
      Statement stmt = conn.createStatement(
        ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
      ResultSet res = stmt.executeQuery(sql);
      if (res.last()) {
        ⟨Flatten results 67a⟩
      }
      conn.close();
    } catch (SQLException e) {
      throw e;
    }
    return output;
  }
```

Defines:
  DBQuery, used in chunk 68b.

Special version for DesktopController

68a ⟨*Read: DBQueryQuick(3)* 68a⟩≡ (131c)

```
  int[] DBQueryQuick(final String sql, int[] outcols, ArrayList<String> header) throws SQLException {
    int[] output = new int[] { };
    try (⟨Open conn 35c⟩) {
      Statement stmt = conn.createStatement(
        ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
      ResultSet res = stmt.executeQuery(sql);
      int ncols = res.getMetaData().getColumnCount();
      for (int i = 1; i <= ncols; i++) {
        header.add(res.getMetaData().getColumnName(i));
      }
      outcols[0] = ncols;
      if (res.last()) {
        ⟨Flatten results 67a⟩
      }
      conn.close();
    } catch (SQLException e) {
      throw e;
    }
    return output;
  }
```

Defines:
  DBQueryQuick, used in chunk 68c.

Method query(2) wraps DBQuery(2).

68b ⟨*Read: query(2)* 68b⟩≡ (140c)

```
  int[] query(final String sql, final int ncols) throws SQLException {
    int[] output = this.storage.DBQuery(sql, ncols);
    return output;
  }
```

Defines:
  query, used in chunks 96b, 137, 167b, and 195.
Uses DBQuery 67b.

68c ⟨*Read: queryQuick(3)* 68c⟩≡ (140c)

```
  int[] queryQuick(final String sql, int[] outcols, ArrayList<String> header) throws SQLException {
    long A0 = System.currentTimeMillis();
    int[] output = this.storage.DBQueryQuick(sql, outcols, header);
    this.dur_query = System.currentTimeMillis() - A0;
    return output;
  }
```

Defines:
  queryQuick, used in chunk 199.
Uses DBQueryQuick 68a.

### 3.3.3 Methods: Read Road Network

`DBQueryMBR`(0)

Method DBQueryMBR(0) returns the minimum-bounding rectangle of the road network. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

| | | | |
|---|---|---|---|
| 0 : min. longitude | 1 : max. longitude | 2 : min. latitude | 3 : max. latitude |

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

69a ⟨*Read: DBQueryMBR(0)* 69a⟩≡ (131c)
```
  int[] DBQueryMBR() throws SQLException {
    try (⟨Open conn 35c⟩) {
      return this.PSQuery(conn, "S64", 4);
    } catch (SQLException e) {
      throw e;
    }
  }
```
Defines:
  DBQueryMBR, used in chunk 69b.
Uses PSQuery 56a and S64 48k.

Method queryMBR(0) wraps DBQueryMBR(0).

69b ⟨*Read: queryMBR(0)* 69b⟩≡ (140c)
```
  int[] queryMBR() throws SQLException {
    int[] output = this.storage.DBQueryMBR();
    return output;
  }
```
Defines:
  queryMBR, used in chunk 205c.
Uses DBQueryMBR 69a.

`DBQueryVertex`(1)

Method DBQueryVertex(1) returns the longitude and latitude coordinates of the given vertex. If the vertex does not exist, a VertexNotFoundException is thrown.

**Parameters:**
Integer v (param. 1): vertex identifier.
**Returns:** results of the query flattened into an integer array, or null if no results.

| | |
|---|---|
| 0 : longitude of v | 1 : latitude of v |

**Side Effects:** none.
**Throws:** VertexNotFoundException if vertex does not exist

69c ⟨*Read: DBQueryVertex(1)* 69c⟩≡ (131c 158c)
```
  int[] DBQueryVertex(final int v) throws VertexNotFoundException {
    if (!this.lu_vertices.containsKey(v)) {
      throw new VertexNotFoundException("Vertex "+v+" not found.");
    }
    int[] output = this.lu_vertices.get(v).clone();
    return new int[] { output[0], output[1], (int) Storage.CSHIFT };
  }
```
Defines:
  DBQueryVertex, used in chunk 70a.
Uses VertexNotFoundException 65b.

Method queryVertex(1) wraps DBQueryVertex(1).

70a ⟨*Read: queryVertex(1)* 70a⟩≡ (140c 149a)
```
  int[] queryVertex(final int v) throws VertexNotFoundException, SQLException {
    int[] output = this.storage.DBQueryVertex(v);
    return output;
  }
```
Defines:
  queryVertex, used in chunks 178, 181, and 182.
Uses DBQueryVertex 69c and VertexNotFoundException 65b.

### DBQueryVertices(0)

Method DBQueryVertices(0) returns all rows in Table V. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

| | | | |
|---|---|---|---|
| 0 : vertex identifier | 1 : longitude of the vertex | 2 : latitude of the vertex | ··· |

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

70b ⟨*Read: DBQueryVertices(0)* 70b⟩≡ (131c)
```
  int[] DBQueryVertices() throws SQLException {
    try (⟨Open conn 35c⟩) {
      return this.PSQuery(conn, "S136", 3);
    } catch (SQLException e) {
      throw e;
    }
  }
```
Defines:
  DBQueryVertices, used in chunks 42a and 70c.
Uses PSQuery 56a and S136 52k.

Method queryVertices(2) wraps DBQueryVertices(2).

70c ⟨*Read: queryVertices(0)* 70c⟩≡ (140c)
```
  int[] queryVertices() throws SQLException {
    int[] output = this.storage.DBQueryVertices();
    return output;
  }
```
Defines:
  queryVertices, never used.
Uses DBQueryVertices 70b.

### DBQueryVerticesCount(0)

Method DBQueryVerticesCount(0) returns the total number of vertices in Table V. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

| |
|---|
| 0 : number of vertices in Table V |

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

70d ⟨*Read: DBQueryVerticesCount(0)* 70d⟩≡ (131c)
```
  int[] DBQueryVerticesCount() throws SQLException {
    try (⟨Open conn 35c⟩) {
      return this.PSQuery(conn, "S62", 1);
    } catch (SQLException e) {
      throw e;
    }
  }
```

Defines:
  DBQueryVerticesCount, used in chunk 71a.
Uses PSQuery 56a and S62 48j.

Method queryVerticesCount(0) wraps DBQueryVerticesCount(0).

71a ⟨*Read: queryVerticesCount(0)* 71a⟩≡ (140c)

```
  int[] queryVerticesCount() throws SQLException {
    int[] output = this.storage.DBQueryVerticesCount();
    return output;
  }
```

Defines:
  queryVerticesCount, used in chunks 193 and 194.
Uses DBQueryVerticesCount 70d.

### DBQueryEdge(**2**)

Method DBQueryEdge(2) returns the distance and maximum free-flow speed along the given edge. An EdgeNotFoundException is thrown if the edge does not exist.

**Parameters:**

| | |
|---|---|
| Integer v1 (param. 1): | source vertex identifier $v_1$ |
| Integer v2 (param. 2): | target vertex identifier $v_2$ |

**Returns:** results of the query flattened into an integer array, or null if no results.

| | |
|---|---|
| $0 : d(v_1, v_2)$ | $1 : v^{\max}(v_1, v_2)$ |

**Side Effects:** none.
**Throws:** EdgeNotFoundException if edge does not exit.

71b ⟨*Read: DBQueryEdge(2)* 71b⟩≡ (131c 158c)

```
  int[] DBQueryEdge(final int v1, final int v2) throws EdgeNotFoundException {
    if (v1 == v2) {
      return new int[] { 0, -1 };  // 0 distance, -1 speed
    }
    if (!(this.lu_edges.containsKey(v1) && this.lu_edges.get(v1).containsKey(v2))) {
      throw new EdgeNotFoundException("Edge ("+v1+", "+v2+") not found.");
    }
    return this.lu_edges.get(v1).get(v2).clone();
  }
```

Defines:
  DBQueryEdge, used in chunks 71c and 114.
Uses EdgeNotFoundException 63d.

Method queryEdge(2) wraps DBQueryEdge(2).

71c ⟨*Read: queryEdge(2)* 71c⟩≡ (140c 149a)

```
  int[] queryEdge(final int v1, final int v2) throws EdgeNotFoundException, SQLException {
    int[] output = this.storage.DBQueryEdge(v1, v2);
    return output;
  }
```

Defines:
  queryEdge, never used.
Uses DBQueryEdge 71b and EdgeNotFoundException 63d.

## DBQueryEdges(0)

Method DBQueryEdges(0) returns all rows in Table E. A `SQLException` is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

|  |  |  |  |  |
|---|---|---|---|---|
| 0 : source vertex identifier $v_1$ | 1 : target vertex identifier $v_2$ | 2 : $d(v_1, v_2)$ | 3 : $v^{\max}(v_1, v_2)$ | ··· |

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

72a ⟨*Read: DBQueryEdges(0)* 72a⟩≡ (131c)
```
  int[] DBQueryEdges() throws SQLException {
    try (⟨Open conn 35c⟩) {
      return this.PSQuery(conn, "S137", 4);
    } catch (SQLException e) {
      throw e;
    }
  }
```
Defines:
`DBQueryEdges`, used in chunks 42b and 72b.
Uses `PSQuery` 56a and `S137` 52l.

Method queryEdges(2) wraps DBQueryEdges(2).

72b ⟨*Read: queryEdges(0)* 72b⟩≡ (140c)
```
  int[] queryEdges() throws SQLException {
    int[] output = this.storage.DBQueryEdges();
    return output;
  }
```
Defines:
`queryEdges`, used in chunk 178.
Uses `DBQueryEdges` 72a.

## DBQueryEdgesCount(0)

Method DBQueryEdgesCount(0) returns the total number of vertices in Table V. A `SQLException` is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

|  |
|---|
| 0 : number of edges in Table E |

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

72c ⟨*Read: DBQueryEdgesCount(0)* 72c⟩≡ (131c)
```
  int[] DBQueryEdgesCount() throws SQLException {
    try (⟨Open conn 35c⟩) {
      return this.PSQuery(conn, "S63", 1);
    } catch (SQLException e) {
      throw e;
    }
  }
```
Defines:
`DBQueryEdgesCount`, used in chunk 72d.
Uses `PSQuery` 56a and `S63` 48l.

Method queryEdgesCount(0) wraps DBQueryEdgesCount(0).

72d ⟨*Read: queryEdgesCount(0)* 72d⟩≡ (140c)
```
  int[] queryEdgesCount() throws SQLException {
    int[] output = this.storage.DBQueryEdgesCount();
    return output;
  }
```

Defines:
  `queryEdgesCount`, used in chunks 193 and 194.
Uses `DBQueryEdgesCount` 72c.

`DBQueryEdgeStatistics`**(0)**

Method `DBQueryEdgeStatistics`(0) returns some edge statistics. A `SQLException` is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

| 0 : min. weight | 1 : max. weight | 2 : avg. weight | 3 : min. speed | 4 : max. speed | 5 : avg. speed |
|---|---|---|---|---|---|

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

73a ⟨*Read: DBQueryEdgeStatistics(0)* 73a⟩≡ (131c)

```
  int[] DBQueryEdgeStatistics() throws SQLException {
    try (⟨Open conn 35c⟩) {
      return this.PSQuery(conn, "S65", 6);
    } catch (SQLException e) {
      throw e;
    }
  }
```

Defines:
  `DBQueryEdgeStatistics`, used in chunk 73b.
Uses `PSQuery` 56a and `S65` 48m.

Method `queryEdgeStatistics`(0) wraps `DBQueryEdgeStatistics`(0).

73b ⟨*Read: queryEdgeStatistics(0)* 73b⟩≡ (140c)

```
  int[] queryEdgeStatistics() throws SQLException {
    int[] output = storage.DBQueryEdgeStatistics();
    return output;
  }
```

Defines:
  `queryEdgeStatistics`, never used.
Uses `DBQueryEdgeStatistics` 73a.

### 3.3.4 Methods: Read User Properties

`DBQueryUser`**(1)**

Method `DBQueryUser`(0) returns the properties of the given user. A `UserNotFoundException` is thrown if the user does not exist.

**Parameters:**
 Integer `uid` (param. 1): user identifier for user $u$
**Returns:** results of the query flattened into an integer array, or `null` if no results.

| 0 : user identifier | 1 : $u_\mathrm{q}$ | 2 : $u_\mathrm{e}$ | 3 : $u_1$ | 4 : $u_\mathrm{o}$ | 5 : $u_\mathrm{d}$ | 6 : $d_u$ |
|---|---|---|---|---|---|---|

**Side Effects:** none.
**Throws:** `UserNotFoundException` if user does not exist.

73c ⟨*Read: DBQueryUser(1)* 73c⟩≡ (131c)

```
  int[] DBQueryUser(final int uid)
  throws UserNotFoundException {
    if (!this.lu_users.containsKey(uid)) {
      throw new UserNotFoundException("User "+uid+" not found.");
    }
    return this.lu_users.get(uid).clone();
  }
```

Defines:

DBQueryUser, used in chunks 74a, 96b, 99e, 108a, and 115a.
Uses UserNotFoundException 65a.

Method queryUser(1) wraps DBQueryUser(1).

74a ⟨*Read: queryUser(1)* 74a⟩≡ (140c 149a)
```
int[] queryUser(final int rid) throws UserNotFoundException, SQLException {
  int[] output = storage.DBQueryUser(rid);
  return output;
}
```
Defines:
queryUser, never used.
Uses DBQueryUser 73c and UserNotFoundException 65a.

DBQueryUsers(0)

Method DBQueryUsers(0) returns all rows in view r_user. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

| 0 : user identifier for user $u$ | 1 : $u_\mathsf{q}$ | 2 : $u_\mathsf{e}$ | 3 : $u_\mathsf{l}$ | 4 : $u_\mathsf{o}$ | 5 : $u_\mathsf{d}$ | 6 : $d_u$ | ⋯ |
|---|---|---|---|---|---|---|---|

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

74b ⟨*Read: DBQueryUsers(0)* 74b⟩≡ (131c)
```
int[] DBQueryUsers() throws SQLException {
  try (⟨Open conn 35c⟩) {
    return this.PSQuery(conn, "S141", 7);
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
DBQueryUsers, used in chunk 43b.
Uses PSQuery 56a and S141 53b.

DBQueryRequestStatus(2)

Method DBQueryRequestStatus(0) returns the status of the given request at the given time (Eq. 2.1). A UserNotFoundException is thrown if the user does not exist. A SQLException is thrown in case of database failure.

**Parameters:**

| | |
|---|---|
| Integer rid (param. 1): | user identifier for request $r$ |
| Integer t (param. 2): | a time |

**Returns:** results of the query flattened into an integer array, or null if no results.

| 0 : status of $r$ at time $t$ |
|---|

**Side Effects:** none.
**Throws:** UserNotFoundException if user does not exist, or SQLException if database failure is encountered.

74c ⟨*Read: DBQueryRequestStatus(2)* 74c⟩≡ (131c)
```
int[] DBQueryRequestStatus(final int rid, final int t)
throws UserNotFoundException, SQLException {
  if (!this.lu_users.containsKey(rid)) {
    throw new UserNotFoundException("User "+rid+" not found.");
  }
  try (⟨Open conn 35c⟩) {
    return this.PSQuery(conn, "S133", 1, rid, t);
  } catch (SQLException e) {
    throw e;
  }
}
```

Defines:
 DBQueryRequestStatus, never used.
Uses PSQuery 56a, S133 52h, and UserNotFoundException 65a.

### DBQueryRequestIsAssigned(2)

TODO. This is really bad. If param 2 is false, the return is blank or the sid of the server assigned to request param 1. But if param 2 is true, the return is either blank or 1; no sid.

75a ⟨*Read: DBQueryRequestIsAssigned(2)* 75a⟩≡ (131c)

```
  int[] DBQueryRequestIsAssigned(final int rid, boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      return this.lu_rstatus.get(rid) ? new int[] { 1 } : new int[] { };
    } else {
      try (⟨Open conn 35c⟩) {
        return this.PSQuery(conn, "S148", 1, rid);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```

Defines:
 DBQueryRequestIsAssigned, used in chunks 43 and 108a.
Uses PSQuery 56a and S148 53h.

### DBQueryRequestDistanceDetour(2)

Method DBQueryRequestDistanceDetour(2) returns the detour distance $D^{\text{detour}}(\mathcal{X}, r)$ (Eq. 2.21) of the given request. A SQLException is thrown in case of database failure.

**Parameters:**
Integer rid (param. 1): request identifier.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : D^{\text{detour}}(\mathcal{X}, r)}$

where $r$ is the request identified by rid (param. 1).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

75b ⟨*Read: DBQueryRequestDistanceDetour(2)* 75b⟩≡ (131c)

```
  int[] DBQueryRequestDistanceDetour(final int rid, boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      return new int[] { this.distance_requests_transit.containsKey(rid)
        ? this.distance_requests_transit.get(rid) - this.lu_users.get(rid)[6]
        : 0 };
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S112", 1, rid);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```

Defines:
 DBQueryRequestDistanceDetour, never used.
Uses PSQuery 56a and S112 51f.

## DBQueryRequestDistanceTransit(2)

Method DBQueryRequestDistanceTransit(2) returns the transit distance $D^{\text{transit}}(\mathcal{X}, r)$ (Eq. 2.19) of the given request. A SQLException is thrown in case of database failure.

**Parameters:**
Integer rid (param. 1): request identifier.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : D^{\text{transit}}(\mathcal{X}, r)}$

where $r$ is the request identified by rid (param. 1).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

76a ⟨*Read: DBQueryRequestDistanceTransit(2)* 76a⟩≡ (131c)

```
int[] DBQueryRequestDistanceTransit(final int rid, boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    return new int[] { this.distance_requests_transit.containsKey(rid)
      ? this.distance_requests_transit.get(rid)
      : 0 };
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S114", 1, rid);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```

Defines:
DBQueryRequestDistanceTransit, used in chunk 36b.
Uses PSQuery 56a and S114 51h.

## DBQueryRequestDurationPickup(2)

Method DBQueryRequestDurationPickup(2) returns the pickup delay $\delta^{\text{pickup}}(\mathcal{X}, r)$ (Eq. 2.18) of the given request. A SQLException is thrown in case of database failure.

**Parameters:**
Integer rid (param. 1): request identifier.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \delta^{\text{pickup}}(\mathcal{X}, r)}$

where $r$ is the request identified by rid (param. 1).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

76b ⟨*Read: DBQueryRequestDurationPickup(2)* 76b⟩≡ (131c)

```
int[] DBQueryRequestDurationPickup(final int rid, boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    return new int[] { this.duration_requests_pickup.containsKey(rid)
        ? this.duration_requests_pickup.get(rid)
        : 0 };
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S118", 1, rid);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```

Defines:
DBQueryRequestDurationPickup, used in chunk 36b.
Uses PSQuery 56a and S118 51l.

DBQueryRequestDurationTransit(**1**)

Method DBQueryRequestDurationTransit(2) returns the transit duration $\delta^{\text{transit}}(\mathcal{X}, r)$ (Eq. 2.20) of the given request. A SQLException is thrown in case of database failure.

**Parameters:**
Integer rid (param. 1): request identifier.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \delta^{\text{transit}}(\mathcal{X}, r)}$

where $r$ is the request identified by rid (param. 1).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

77a ⟨*Read: DBQueryRequestDurationTransit(2)* 77a⟩≡ (131c)
```
  int[] DBQueryRequestDurationTransit(final int rid, boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      return new int[] { this.duration_requests_transit.containsKey(rid)
          ? this.duration_requests_transit.get(rid)
          : 0 };
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S120", 1, rid);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```
Defines:
DBQueryRequestDurationTransit, used in chunk 36b.
Uses PSQuery 56a and S120 51n.

DBQueryRequestDurationTravel(**2**)

Method DBQueryRequestDurationTravel(2) returns the travel duration $\delta^{\text{travel}}(\mathcal{X}, r)$ (Eq. 2.23) of the given request. A SQLException is thrown in case of database failure.

**Parameters:**
Integer rid (param. 1): request identifier.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \delta^{\text{travel}}(\mathcal{X}, r)}$

where $r$ is the request identified by rid (param. 1).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

77b ⟨*Read: DBQueryRequestDurationTravel(2)* 77b⟩≡ (131c)
```
  int[] DBQueryRequestDurationTravel(final int rid, boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      return new int[] { this.duration_requests_travel.containsKey(rid)
          ? this.duration_requests_travel.get(rid)
          : 0 };
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S122", 1, rid);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```
Defines:
DBQueryRequestDurationTravel, used in chunk 36b.
Uses PSQuery 56a and S122 52b.

### DBQueryRequestTimeOfDeparture(1)

Method DBQueryRequestTimeOfDeparture(1) returns the departure time $t^{\text{depart}}(\mathcal{X}, r)$ (Eq. 2.16) of the given request. A SQLException is thrown in case of database failure.

**Parameters:**
Integer rid (param. 1): request identifier.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : t^{\text{depart}}(\mathcal{X}, r)}$

where $r$ is the request identified by rid (param. 1).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

78a ⟨*Read: DBQueryRequestTimeOfDeparture(1)* 78a⟩≡ (131c)
```
  int[] DBQueryRequestTimeOfDeparture(final int rid) throws SQLException {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S124", 1, rid);
    } catch (SQLException e) {
      throw e;
    }
  }
```
Defines:
DBQueryRequestTimeOfDeparture, used in chunk 78b.
Uses PSQuery 56a and S124 52d.

Method queryRequestTimeOfDeparture(1) wraps DBQueryRequestTimeOfDeparture(1).

78b ⟨*Read: queryRequestTimeOfDeparture(1)* 78b⟩≡ (140c)
```
  int[] queryRequestTimeOfDeparture(final int rid) throws SQLException {
    int[] output = storage.DBQueryRequestTimeOfDeparture(rid);
    return output;
  }
```
Defines:
queryRequestTimeOfDeparture, never used.
Uses DBQueryRequestTimeOfDeparture 78a.

### DBQueryRequestTimeOfArrival(1)

Method DBQueryRequestTimeOfArrival(1) returns the arrival time $t^{\text{arrive}}(\mathcal{X}, r)$ (Eq. 2.17) of the given request. A SQLException is thrown in case of database failure.

**Parameters:**
Integer rid (param. 1): request identifier.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : t^{\text{arrive}}(\mathcal{X}, r)}$

where $r$ is the request identified by rid (param. 1).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

78c ⟨*Read: DBQueryRequestTimeOfArrival(1)* 78c⟩≡ (131c)
```
  int[] DBQueryRequestTimeOfArrival(final int rid) throws SQLException {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S126", 1, rid);
    } catch (SQLException e) {
      throw e;
    }
  }
```
Defines:
DBQueryRequestTimeOfArrival, used in chunk 79a.
Uses PSQuery 56a and S126 52f.

Method queryRequestTimeOfArrival(1) wraps DBQueryRequestTimeOfArrival(1).

79a ⟨*Read: queryRequestTimeOfArrival(1)* 79a⟩≡ (140c)
```
int[] queryRequestTimeOfArrival(final int rid) throws SQLException {
  int[] output = storage.DBQueryRequestTimeOfArrival(rid);
  return output;
}
```
Defines:
`queryRequestTimeOfArrival`, never used.
Uses `DBQueryRequestTimeOfArrival` 78c.

## `DBQueryRequestsCount`(0)

Method DBQueryRequestsCount(0) returns the total number of requests in Table R. A `SQLException` is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

| |
|---|
| 0 : number of requests in Table R |

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

79b ⟨*Read: DBQueryRequestsCount(0)* 79b⟩≡ (131c)
```
int[] DBQueryRequestsCount() throws SQLException {
  // try (⟨Open conn 35c⟩) {
  //   return this.PSQuery(conn, "S67", 1);
  // } catch (SQLException e) {
  //   throw e;
  // }
  return new int[] { this.count_requests };
}
```
Defines:
`DBQueryRequestsCount`, used in chunk 79c.
Uses `PSQuery` 56a and `S67` 49f.

Method queryRequestsCount(0) wraps DBQueryRequestsCount(0).

79c ⟨*Read: queryRequestsCount(0)* 79c⟩≡ (140c)
```
int[] queryRequestsCount() throws SQLException {
  int[] output = storage.DBQueryRequestsCount();
  return output;
}
```
Defines:
`queryRequestsCount`, used in chunks 193 and 195.
Uses `DBQueryRequestsCount` 79b.

## `DBQueryRequestsCountActive`(1)

79d ⟨*Read: DBQueryRequestsCountActive(1)* 79d⟩≡ (131c)
```
int[] DBQueryRequestsCountActive(final int t) throws SQLException {
  try (⟨Open conn 35c⟩) {
    return this.PSQuery(conn, "S161", 1, t, t, t);
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
`DBQueryRequestsCountActive`, used in chunk 80a.
Uses `PSQuery` 56a and `S161` 54g.

80a ⟨*Read: queryRequestsCountActive(1)* 80a⟩≡ (140c)
```
int[] queryRequestsCountActive(final int t) throws SQLException {
  int[] output = storage.DBQueryRequestsCountActive(t);
  return output;
}
```
Defines:
  `queryRequestsCountActive`, used in chunk 189h.
Uses `DBQueryRequestsCountActive` 79d.

### DBQueryRequestsCountAppeared(0)

80b ⟨*Read: DBQueryRequestsCountAppeared(0)* 80b⟩≡
```
int[] DBQueryRequestsCountAppeared() throws SQLException {
  int[] output = new int[] { };
  // ...
  return output;
}
```
Defines:
  `DBQueryRequestsCountAppeared`, never used.

80c ⟨*Read: queryRequestsCountAppeared(0)* 80c⟩≡ (140c)
```
int[] queryRequestsCountAppeared() throws SQLException {
  int[] output = new int[] { this.lu_rseen.size() };
  return output;
}
```
Defines:
  `queryRequestsCountAppeared`, used in chunk 183.

### DBQueryRequestsCountAssigned(0)

80d ⟨*Read: DBQueryRequestsCountAssigned(0)* 80d⟩≡ (131c)
```
int[] DBQueryRequestsCountAssigned() throws SQLException {
  return new int[] { this.count_assigned };
}
```
Defines:
  `DBQueryRequestsCountAssigned`, used in chunks 80e and 98d.

80e ⟨*Read: queryRequestsCountAssigned(0)* 80e⟩≡ (140c)
```
int[] queryRequestsCountAssigned() throws SQLException {
  int[] output = storage.DBQueryRequestsCountAssigned();
  return output;
}
```
Defines:
  `queryRequestsCountAssigned`, never used.
Uses `DBQueryRequestsCountAssigned` 80d.

### DBQueryRequestsCountCompleted(1)

80f ⟨*Read: DBQueryRequestsCountCompleted(1)* 80f⟩≡ (131c)
```
int[] DBQueryRequestsCountCompleted(final int t) throws SQLException {
  try (⟨Open conn 35c⟩) {
    return this.PSQuery(conn, "S161", 1, t, t, t);
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
  `DBQueryRequestsCountCompleted`, used in chunk 81a.
Uses `PSQuery` 56a and `S161` 54g.

81a ⟨*Read: queryRequestsCountCompleted(1)* 81a⟩≡ (140c)
```
int[] queryRequestsCountCompleted(final int t) throws SQLException {
  int[] output = storage.DBQueryRequestsCountCompleted(t);
  return output;
}
```
Defines:
  `queryRequestsCountCompleted`, used in chunk 189i.
Uses `DBQueryRequestsCountCompleted` 80f.

### `DBQueryRequestsQueued(1)`

Method `DBQueryRequestsQueued`(1) returns the requests eligible for assignment at the given time. A request $r$ is "eligible" if it is not assigned at the given time, and if the given time is between the request's early time $r_e$ and ($r_e$ + `REQUEST_TIMEOUT`). A `SQLException` is thrown in case of database failure.

**Parameters:**
Integer `t` (param. 1): a time
**Returns:** results of the query flattened into an integer array, or `null` if no results.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 : user identifier for user $u$ | 1 : $u_q$ | 2 : $u_e$ | 3 : $u_l$ | 4 : $u_o$ | 5 : $u_d$ | 6 : $d_u$ | ··· |

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

81b ⟨*Read: DBQueryRequestsQueued(1)* 81b⟩≡ (131c) 81c▷
```
int[] DBQueryRequestsQueued(final int t) throws SQLException {
  try (⟨Open conn 35c⟩) {
```
Defines:
  `DBQueryRequestsQueued`, used in chunks 81e and 137.

Our approach is to first select all requests where $t$ is between the request's early time $r_e$ and $r_e$+`REQUEST_TIMEOUT`. Then, we return a filtered subset of these requests that are unassigned. As we don't know how many requests will returned in the end, we initialize a temporary array `temp1` to hold the pre-filter number of requests.

81c ⟨*Read: DBQueryRequestsQueued(1)* 81b⟩+≡ (131c) ◁81b 81d▷
```
      final int[] output = this.PSQuery(conn, "S143", 7, t, t, REQUEST_TIMEOUT);
      int[] temp1 = new int[output.length];
      int j = 0;
      for (int i = 0; i < (output.length - 6); i += 7) {
        if (this.lu_rstatus.get(output[i]) == false) {
          temp1[(j + 0)] = output[(i + 0)];
          temp1[(j + 1)] = output[(i + 1)];
          temp1[(j + 2)] = output[(i + 2)];
          temp1[(j + 3)] = output[(i + 3)];
          temp1[(j + 4)] = output[(i + 4)];
          temp1[(j + 5)] = output[(i + 5)];
          temp1[(j + 6)] = output[(i + 6)];
          j += 7;
        }
      }
```
Uses `PSQuery` 56a and `S143` 53d.

81d ⟨*Read: DBQueryRequestsQueued(1)* 81b⟩+≡ (131c) ◁81c
```
      return Arrays.copyOf(temp1, j);
    } catch (SQLException e) {
      throw e;
    }
  }
```

Method `queryRequestsQueued`(1) wraps `DBQueryRequestsQueued`(1).

81e ⟨*Read: queryRequestsQueued(1)* 81e⟩≡ (140c)
```
int[] queryRequestsQueued(final int t) throws SQLException {
  int[] output = storage.DBQueryRequestsQueued(t);
  return output;
}
```

Defines:
  `queryRequestsQueued`, never used.
Uses `DBQueryRequestsQueued` 81b.

### `DBQueryRequestsWaiting(1)`

82a ⟨*Read: DBQueryRequestsWaiting(1)* 82a⟩≡ (131c)

```
  int[] DBQueryRequestsWaiting(final int t) throws SQLException {
    try (⟨Open conn 35c⟩) {
      return this.PSQuery(conn, "S162", 2, t, t, REQUEST_TIMEOUT, t);
    } catch (SQLException e) {
      throw e;
    }
  }
```

Defines:
  `DBQueryRequestsWaiting`, used in chunk 82b.
Uses `PSQuery` 56a and `S162` 54h.

82b ⟨*Read: queryRequestsWaiting(1)* 82b⟩≡ (140c)

```
  int[] queryRequestsWaiting(final int t) throws SQLException {
    long A0 = System.currentTimeMillis();
    int[] output = storage.DBQueryRequestsWaiting(t);
    return output;
  }
```

Defines:
  `queryRequestsWaiting`, used in chunk 181.
Uses `DBQueryRequestsWaiting` 82a.

### `DBQueryServerRoute(1)`

Method `DBQueryServerRoute`(1) returns the route for the given server identified by `sid` (param. 1) at time $t$ (param. 2). A `SQLException` is thrown in case of database failure.

**Parameters:**
 Integer `sid` (param. 1): server identifier.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

| ⋯ | $2(i-1) : \pi_t(w_i)$ | $2(i-1)+1 : \pi_v(w_i)$ | ⋯ |
|---|---|---|---|

where $1 \leq i \leq |w|$ and $w$ is the route for the given server identified by `sid` (param. 1).
**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

82c ⟨*Read: DBQueryServerRoute(1)* 82c⟩≡ (131c)

```
  int[] DBQueryServerRoute(final int sid) throws SQLException {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S60", 2, sid);
    } catch (SQLException e) {
      throw e;
    }
  }
```

Defines:
  `DBQueryServerRoute`, used in chunks 82d and 125b.
Uses `PSQuery` 56a and `S60` 49i.

Method `queryServerRoute`(1) wraps `DBQueryServerRoute`(1).

82d ⟨*Read: queryServerRoute(1)* 82d⟩≡ (140c)

```
  int[] queryServerRoute(final int sid) throws SQLException {
    int[] output = storage.DBQueryServerRoute(sid);
    return output;
  }
```

Defines:
  `queryServerRoute`, never used.
Uses `DBQueryServerRoute` 82c.

### DBQueryServerRouteRemaining(2)

Method DBQueryServerRouteRemaining(2) returns the remaining route for the given server at the given time. A SQLException is thrown in case of database failure.

**Parameters:**
Integer sid (param. 1): server identifier.
Integer t (param. 2): a time.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\cdots \boxed{2(i-1) : \pi_{\mathtt{t}}((w_{>t})_i)}\boxed{2(i-1)+1 : \pi_{\mathtt{v}}((w_{>t})_i)} \cdots$

where $1 \leq i \leq |w_{>t}|$ and $w_{>t}$ is the remaining route for the given server identified by sid (param. 1) at time $t$ (param. 2).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

83a ⟨*Read: DBQueryServerRouteRemaining(2)* 83a⟩≡ (131c)
```
int[] DBQueryServerRouteRemaining(final int sid, final int t) throws SQLException {
  try (⟨Open conn 35c⟩) {
    return PSQuery(conn, "S129", 2, sid, t);
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
DBQueryServerRouteRemaining, used in chunk 83b.
Uses PSQuery 56a and S129 49j.

Method queryServerRouteRemaining(2) wraps DBQueryServerRouteRemaining(2).

83b ⟨*Read: queryServerRouteRemaining(2)* 83b⟩≡ (140c 149a)
```
int[] queryServerRouteRemaining(final int sid, final int t) throws SQLException {
  int[] output = this.storage.DBQueryServerRouteRemaining(sid, t);
  return output;
}
```
Defines:
queryServerRouteRemaining, never used.
Uses DBQueryServerRouteRemaining 83a.

### DBQueryServerRouteActive(1)

Gets the "active" route, used for Desktop interpolation. The active route includes the last-visited vertex, the "active" vertex that the server is currently traveling to, and the next vertex after the active vertex.

83c ⟨*Read: DBQueryServerRouteActive(1)* 83c⟩≡ (131c)
```
int[] DBQueryServerRouteActive(final int sid) throws SQLException {
  try (⟨Open conn 35c⟩) {
    return PSQuery(conn, "S152", 2, sid, this.lu_lvt.get(sid), 3);
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
DBQueryServerRouteActive, used in chunk 83d.
Uses PSQuery 56a and S152 53l.

83d ⟨*Read: queryServerRouteActive(1)* 83d⟩≡ (140c 149a)
```
int[] queryServerRouteActive(final int sid) throws SQLException {
  int[] output = this.storage.DBQueryServerRouteActive(sid);
  return output;
}
```
Defines:
queryServerRouteActive, used in chunk 182.
Uses DBQueryServerRouteActive 83c.

### `DBQueryServerSchedule`(1)

Method `DBQueryServerSchedule`(1) returns the schedule for the given server. A `SQLException` is thrown in case of database failure.

**Parameters:**
Integer `sid` (param. 1): server identifier.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\cdots \boxed{4(j-1) : \pi_{\mathrm{t}}(b_j)}\boxed{4(j-1)+1 : \pi_{\mathrm{v}}(b_j)}\boxed{4(j-1)+2 : \text{a server ID in } \pi_{\mathrm{L}}(b_j)}\boxed{4(j-1)+3 : \text{a request ID in } \pi_{\mathrm{L}}(b_j)} \cdots$

where $1 \leq j \leq |b|$ and $b$ is the schedule for the given server identified by `sid` (param. 1). If a label is empty (*e.g.* not all waypoints will have a server identifier in their label set), the element will be 0. If a waypoint has multiple labels, the waypoint will be written once for each of the labels. The returned sequence is in time-ascending order but **is not guaranteed** to be in the same order as the actual pick-ups and drop-offs, *e.g.* if a waypoint has multiple labels with some indicating pick-ups and some indicating drop-offs, the ordering of these waypoints is uncertain.
**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

84a ⟨*Read: DBQueryServerSchedule(1)* 84a⟩≡ (131c)
```
  int[] DBQueryServerSchedule(final int sid) throws SQLException {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S61", 4, sid);
    } catch (SQLException e) {
      throw e;
    }
  }
```
Defines:
`DBQueryServerSchedule`, used in chunk 84b.
Uses `PSQuery` 56a and `S61` 49k.

Method `queryServerSchedule`(1) wraps `DBQueryServerSchedule`(1).

84b ⟨*Read: queryServerSchedule(1)* 84b⟩≡ (140c)
```
  int[] queryServerSchedule(final int sid) throws SQLException {
    int[] output = storage.DBQueryServerSchedule(sid);
    return output;
  }
```
Defines:
`queryServerSchedule`, never used.
Uses `DBQueryServerSchedule` 84a.

### `DBQueryServerScheduleRemaining`(2)

Method `DBQueryServerScheduleRemaining`(2) returns the remaining schedule for the given server at the given time. A `SQLException` is thrown in case of database failure.

**Parameters:**
Integer `sid` (param. 1): server identifier.
Integer `t` (param. 2): a time.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\cdots \boxed{4(j-1) : \pi_{\mathrm{t}}((b_{>t})_j)}\boxed{4(j-1)+1 : \pi_{\mathrm{v}}((b_{>t})_j)}\boxed{4(j-1)+2 : \text{a server ID in } \pi_{\mathrm{L}}((b_{>t})_j)}\boxed{4(j-1)+3 : \text{a request ID in } \pi_{\mathrm{L}}((b_{>t})_j)} \cdots$

where $1 \leq j \leq |b_{>t}|$ and $b_{>t}$ is the remaining schedule for the given server identified by `sid` (param. 1) at time $t$ (param. 2). If a label is empty (*e.g.* not all waypoints will have a server identifier in their label set), the element will be 0. If a waypoint has multiple labels, the waypoint will be written once for each of the labels. The returned sequence is in time-ascending order and **is guaranteed** to be in the same order as the actual pick-ups and drop-offs.
**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

84c ⟨*Read: DBQueryServerScheduleRemaining(2)* 84c⟩≡ (131c)
```
  int[] DBQueryServerScheduleRemaining(final int sid, final int t)
  throws SQLException {
    int[] output = new int[] { };
    try (⟨Open conn 35c⟩) {
      int[] temp = PSQuery(conn, "S144", 3, sid, t);
      output = new int[(4*temp.length/3 + 4)];
      int j = 0;
      for (int i = 0; i < (temp.length - 2); i += 3) {
        output[(j + 0)] = temp[(i + 0)];
        output[(j + 1)] = temp[(i + 1)];
        output[(j + 2)] = 0;
        output[(j + 3)] = temp[(i + 2)];
        j += 4;
      }
      temp = PSQuery(conn, "S145", 2, sid);
      output[(j + 0)] = temp[0];
      output[(j + 1)] = temp[1];
      output[(j + 2)] = sid;
      output[(j + 3)] = 0;
    } catch (SQLException e) {
      throw e;
    }
    return output;
  }
```

Defines:
`DBQueryServerScheduleRemaining`, used in chunk 85a.
Uses `PSQuery` 56a, `S144` 53e, and `S145` 53f.

Method `queryServerScheduleRemaining`(2) wraps `DBQueryServerScheduleRemaining`(2).

85a ⟨*Read: queryServerScheduleRemaining(2)* 85a⟩≡ (149a)

```
  int[] queryServerScheduleRemaining(final int sid, final int t) throws SQLException {
    int[] output = this.storage.DBQueryServerScheduleRemaining(sid, t);
    return output;
  }
```

Defines:
`queryServerScheduleRemaining`, never used.
Uses `DBQueryServerScheduleRemaining` 84c.

## `DBQueryServerLoadMax`(2)

Method `DBQueryServerLoadMax`(2) returns the maximum load for the given server at the given time. The "maximum load" is equal to the load burden $Q(\mathcal{X}, s, t)$ *plus* the sum of the loads of the requests that are dropped off by the server at $t$. In other words it is the number of occupied seats at $t$ before any drop-offs happen. A `SQLException` is thrown in case of database failure.

**Parameters:**

| | |
|---|---|
| Integer `sid` (param. 1): | server identifier. |
| Integer `t` (param. 2): | a time. |

**Returns:** results of the query flattened into an integer array, or `null` if no results.

| |
|---|
| 0 : maximum load on the server |

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

85b ⟨*Read: DBQueryServerLoadMax(2)* 85b⟩≡ (131c)

```
  int[] DBQueryServerLoadMax(final int sid, final int t) throws SQLException {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S73", 1, sid, t);
    } catch (SQLException e) {
      throw e;
    }
  }
```

Defines:
`DBQueryServerLoadMax`, used in chunk 86a.
Uses `PSQuery` 56a and `S73` 50d.

Method queryServerLoadMax(2) wraps DBQueryServerLoadMax(2).

86a ⟨*Read: queryServerLoadMax(2)* 86a⟩≡ (149a)

```
int[] queryServerLoadMax(final int sid, final int t) throws SQLException {
  int[] output = this.storage.DBQueryServerLoadMax(sid, t);
  return output;
}
```

Defines:
  `queryServerLoadMax`, never used.
Uses `DBQueryServerLoadMax` 85b.

## `DBQueryServerCapacityViolations`(4)

Method `DBQueryServerCapacityViolations`(4) returns the number of schedule events in excess of server capacity within a given time range if a new load were applied during that range. A `SQLException` is thrown in case of database failure.

**Parameters:**

| | |
|---|---|
| Integer `sid` (param. 1): | server identifier. |
| Integer `rq` (param. 2): | additional load to apply. |
| Integer `tp` (param. 3): | start time. |
| Integer `td` (param. 4): | end time. |

**Returns:** results of the query flattened into an integer array, or `null` if no results.

| 0 : *count* |
|---|

where *count* is the number of events in exceeding server capacity.
**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

86b ⟨*Read: DBQueryServerCapacityViolations(4)* 86b⟩≡ (131c)

```
int[] DBQueryServerCapacityViolations(final int sid,
    final int rq, final int tp, final int td) throws SQLException {
  try (⟨Open conn 35c⟩) {
    return PSQuery(conn, "S163", 1, sid, rq, td, td, tp, td);
  } catch (SQLException e) {
    throw e;
  }
}
```

Defines:
  `DBQueryServerCapacityViolations`, used in chunk 86c.
Uses `PSQuery` 56a and `S163` 54i.

86c ⟨*Read: queryServerCapacityViolations(4)* 86c⟩≡ (149a)

```
int[] queryServerCapacityViolations(final int sid,
    final int rq, final int tp, final int td) throws SQLException {
  return this.storage.DBQueryServerCapacityViolations(sid, rq, tp, td);
}
```

Defines:
  `queryServerCapacityViolations`, never used.
Uses `DBQueryServerCapacityViolations` 86b.

## DBQueryServerDistance(2)

Method DBQueryServerDistance(2) returns the travel distance $D(w)$ of the given server. A SQLException is thrown in case of database failure.

**Parameters:**

| | |
|---|---|
| Integer sid (param. 1): | server identifier. |
| Boolean flag_usecache (param. 1): | false to force retrieval from database. |

**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : D(w)}$

where $w$ is the route of the given server identified by sid (param. 1).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

87a ⟨*Read: DBQueryServerDistance(2)* 87a⟩≡ (131c)
```
  int[] DBQueryServerDistance(final int sid, boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      return new int[] { this.distance_servers.get(sid) };
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S104", 1, sid);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```
Defines:
DBQueryServerDistance, used in chunks 87b, 95b, and 101a.
Uses PSQuery 56a and S104 50j.

87b ⟨*Read: queryServerDistance(2)* 87b⟩≡ (140c)
```
  int[] queryServerDistance(final int sid, boolean flag_usecache) throws SQLException {
    return this.storage.DBQueryServerDistance(sid, flag_usecache);
  }
```
Defines:
queryServerDistance, never used.
Uses DBQueryServerDistance 87a.

## DBQueryServerDistanceRemaining(2)

Method DBQueryServerDistanceRemaining(2) returns the remaining distance $D(w_{>t})$ for the given server at the given time. A SQLException is thrown in case of database failure.

**Parameters:**

| | |
|---|---|
| Integer sid (param. 1): | server identifier. |
| Integer t (param. 2): | a time. |

**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : D(w_{>t})}$

where $w_{>t}$ is the remaining route for the given server identified by sid (param. 1).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

87c ⟨*Read: DBQueryServerDistanceRemaining(2)* 87c⟩≡ (131c)
```
  int[] DBQueryServerDistanceRemaining(final int sid, final int t)
  throws SQLException {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S142", 1, sid, t);
    } catch (SQLException e) {
      throw e;
    }
  }
```

Defines:
`DBQueryServerDistanceRemaining`, used in chunks 36a and 88a.
Uses `PSQuery` 56a and `S142` 53c.

Method `queryServerDistanceRemaining`(2) wraps `DBQueryServerDistanceRemaining`(2).

88a ⟨*Read: queryServerDistanceRemaining(2)* 88a⟩≡ (149a)
```
int[] queryServerDistanceRemaining(final int sid, final int t) throws SQLException {
  int[] output = this.storage.DBQueryServerDistanceRemaining(sid, t);
  return output;
}
```
Defines:
`queryServerDistanceRemaining`, never used.
Uses `DBQueryServerDistanceRemaining` 87c.

### `DBQueryServerDistanceCruising`(2)

Method `DBQueryServerDistanceCruising`(2) returns the cruising distance $D^{\text{cruise}}(\mathcal{X}, s)$ (Eq. 2.12) of the given server. A `SQLException` is thrown in case of database failure.

**Parameters:**
Integer `sid` (param. 1): server identifier.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\boxed{0 : D^{\text{cruise}}(\mathcal{X}, s)}$

where $s$ is the server identified by `sid` (param. 1).
**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

88b ⟨*Read: DBQueryServerDistanceCruising(2)* 88b⟩≡ (131c)
```
int[] DBQueryServerDistanceCruising(final int sid, boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    return new int [] { this.distance_servers_cruising.get(sid) };
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S106", 1, sid);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```
Defines:
`DBQueryServerDistanceCruising`, never used.
Uses `PSQuery` 56a and `S106` 50l.

### `DBQueryServerDistanceService`(2)

Method `DBQueryServerDistanceService`(2) returns the service distance $D^{\text{service}}(\mathcal{X}, s)$ (Eq. 2.13) of the given server. A `SQLException` is thrown in case of database failure.

**Parameters:**
Integer `sid` (param. 1): server identifier.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\boxed{0 : D^{\text{service}}(\mathcal{X}, s)}$

where $s$ is the server identified by `sid` (param. 1).
**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

88c ⟨*Read: DBQueryServerDistanceService(2)* 88c⟩≡ (131c)
```
int[] DBQueryServerDistanceService(final int sid, boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    return new int [] { this.distance_servers.get(sid) - this.distance_servers_cruising.get(sid) };
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S108", 1, sid);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```

Defines:
  `DBQueryServerDistanceService`, never used.
Uses `PSQuery` 56a and `S108` 51b.

## `DBQueryServerDurationRemaining`(2)

Method `DBQueryServerDurationRemaining`(2) returns the remaining duration $\delta(w_{>t})$ for the given server at the given time. A `SQLException` is thrown in case of database failure.

**Parameters:**
Integer `sid` (param. 1): server identifier.
Integer `t` (param. 2): a time.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\boxed{0 : \delta(w_{>t})}$

where $w_{>t}$ is the remaining route for the given server identified by `sid` (param. 1).
**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

89a ⟨*Read: DBQueryServerDurationRemaining(2)* 89a⟩≡ (131c)

```
  int[] DBQueryServerDurationRemaining(final int sid, final int t)
  throws SQLException {
    try (⟨Open conn 35c⟩) {
      int[] output = PSQuery(conn, "S127", 1, sid, t);
      if (output != null) {
        output[0] -= t;
      }
      return output;
    } catch (SQLException e) {
      throw e;
    }
  }
```

Defines:
  `DBQueryServerDurationRemaining`, used in chunk 89b.
Uses `PSQuery` 56a and `S127` 52g.

Method `queryServerDurationRemaining`(2) wraps `DBQueryServerDurationRemaining`(2).

89b ⟨*Read: queryServerDurationRemaining(2)* 89b⟩≡ (149a)

```
  int[] queryServerDurationRemaining(final int sid, final int t) throws SQLException {
    int[] output = this.storage.DBQueryServerDurationRemaining(sid, t);
    return output;
  }
```

Defines:
  `queryServerDurationRemaining`, never used.
Uses `DBQueryServerDurationRemaining` 89a.

### DBQueryServerDurationTravel(2)

Method DBQueryServerDurationTravel(2) returns the travel duration $\delta^{\text{travel}}(\mathcal{X}, r)$ (Eq. 2.23) of the given server. A SQLException is thrown in case of database failure.

**Parameters:**
Integer sid (param. 1): server identifier.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \delta^{\text{travel}}(\mathcal{X}, s)}$

where $s$ is the server identified by sid (param. 1).
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

90a ⟨*Read: DBQueryServerDurationTravel(2)* 90a⟩≡ (131c)
```
int[] DBQueryServerDurationTravel(final int sid, boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    return new int[] { this.duration_servers.containsKey(sid)
      ? this.duration_servers.get(sid)
      : 0 };
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S116", 1, sid);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```
Defines:
DBQueryServerDurationTravel, used in chunk 90b.
Uses PSQuery 56a and S116 51j.

90b ⟨*Read: queryServerDurationTravel(2)* 90b⟩≡ (149a)
```
int[] queryServerDurationTravel(final int sid, boolean flag_usecache) throws SQLException {
  return storage.DBQueryServerDurationTravel(sid, flag_usecache);
}
```
Defines:
queryServerDurationTravel, never used.
Uses DBQueryServerDurationTravel 90a.

### DBQueryServerDurationCruising(2)

90c ⟨*Read: DBQueryServerDurationCruising(2)* 90c⟩≡ (131c)
```
int[] DBQueryServerDurationCruising(final int sid, boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    return new int[] { this.duration_servers_cruising.get(sid) };
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S158", 1, sid, sid);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```
Defines:
DBQueryServerDurationCruising, never used.
Uses PSQuery 56a and S158 54d.

### DBQueryServerDurationService(2)

90d ⟨*Read: DBQueryServerDurationService(2)* 90d⟩≡ (131c)
```
int[] DBQueryServerDurationService(final int sid, boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    return new int[] { (this.duration_servers.get(sid)
        - this.duration_servers_cruising.get(sid)) };
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S157", 1, sid);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```

Defines:
`DBQueryServerDurationService`, never used.
Uses `PSQuery` 56a and `S157` 54c.

## `DBQueryServerTimeOfDeparture(1)`

Method `DBQueryServerTimeOfDeparture`(1) returns the departure time $t^{\text{depart}}(\mathcal{X}, s)$ (Eq. 2.16) of the given server. A `SQLException` is thrown in case of database failure.

**Parameters:**
Integer `sid` (param. 1): server identifier.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\boxed{0 : t^{\text{depart}}(\mathcal{X}, s)}$

where $s$ is the server identified by `sid` (param. 1).
**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

91a ⟨*Read: DBQueryServerTimeOfDeparture(1)* 91a⟩≡ (131c)
```
  int[] DBQueryServerTimeOfDeparture(final int sid) throws SQLException {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S125", 1, sid);
    } catch (SQLException e) {
      throw e;
    }
  }
```

Defines:
`DBQueryServerTimeOfDeparture`, used in chunk 91b.
Uses `PSQuery` 56a and `S125` 52e.

Method `queryServerTimeOfDeparture`(1) wraps `DBQueryServerTimeOfDeparture`(1).

91b ⟨*Read: queryServerTimeOfDeparture(1)* 91b⟩≡ (140c)
```
  int[] queryServerTimeOfDeparture(final int sid) throws SQLException {
    int[] output = storage.DBQueryServerTimeOfDeparture(sid);
    return output;
  }
```

Defines:
`queryServerTimeOfDeparture`, never used.
Uses `DBQueryServerTimeOfDeparture` 91a.

## `DBQueryServerTimeOfArrival(1)`

Method `DBQueryServerTimeOfArrival`(1) returns the arrival time $t^{\text{arrive}}(\mathcal{X}, s)$ (Eq. 2.17) of the given server. A `SQLException` is thrown in case of database failure.

**Parameters:**
Integer `sid` (param. 1): server identifier.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\boxed{0 : t^{\text{arrive}}(\mathcal{X}, s)}$

where $s$ is the request identified by `sid` (param. 1).
**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

92a ⟨*Read: DBQueryServerTimeOfArrival(1)* 92a⟩≡ (131c)
```
int[] DBQueryServerTimeOfArrival(final int sid) throws SQLException {
  try (⟨Open conn 35c⟩) {
    return PSQuery(conn, "S127", 1, sid);
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
`DBQueryServerTimeOfArrival`, used in chunk 92b.
Uses `PSQuery` 56a and `S127` 52g.

Method `queryServerTimeOfArrival`(1) wraps `DBQueryServerTimeOfArrival`(1).

92b ⟨*Read: queryServerTimeOfArrival(1)* 92b⟩≡
```
int[] queryServerTimeOfArrival(final int sid) throws SQLException {
  int[] output = storage.DBQueryServerTimeOfArrival(sid);
  return output;
}
```
Defines:
`queryServerTimeOfArrival`, never used.
Uses `DBQueryServerTimeOfArrival` 92a.

## `DBQueryServerAssignmentsPending`(2)

Method `DBQueryServerAssignmentsPending`(2) returns the requests that will be picked up by the given server beyond the given time. A `SQLException` is thrown in case of database failure.

**Parameters:**

Integer `sid` (param. 1): server identifier.
Integer `t` (param. 2): a time.

**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\cdots \boxed{i : \text{identifier for request } r_i} \cdots$

where $1 \leq i \leq |R^{\text{pending}}(\mathcal{X}, s, t)|$, $r_i \in R^{\text{pending}}(\mathcal{X}, s, t)$, and $R^{\text{pending}}(\mathcal{X}, s, t) = (R(\mathcal{X}, s, H) \setminus R(\mathcal{X}, s, t))$ for time horizon $H$, server $s$ identified by `sid` (param. 1), and time $t$ given by param. 2.
**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

92c ⟨*Read: DBQueryServerAssignmentsPending(2)* 92c⟩≡ (131c)
```
int[] DBQueryServerAssignmentsPending(final int sid, final int t)
throws SQLException {
  try (⟨Open conn 35c⟩) {
    return PSQuery(conn, "S100", 1, t, sid);
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
`DBQueryServerAssignmentsPending`, never used.
Uses `PSQuery` 56a and `S100` 50f.

DBQueryServerAssignmentsCompleted**(2)**

Method DBQueryServerAssignmentsCompleted(2) returns the requests that have been dropped off by the given server on or before the given time, in other words $R(\mathcal{X}, s, t)$ (Eq. 2.7). A SQLException is thrown in case of database failure.

**Parameters:**
Integer sid (param. 1): server identifier.
Integer t (param. 2): a time.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\cdots$ | $i$ : identifier for request $r_i$ | $\cdots$

where $1 \leq i \leq |R(\mathcal{X}, s, t)|$ and $r_i \in R(\mathcal{X}, s, t)$ for server $s$ identified by sid (param. 1) at time $t$ given by param. 2.
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

93a ⟨*Read: DBQueryServerAssignmentsCompleted(2)* 93a⟩≡ (131c)
```
int[] DBQueryServerAssignmentsCompleted(final int sid, final int t)
throws SQLException {
  try (⟨Open conn 35c⟩) {
    return PSQuery(conn, "S101", 1, t, sid);
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
DBQueryServerAssignmentsCompleted, never used.
Uses PSQuery 56a and S101 50g.

DBQueryServersActive**(1)**

Method DBQueryServersActive(1) returns the identifiers of the active servers at the given time. A server is "active" if its service has not ended. A SQLException is thrown in case of database failure.

**Parameters:**
Integer t (param. 1): a time
**Returns:** results of the query flattened into an integer array, or null if no results.

| 0 : a server identifier | $\cdots$ |

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

93b ⟨*Read: DBQueryServersActive(1)* 93b⟩≡ (131c)
```
int[] DBQueryServersActive(final int t) throws SQLException {
  try (⟨Open conn 35c⟩) {
    return this.PSQuery(conn, "S134", 1, t, t, t);
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
DBQueryServersActive, used in chunk 93c.
Uses PSQuery 56a and S134 52i.

93c ⟨*Read: queryServersActive(1)* 93c⟩≡ (140c)
```
int[] queryServersActive(final int t) throws SQLException {
  int[] output = this.storage.DBQueryServersActive(t);
  return output;
}
```
Defines:
queryServersActive, used in chunk 182.
Uses DBQueryServersActive 93b.

DBQueryServersCount(**0**)

Method DBQueryCountSevers(0) returns the total number of servers in Table S. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

| |
|---|
| 0 : number of servers in Table S |

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

94a ⟨*Read: DBQueryServersCount(0)* 94a⟩≡ (131c)
```
  int[] DBQueryServersCount() throws SQLException {
    try (⟨Open conn 35c⟩) {
      return this.PSQuery(conn, "S66", 1);
    } catch (SQLException e) {
      throw e;
    }
  }
```
Defines:
DBQueryServersCount, used in chunk 94b.
Uses PSQuery 56a and S66 49c.

Method queryServersCount(0) wraps DBQueryServersCount(0).

94b ⟨*Read: queryServersCount(0)* 94b⟩≡ (140c)
```
  int[] queryServersCount() throws SQLException {
    int[] output = storage.DBQueryServersCount();
    return output;
  }
```
Defines:
queryServersCount, used in chunks 193 and 195.
Uses DBQueryServersCount 94a.

DBQueryServersCountActive(**1**)

Method DBQueryServersCountActive(1) returns the identifiers of the active servers at the given time. A server is "active" if its service has not ended. A SQLException is thrown in case of database failure.

**Parameters:**
Integer t (param. 1): a time
**Returns:** results of the query flattened into an integer array, or null if no results.

| | |
|---|---|
| 0 : a server identifier | ··· |

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

94c ⟨*Read: DBQueryServersCountActive(1)* 94c⟩≡ (131c)
```
  int[] DBQueryServersCountActive(final int t) throws SQLException {
    try (⟨Open conn 35c⟩) {
      return new int[] { this.PSQuery(conn, "S134", 1, t, t, t).length/2 };
    } catch (SQLException e) {
      throw e;
    }
  }
```
Defines:
DBQueryServersCountActive, used in chunk 94d.
Uses PSQuery 56a and S134 52i.

94d ⟨*Read: queryServersCountActive(1)* 94d⟩≡ (140c)
```
  int[] queryServersCountActive(final int t) throws SQLException {
    int[] output = this.storage.DBQueryServersCountActive(t);
    return output;
  }
```

Defines:
  `queryServersCountActive`, used in chunk 190b.
Uses `DBQueryServersCountActive` 94c.

## `DBQueryServersCountAppeared`(0)

TODO. Very bad method name. A server counts as "appeared" only if its route distance is greater than 0. For example, a taxi that is idling and has never moved does not count as "appeared".

95a ⟨*Read: DBQueryServersCountAppeared(0)* 95a⟩≡
```
  int[] DBQueryServersCountAppeared() throws SQLException {
    int[] output = new int[] { };
    // ...
    return output;
  }
```
Defines:
  `DBQueryServersCountAppeared`, never used.

95b ⟨*Read: queryServersCountAppeared(0)* 95b⟩≡ (140c)
```
  int[] queryServersCountAppeared() throws SQLException {
    int[] output = new int[] { 0 };
    for (int sid : this.lu_sseen.keySet()) {
      if (this.storage.DBQueryServerDistance(sid, true)[0] > 0) {
        output[0]++;
      }
    }
    return output;
  }
```
Defines:
  `queryServersCountAppeared`, used in chunk 183.
Uses `DBQueryServerDistance` 87a.

## `DBQueryServersLocations`(1)

Method `DBQueryServersLocations`(1) returns the last-known locations of all servers (including inactive servers) at the given time. The "last-known location" is the waypoint in the server's route $w$ with a time component closest to but not exceeding the given time, in other words $w_{\leq t|w_{\leq t}|}$. A `SQLException` is thrown in case of database failure.

**Parameters:**
  Integer `t` (param. 1): a time
**Returns:** results of the query flattened into an integer array, or `null` if no results.

| 0 : a server identifier | 1 : time of last-known location | 2 : vertex of last-known location | ⋯ |
|---|---|---|---|

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

95c ⟨*Read: DBQueryServersLocations(1)* 95c⟩≡ (131c)
```
  int[] DBQueryServersLocations(final int t) throws SQLException {
    try (⟨Open conn 35c⟩) {
      return this.PSQuery(conn, "S59", 3, t, t, t, t);
    } catch (SQLException e) {
      throw e;
    }
  }
```
Defines:
  `DBQueryServersLocations`, never used.
Uses `PSQuery` 56a and `S59` 49g.

DBQueryServersLocationsActive(1)

SINGLE-THREAD ONLY. Method DBQueryServersLocationsActive(1) returns the last-known locations of all active servers at the given time. A server is "active" if its service has not ended, in other words it has not arrived at its own destination. The "last-known location" is the waypoint in the server's route $w$ with a time component closest to but not exceeding the given time, in other words $w_{\le t|w_{\le t}|}$. A SQLException is thrown in case of database failure.

**Parameters:**
Integer t (param. 1): a time
**Returns:** results of the query flattened into an integer array, or null if no results.

| ⋯ | $3(i-1)$ : identifier for server $s_i$ | $3(i-1)+1$ : $\pi_t$ of last-known location of $s_i$ | $3(i-1)+2$ : $\pi_v$ of last-known location of $s_i$ | ⋯ |
|---|---|---|---|---|

where $1 \le i \le |\mathcal{S}^{\text{active}}|$, $s_i \in \mathcal{S}^{\text{active}}$, and $\mathcal{S}^{\text{active}} = \{s \in \mathcal{S} \mid t^{\text{arrive}}(\mathcal{X}, s) > t\}|$ for $t$ given by param. 1.
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

96a ⟨*Read: DBQueryServersLocationsActive(1)* 96a⟩≡ (131c) 96b ▷
```
  int[] DBQueryServersLocationsActive(final int t) throws SQLException {
    int[] output = new int[] { };
    try (⟨Open conn 35c⟩) {
      int j = 0;
```
Defines:
DBQueryServersLocationsActive, used in chunks 97a and 139.

Our approach is to first use statement S134 to get the active servers. Then for each active server, we use either statement S135 or S147 to get its last-known location.

96b ⟨*Read: DBQueryServersLocationsActive(1)* 96a⟩+≡ (131c) ◁ 96a
```
      // Query S134 selects from CW. The query time is not expected to grow
      // because Table CW does not grow as we pre-load all the servers when we
      // load the problem instance.
      final int[] temp1 = this.PSQuery(conn, "S134", 2, t, t, t);  // <-- 10 ms/call
      output = new int[(3*(temp1.length/2))];
      for (int i = 0; i < temp1.length - 1; i += 2) {
        final int sid = temp1[(i + 0)];
        final int  te = temp1[(i + 1)];
        // Query S135 selects from W. The query time is expected to grow
        // O(log(|W|)) because we have indexes on the relevant columns,
        // implemented in Derby as B+trees (https://db.apache.org/derby/papers/btree_package.html).
        // The subquery in S135 is a range query with a tight range.
        // Query S147 is a key-lookup and also grows O(log(|W|)).
        final int lvt = this.lu_lvt.get(sid);
        final int[] temp2 = (t < te
          ? this.PSQuery(conn, "S135", 2, sid, sid, lvt, t, t)
          : this.PSQuery(conn, "S147", 2, sid, sid));
        output[(j + 0)] = sid;
        if (temp2.length == 0) {
          // Means server hasn't left origin yet, we just get se, so
          int[] temp3 = DBQueryUser(sid);
          output[(j + 1)] = temp3[2];
          output[(j + 2)] = temp3[4];
          this.lu_lvt.put(sid, t);
        } else {
          output[(j + 1)] = temp2[0];
          output[(j + 2)] = temp2[1];
          this.lu_lvt.put(sid, temp2[0]);
        }
        j += 3;
      }
    } catch (SQLException e) {
      throw e;
    } catch (UserNotFoundException e) {
      // Should never happen
      System.err.println("Fatal error: "+e.toString());
      System.exit(1);
    }
    return output;
  }
```

Uses DBQueryUser 73c, PSQuery 56a, query 68b, S134 52i, S135 52j, S147 53g, and UserNotFoundException 65a.

Method querySreversLocationsActive(1) wraps DBQueryServersLocationsActive(1).

97a ⟨*Read: queryServersLocationsActive(1)* 97a⟩≡ (140c 149a)

```
  int[] queryServersLocationsActive(final int t) throws SQLException {
    int[] output = this.storage.DBQueryServersLocationsActive(t);
    return output;
  }
```

Defines:
  queryServersLocationsActive, never used.
Uses DBQueryServersLocationsActive 96a.

### 3.3.5 Methods: Read Metrics

DBQueryMetricServiceRate(1)

Method DBQueryMetricServiceRate(1) returns the service rate $\mu$ (Eq. 2.8). A SQLException is thrown in case of database failure.

**Parameters:**
 Boolean flag_usecache (param. 1): false to force retrieval from database.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \mu \times 10^4}$

Note that the service rate is **multiplied by** $10^4$ so that it can be returned as an integer with 2 decimal points precision, for example if $\mu = .1234$, then DBQueryMetricServiceRate(1) returns 1234.
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

97b ⟨*Read: DBQueryMetricServiceRate(1)* 97b⟩≡ (131c)

```
  int[] DBQueryMetricServiceRate(boolean flag_usecache) throws SQLException {
    int[] output = new int[] { };
    if (flag_usecache) {
      output = new int[] { (int) (10000*(this.count_assigned
          / (double) this.count_requests)) };
    } else {
      try (⟨Open conn 35c⟩) {
        output = PSQuery(conn, "S102", 1);
      } catch (SQLException e) {
        throw e;
      }
    }
    return output;
  }
```

Defines:
  DBQueryMetricServiceRate, used in chunks 97c and 98a.
Uses PSQuery 56a and S102 50h.

Method DBQueryMetricServiceRate(0) calls DBQueryMetricServiceRate(1) with a default parameter.

97c ⟨*Read: DBQueryMetricServiceRate(0)* 97c⟩≡ (132)

```
  int[] DBQueryMetricServiceRate() throws SQLException {
    return DBQueryMetricServiceRate(true);
  }
```

Uses DBQueryMetricServiceRate 97b.

Method queryMetricServiceRate(1) wraps DBQueryMetricServiceRate(1).

98a ⟨*Read: queryMetricServiceRate(1)* 98a⟩≡ (140c)
```
int[] queryMetricServiceRate(boolean flag_usecache) throws SQLException {
  int[] output = storage.DBQueryMetricServiceRate(flag_usecache);
  return output;
}
```
Defines:
`queryMetricServiceRate`, used in chunk 98b.
Uses `DBQueryMetricServiceRate` 97b.

Method queryMetricServiceRate(0) calls queryMetricServiceRate(1) with a default parameter.

98b ⟨*Read: queryMetricServiceRate(0)* 98b⟩≡ (141a)
```
int[] queryMetricServiceRate() throws SQLException {
  return queryMetricServiceRate(true);
}
```
Uses `queryMetricServiceRate` 98a.

### DBQueryMetricServiceRateRunning(0)

98c ⟨*Read: DBQueryMetricServiceRateRunning(0)* 98c⟩≡
```
int[] DBQueryMetricServiceRateRunning() throws SQLException {
  int[] output = new int[] { };
  // ...
  return output;
}
```
Defines:
`DBQueryMetricServiceRateRunning`, never used.

98d ⟨*Read: queryMetricServiceRateRunning(0)* 98d⟩≡ (140c)
```
int[] queryMetricServiceRateRunning() throws SQLException {
  int[] output = new int[] {
      Math.min((int) (10000*(this.storage.DBQueryRequestsCountAssigned()[0]
        / (double) this.lu_rseen.size())), 10000) };
  return output;
}
```
Defines:
`queryMetricServiceRateRunning`, used in chunk 187a.
Uses `DBQueryRequestsCountAssigned` 80d.

### DBQueryMetricUserDistanceBaseTotal(1)

Method DBQueryMetricUserDistanceBaseTotal(1) returns the base distance $D^{\text{base}}(\mathcal{U})$ (Eq. 2.9). A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

| |
|---|
| $0 : D^{\text{base}}(\mathcal{U})$ |

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

98e ⟨*Read: DBQueryMetricUserDistanceBaseTotal(1)* 98e⟩≡ (131c)
```
int[] DBQueryMetricUserDistanceBaseTotal(boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    return new int[] { this.sum_distance_base_requests + this.sum_distance_base_servers };
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S103", 1);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```

Defines:
  DBQueryMetricUserDistanceBaseTotal, used in chunk 99.
Uses PSQuery 56a and S103 50i.

Method DBQueryMetricUserDistanceBaseTotal(0) calls DBQueryMetricUserDistanceBaseTotal(1) with a default parameter.

99a ⟨*Read: DBQueryMetricUserDistanceBaseTotal(0)* 99a⟩≡ (132)
```
  int[] DBQueryMetricUserDistanceBaseTotal() throws SQLException {
    return DBQueryMetricUserDistanceBaseTotal(true);
  }
```
Uses DBQueryMetricUserDistanceBaseTotal 98e.

Method queryMetricUserDistanceBaseTotal(1) wraps DBQueryMetricUserDistanceBaseTotal(1).

99b ⟨*Read: queryMetricUserDistanceBaseTotal(1)* 99b⟩≡ (140c)
```
  int[] queryMetricUserDistanceBaseTotal(boolean flag_usecache) throws SQLException {
    int[] output = storage.DBQueryMetricUserDistanceBaseTotal(flag_usecache);
    return output;
  }
```
Defines:
  queryMetricUserDistanceBaseTotal, used in chunk 99c.
Uses DBQueryMetricUserDistanceBaseTotal 98e.

Method queryMetricUserDistanceBaseTotal(0) calls queryMetricUserDistanceBaseTotal(1) with a default parameter.

99c ⟨*Read: queryMetricUserDistanceBaseTotal(0)* 99c⟩≡ (141a)
```
  int[] queryMetricUserDistanceBaseTotal() throws SQLException {
    return queryMetricUserDistanceBaseTotal(true);
  }
```
Uses queryMetricUserDistanceBaseTotal 99b.

### DBQueryMetricUserDistanceBaseRunning(0)

99d ⟨*Read: DBQueryMetricUserDistanceBaseRunning(0)* 99d⟩≡ (131c)
```
  int[] DBQueryMetricUserDistanceBaseRunning() throws SQLException {
    int[] output = new int[] { };
    // ...
    return output;
  }
```
Defines:
  DBQueryMetricUserDistanceBaseRunning, never used.

99e ⟨*Read: queryMetricUserDistanceBaseRunning(0)* 99e⟩≡ (140c)
```
  int[] queryMetricUserDistanceBaseRunning()
  throws SQLException, UserNotFoundException {
    int[] output = new int[] { 0 };
    for (int sid : this.lu_sseen.keySet()) {
      output[0] += this.storage.DBQueryUser(sid)[6];
    }
    for (int rid : this.lu_rseen.keySet()) {
      output[0] += this.storage.DBQueryUser(rid)[6];
    }
    return output;
  }
```
Defines:
  queryMetricUserDistanceBaseRunning, used in chunk 187b.
Uses DBQueryUser 73c and UserNotFoundException 65a.

### DBQueryMetricServerDistanceTotal(1)

Method DBQueryMetricServerDistanceTotal(1) returns the total travel distance of all the servers. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \sum_{s \in \mathcal{S}} D(W(\mathcal{X}, s))}$

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

100a ⟨*Read: DBQueryMetricServerDistanceTotal(1)* 100a⟩≡ (131c)
```
  int[] DBQueryMetricServerDistanceTotal(boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      final int[] output = new int[] { 0 };
      this.distance_servers.forEach((sid, val) -> output[0] += val);
      return output;
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S105", 1);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```
Defines:
 DBQueryMetricServerDistanceTotal, used in chunk 100.
Uses PSQuery 56a and S105 50k.

Method DBQueryMetricServerDistanceTotal(0) calls DBQueryMetricServerDistanceTotal(1) with a default parameter.

100b ⟨*Read: DBQueryMetricServerDistanceTotal(0)* 100b⟩≡ (132)
```
  int[] DBQueryMetricServerDistanceTotal() throws SQLException {
    return DBQueryMetricServerDistanceTotal(true);
  }
```
Uses DBQueryMetricServerDistanceTotal 100a.

Method queryMetricServerDistanceTotal(1) wraps DBQueryMetricServerDistanceTotal(1).

100c ⟨*Read: queryMetricServerDistanceTotal(1)* 100c⟩≡ (140c)
```
  int[] queryMetricServerDistanceTotal(boolean flag_usecache) throws SQLException {
    int[] output = storage.DBQueryMetricServerDistanceTotal(flag_usecache);
    return output;
  }
```
Defines:
 queryMetricServerDistanceTotal, used in chunks 100d and 188a.
Uses DBQueryMetricServerDistanceTotal 100a.

Method queryMetricServerDistanceTotal(0) calls queryMetricServerDistanceTotal(1) with a default parameter.

100d ⟨*Read: queryMetricServerDistanceTotal(0)* 100d⟩≡ (141a)
```
  int[] queryMetricServerDistanceTotal() throws SQLException {
    return queryMetricServerDistanceTotal(true);
  }
```
Uses queryMetricServerDistanceTotal 100c.

### DBQueryMetricServerDistanceRunning(0)

100e ⟨*Read: DBQueryMetricServerDistanceRunning(0)* 100e⟩≡ (131c)
```
  int[] DBQueryMetricServerDistanceRunning() throws SQLException {
    int[] output = new int[] { };
    // ...
    return output;
  }
```

Defines:
`DBQueryMetricServerDistanceRunning`, never used.

101a ⟨*Read: queryMetricServerDistanceRunning(0)* 101a⟩≡ (140c)

```
  int[] queryMetricServerDistanceRunning() throws SQLException {
    int[] output = new int[] { 0 };
    for (int sid : this.lu_sseen.keySet()) {
      output[0] += this.storage.DBQueryServerDistance(sid, true)[0];
    }
    return output;
  }
```

Defines:
`queryMetricServerDistanceRunning`, used in chunk 187b.
Uses `DBQueryServerDistance` 87a.

### `DBQueryMetricServerDistanceBaseTotal`(0)

Method `DBQueryMetricServerDistanceBaseTotal`(0) returns the base distance of all the servers. A `SQLException` is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

| |
|---|
| $0 : \sum_{s \in \mathcal{S}} d_s$ |

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

101b ⟨*Read: DBQueryMetricServerDistanceBaseTotal(0)* 101b⟩≡ (131c)

```
  int[] DBQueryMetricServerDistanceBaseTotal() throws SQLException {
    // try (⟨Open conn 35c⟩) {
    //   return PSQuery(conn, "S110", 1);
    // } catch (SQLException e) {
    //   throw e;
    // }
    return new int[] { this.sum_distance_base_servers };
  }
```

Defines:
`DBQueryMetricServerDistanceBaseTotal`, used in chunk 101c.
Uses `PSQuery` 56a and `S110` 51d.

Method `queryMetricServerDistanceBaseTotal`(0) wraps `DBQueryMetricServerDistanceBaseTotal`(0).

101c ⟨*Read: queryMetricServerDistanceBaseTotal(0)* 101c⟩≡ (140c)

```
  int[] queryMetricServerDistanceBaseTotal() throws SQLException {
    int[] output = storage.DBQueryMetricServerDistanceBaseTotal();
    return output;
  }
```

Defines:
`queryMetricServerDistanceBaseTotal`, never used.
Uses `DBQueryMetricServerDistanceBaseTotal` 101b.

### `DBQueryMetricServerDistanceCruisingTotal`(1)

Method `DBQueryMetricServerDistanceCruisingTotal`(1) returns the total cruising distance of all servers. A `SQLException` is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

| |
|---|
| $0 : \sum_{s \in \mathcal{S}} D^{\text{cruise}}(\mathcal{X}, s)$ |

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

102a ⟨*Read: DBQueryMetricServerDistanceCruisingTotal(1)* 102a⟩≡ (131c)
```
  int[] DBQueryMetricServerDistanceCruisingTotal(boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      final int[] output = new int[] { 0 };
      this.distance_servers_cruising.forEach((sid, val) -> output[0] += val);
      return output;
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S107", 1);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```
Defines:
`DBQueryMetricServerDistanceCruisingTotal`, used in chunk 102.
Uses `PSQuery` 56a and `S107` 51a.

Method `DBQueryMetricServerDistanceCruisingTotal`(0) calls `DBQueryMetricServerDistanceCruisingTotal`(1) with a default parameter.

102b ⟨*Read: DBQueryMetricServerDistanceCruisingTotal(0)* 102b⟩≡ (132)
```
  int[] DBQueryMetricServerDistanceCruisingTotal() throws SQLException {
    return DBQueryMetricServerDistanceCruisingTotal(true);
  }
```
Uses `DBQueryMetricServerDistanceCruisingTotal` 102a.

Method `queryMetricServerDistanceCruisingTotal`(1) wraps `DBQueryMetricServerDistanceCruisingTotal`(1).

102c ⟨*Read: queryMetricServerDistanceCruisingTotal(1)* 102c⟩≡ (140c)
```
  int[] queryMetricServerDistanceCruisingTotal(boolean flag_usecache) throws SQLException {
    int[] output = storage.DBQueryMetricServerDistanceCruisingTotal(flag_usecache);
    return output;
  }
```
Defines:
`queryMetricServerDistanceCruisingTotal`, used in chunks 102d and 188c.
Uses `DBQueryMetricServerDistanceCruisingTotal` 102a.

Method `queryMetricServerDistanceCruisingTotal`(0) calls `queryMetricServerDistanceCruisingTotal`(1) with a default parameter.

102d ⟨*Read: queryMetricServerDistanceCruisingTotal(0)* 102d⟩≡ (141a)
```
  int[] queryMetricServerDistanceCruisingTotal() throws SQLException {
    return queryMetricServerDistanceCruisingTotal(true);
  }
```
Uses `queryMetricServerDistanceCruisingTotal` 102c.

### `DBQueryMetricServerDistanceServiceTotal`(1)

Method `DBQueryMetricServerDistanceServiceTotal`(1) returns the total service distance of all servers. A `SQLException` is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\boxed{0 : \sum_{s \in \mathcal{S}} D^{\text{service}}(\mathcal{X}, s)}$

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

102e ⟨*Read: DBQueryMetricServerDistanceServiceTotal(1)* 102e⟩≡ (131c)
```
  int[] DBQueryMetricServerDistanceServiceTotal(boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      final int[] output = new int[] { 0 };
      this.distance_servers.forEach((sid, val) -> output[0] += (val - this.distance_servers_cruising.get(sid)))
      return output;
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S109", 1);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```

Defines:
  DBQueryMetricServerDistanceServiceTotal, used in chunk 103.
Uses PSQuery 56a and S109 51c.

Method DBQueryMetricServerDistanceServiceTotal(0) calls DBQueryMetricServerDistanceServiceTotal(1) with a default parameter.

103a ⟨*Read: DBQueryMetricServerDistanceServiceTotal(0)* 103a⟩≡ (132)

```
int[] DBQueryMetricServerDistanceServiceTotal() throws SQLException {
  return DBQueryMetricServerDistanceServiceTotal(true);
}
```

Uses DBQueryMetricServerDistanceServiceTotal 102e.

Method queryMetricServerDistanceServiceTotal(1) wraps DBQueryMetricServerDistanceServiceTotal(1).

103b ⟨*Read: queryMetricServerDistanceServiceTotal(1)* 103b⟩≡ (140c)

```
int[] queryMetricServerDistanceServiceTotal(boolean flag_usecache) throws SQLException {
  int[] output = storage.DBQueryMetricServerDistanceServiceTotal(flag_usecache);
  return output;
}
```

Defines:
  queryMetricServerDistanceServiceTotal, used in chunks 103c and 188b.
Uses DBQueryMetricServerDistanceServiceTotal 102e.

Method queryMetricServerDistanceServiceTotal(0) calls queryMetricServerDistanceServiceTotal(1) with a default parameter.

103c ⟨*Read: queryMetricServerDistanceServiceTotal(0)* 103c⟩≡ (141a)

```
int[] queryMetricServerDistanceServiceTotal() throws SQLException {
  return queryMetricServerDistanceServiceTotal(true);
}
```

Uses queryMetricServerDistanceServiceTotal 103b.

## DBQueryMetricServerDurationTravelTotal(1)

Method DBQueryMetricServerDurationTravelTotal(1) returns the total travel duration of all servers. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \sum_{s \in \mathcal{S}} \delta^{\text{travel}}(\mathcal{X}, s)}$

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

103d ⟨*Read: DBQueryMetricServerDurationTravelTotal(1)* 103d⟩≡ (131c)

```
int[] DBQueryMetricServerDurationTravelTotal(boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    final int[] output = new int[] { 0 };
    this.duration_servers.forEach((sid, val) -> output[0] += val);
    return output;
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S117", 1);
    } catch (SQLException e) {
      throw e;
    }
    }
  }
```

Defines:
  DBQueryMetricServerDurationTravelTotal, used in chunk 104.
Uses PSQuery 56a and S117 51k.

Method DBQueryMetricServerDurationTravelTotal(0) calls DBQueryMetricServerDurationTravelTotal(1) with a default parameter.

104a ⟨*Read: DBQueryMetricServerDurationTravelTotal(0)* 104a⟩≡ (132)

```
  int[] DBQueryMetricServerDurationTravelTotal() throws SQLException {
    return DBQueryMetricServerDurationTravelTotal(true);
  }
```

Uses DBQueryMetricServerDurationTravelTotal 103d.

Method queryMetricServerDurationTravelTotal(1) wraps DBQueryMetricServerDurationTravelTotal(1).

104b ⟨*Read: queryMetricServerDurationTravelTotal(1)* 104b⟩≡ (140c)

```
  int[] queryMetricServerDurationTravelTotal(boolean flag_usecache) throws SQLException {
    int[] output = storage.DBQueryMetricServerDurationTravelTotal(flag_usecache);
    return output;
  }
```

Defines:
  queryMetricServerDurationTravelTotal, used in chunks 104c and 188d.
Uses DBQueryMetricServerDurationTravelTotal 103d.

Method queryMetricServerDurationTravelTotal(0) calls queryMetricServerDurationTravelTotal(1) with a default parameter.

104c ⟨*Read: queryMetricServerDurationTravelTotal(0)* 104c⟩≡ (141a)

```
  int[] queryMetricServerDurationTravelTotal() throws SQLException {
    return queryMetricServerDurationTravelTotal(true);
  }
```

Uses queryMetricServerDurationTravelTotal 104b.

## DBQueryMetricServerDurationCruisingTotal(1)

104d ⟨*Read: DBQueryMetricServerDurationCruisingTotal(1)* 104d⟩≡ (131c)

```
  int[] DBQueryMetricServerDurationCruisingTotal(boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      final int[] output = new int[] { 0 };
      this.duration_servers_cruising.forEach((sid, val) -> output[0] += val);
      return output;
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S160", 1);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```

Defines:
  DBQueryMetricServerDurationCruisingTotal, used in chunks 104e and 105a.
Uses PSQuery 56a and S160 54f.

Method DBQueryMetricServerDurationCruisingTotal(0) calls DBQueryMetricServerDurationCruisingTotal(1) with a default parameter.

104e ⟨*Read: DBQueryMetricServerDurationCruisingTotal(0)* 104e⟩≡ (132)

```
  int[] DBQueryMetricServerDurationCruisingTotal() throws SQLException {
    return DBQueryMetricServerDurationCruisingTotal(true);
  }
```

Uses DBQueryMetricServerDurationCruisingTotal 104d.

105a ⟨*Read: queryMetricServerDurationCruisingTotal(1)* 105a⟩≡ (140c)
```
int[] queryMetricServerDurationCruisingTotal(boolean flag_usecache) throws SQLException {
  int[] output = storage.DBQueryMetricServerDurationCruisingTotal(flag_usecache);
  return output;
}
```
Defines:
queryMetricServerDurationCruisingTotal, used in chunks 105b and 188f.
Uses DBQueryMetricServerDurationCruisingTotal 104d.

Method queryMetricServerDurationCruisingTotal(0) calls queryMetricServerDurationCruisingTotal(1) with a default parameter.

105b ⟨*Read: queryMetricServerDurationCruisingTotal(0)* 105b⟩≡ (141a)
```
int[] queryMetricServerDurationCruisingTotal() throws SQLException {
  return queryMetricServerDurationCruisingTotal(true);
}
```
Uses queryMetricServerDurationCruisingTotal 105a.

## DBQueryMetricServerDurationServiceTotal(1)

105c ⟨*Read: DBQueryMetricServerDurationServiceTotal(1)* 105c⟩≡ (131c)
```
int[] DBQueryMetricServerDurationServiceTotal(boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    final int[] output = new int[] { 0 };
    this.duration_servers.forEach((sid, val) ->
      output[0] += (val - this.duration_servers_cruising.get(sid))
    );
    return output;
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S158", 1);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```
Defines:
DBQueryMetricServerDurationServiceTotal, used in chunk 105.
Uses PSQuery 56a and S158 54d.

Method DBQueryMetricServerDurationServiceTotal(0) calls DBQueryMetricServerDurationServiceTotal(1) with a default parameter.

105d ⟨*Read: DBQueryMetricServerDurationServiceTotal(0)* 105d⟩≡ (132)
```
int[] DBQueryMetricServerDurationServiceTotal() throws SQLException {
  return DBQueryMetricServerDurationServiceTotal(true);
}
```
Uses DBQueryMetricServerDurationServiceTotal 105c.

105e ⟨*Read: queryMetricServerDurationServiceTotal(1)* 105e⟩≡ (140c)
```
int[] queryMetricServerDurationServiceTotal(boolean flag_usecache) throws SQLException {
  int[] output = storage.DBQueryMetricServerDurationServiceTotal(flag_usecache);
  return output;
}
```
Defines:
queryMetricServerDurationServiceTotal, used in chunks 105f and 188e.
Uses DBQueryMetricServerDurationServiceTotal 105c.

Method queryMetricServerDurationServiceTotal(0) calls queryMetricServerDurationServiceTotal(1) with a default parameter.

105f ⟨*Read: queryMetricServerDurationServiceTotal(0)* 105f⟩≡ (141a)
```
int[] queryMetricServerDurationServiceTotal() throws SQLException {
  return queryMetricServerDurationServiceTotal(true);
}
```
Uses queryMetricServerDurationServiceTotal 105e.

### DBQueryMetricServerTWViolationsTotal(0)

106a ⟨*Read: DBQueryMetricServerTWViolationsTotal(0)* 106a⟩≡ (131c)
```
int[] DBQueryMetricServerTWViolationsTotal() throws SQLException {
  try (⟨Open conn 35c⟩) {
    return PSQuery(conn, "S150", 1);
  } catch (SQLException e) {
    throw e;
  }
}
```
Defines:
  `DBQueryMetricServerTWViolationsTotal`, used in chunk 106b.
Uses `PSQuery` 56a and `S150` 53j.

106b ⟨*Read: queryMetricServerTWViolationsTotal(0)* 106b⟩≡ (140c)
```
int[] queryMetricServerTWViolationsTotal() throws SQLException {
  int[] output = storage.DBQueryMetricServerTWViolationsTotal();
  return output;
}
```
Defines:
  `queryMetricServerTWViolationsTotal`, used in chunk 190d.
Uses `DBQueryMetricServerTWViolationsTotal` 106a.

### DBQueryMetricRequestDistanceBaseTotal(0)

Method `DBQueryMetricRequestDistanceBaseTotal`(0) returns the base distance of all the requests. A `SQLException` is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\boxed{0 : \sum_{r\in\mathcal{R}} d_r}$

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

106c ⟨*Read: DBQueryMetricRequestDistanceBaseTotal(0)* 106c⟩≡ (131c)
```
int[] DBQueryMetricRequestDistanceBaseTotal() throws SQLException {
  // try (⟨Open conn 35c⟩) {
  //   return PSQuery(conn, "S111", 1);
  // } catch (SQLException e) {
  //   throw e;
  // }
  return new int[] { this.sum_distance_base_requests };
}
```
Defines:
  `DBQueryMetricRequestDistanceBaseTotal`, used in chunk 106d.
Uses `PSQuery` 56a and `S111` 51e.

Method `queryMetricRequestDistanceBaseTotal`(0) wraps `DBQueryMetricRequestDistanceBaseTotal`(0).

106d ⟨*Read: queryMetricRequestDistanceBaseTotal(0)* 106d⟩≡ (140c)
```
int[] queryMetricRequestDistanceBaseTotal() throws SQLException {
  int[] output = storage.DBQueryMetricRequestDistanceBaseTotal();
  return output;
}
```
Defines:
  `queryMetricRequestDistanceBaseTotal`, never used.
Uses `DBQueryMetricRequestDistanceBaseTotal` 106c.

## DBQueryMetricRequestDistanceBaseUnassignedTotal(1)

Method DBQueryMetricRequestDistanceBaseUnassignedTotal(1) returns the base distance of all the unassigned requests. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \sum_{r \in R^{ko}(\mathcal{X},H)} d_r}$

where $H$ is the time horizon.
**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

107a ⟨*Read: DBQueryMetricRequestDistanceBaseUnassignedTotal(1)* 107a⟩≡ (131c)
```
  int[] DBQueryMetricRequestDistanceBaseUnassignedTotal(boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      return new int[] { this.sum_distance_unassigned };
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S138", 1);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```
Defines:
  DBQueryMetricRequestDistanceBaseUnassignedTotal, used in chunk 107.
Uses PSQuery 56a and S138 52m.

Method DBQueryMetricRequestDistanceBaseUnassignedTotal(0) calls DBQueryMetricRequestDistanceBaseUnassignedTotal(1) with a default parameter.

107b ⟨*Read: DBQueryMetricRequestDistanceBaseUnassignedTotal(0)* 107b⟩≡ (132)
```
  int[] DBQueryMetricRequestDistanceBaseUnassignedTotal() throws SQLException {
    return DBQueryMetricRequestDistanceBaseUnassignedTotal(true);
  }
```
Uses DBQueryMetricRequestDistanceBaseUnassignedTotal 107a.

Method queryMetricRequestDistanceBaseUnassignedTotal(1) wraps DBQueryMetricRequestDistanceBaseUnassignedTotal(1).

107c ⟨*Read: queryMetricRequestDistanceBaseUnassignedTotal(1)* 107c⟩≡ (140c)
```
  int[] queryMetricRequestDistanceBaseUnassignedTotal(boolean flag_usecache) throws SQLException {
    int[] output = storage.DBQueryMetricRequestDistanceBaseUnassignedTotal(flag_usecache);
    return output;
  }
```
Defines:
  queryMetricRequestDistanceBaseUnassignedTotal, used in chunks 107d and 188g.
Uses DBQueryMetricRequestDistanceBaseUnassignedTotal 107a.

Method queryMetricRequestDistanceBaseUnassignedTotal(0) calls queryMetricRequestDistanceBaseUnassignedTotal(1) with a default parameter.

107d ⟨*Read: queryMetricRequestDistanceBaseUnassignedTotal(0)* 107d⟩≡ (141a)
```
  int[] queryMetricRequestDistanceBaseUnassignedTotal() throws SQLException {
    return queryMetricRequestDistanceBaseUnassignedTotal(true);
  }
```
Uses queryMetricRequestDistanceBaseUnassignedTotal 107c.

## DBQueryMetricRequestDistanceBaseUnassignedRunning(0)

107e ⟨*Read: DBQueryMetricRequestDistanceBaseUnassignedRunning(0)* 107e⟩≡ (131c)
```
  int[] DBQueryMetricRequestDistanceBaseUnassignedRunning() throws SQLException {
    int[] output = new int[] { };
    // ...
    return output;
  }
```

Defines:
`DBQueryMetricRequestDistanceBaseUnassignedRunning`, never used.

108a ⟨*Read: queryMetricRequestDistanceBaseUnassignedRunning(0)* 108a⟩≡ (140c)

```
  int[] queryMetricRequestDistanceBaseUnassignedRunning()
  throws SQLException, UserNotFoundException {
    int[] output = new int[] { 0 };
    for (int rid : this.lu_rseen.keySet()) {
      if (this.storage.DBQueryRequestIsAssigned(rid, true).length == 0) {
        output[0] += this.storage.DBQueryUser(rid)[6];
      }
    }
    return output;
  }
```

Uses `DBQueryRequestIsAssigned` 75a, `DBQueryUser` 73c, and `UserNotFoundException` 65a.

## `DBQueryMetricRequestDistanceDetourTotal`(1)

Method `DBQueryMetricRequestDistanceDetourTotal`(1) returns the total detour distance of all requests. A `SQLException` is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\boxed{0 : \sum_{r\in\mathcal{R}} D^{\text{detour}}(\mathcal{X}, r)}$

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

108b ⟨*Read: DBQueryMetricRequestDistanceDetourTotal(1)* 108b⟩≡ (131c)

```
  int[] DBQueryMetricRequestDistanceDetourTotal(boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      final int[] output = new int[] { 0 };
      this.distance_requests_transit.forEach((rid, val) ->
        output[0] += (val - this.lu_users.get(rid)[6])
      );
      return output;
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S113", 1);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```

Defines:
`DBQueryMetricRequestDistanceDetourTotal`, used in chunk 108.
Uses `PSQuery` 56a and `S113` 51g.

Method `DBQueryMetricRequestDistanceDetourTotal`(0) calls `DBQueryMetricRequestDistanceDetourTotal`(1) with a default parameter.

108c ⟨*Read: DBQueryMetricRequestDistanceDetourTotal(0)* 108c⟩≡ (132)

```
  int[] DBQueryMetricRequestDistanceDetourTotal() throws SQLException {
    return DBQueryMetricRequestDistanceDetourTotal(true);
  }
```

Uses `DBQueryMetricRequestDistanceDetourTotal` 108b.

Method `queryMetricRequestDistanceDetourTotal`(1) wraps `DBQueryMetricRequestDistanceDetourTotal`(1).

108d ⟨*Read: queryMetricRequestDistanceDetourTotal(1)* 108d⟩≡ (140c)

```
  int[] queryMetricRequestDistanceDetourTotal(boolean flag_usecache) throws SQLException {
    int[] output = storage.DBQueryMetricRequestDistanceDetourTotal(flag_usecache);
    return output;
  }
```

Defines:
  queryMetricRequestDistanceDetourTotal, used in chunks 109a and 189b.
Uses DBQueryMetricRequestDistanceDetourTotal 108b.

Method queryMetricRequestDistanceDetourTotal(0) calls queryMetricRequestDistanceDetourTotal(1) with a default parameter.

109a ⟨*Read: queryMetricRequestDistanceDetourTotal(0)* 109a⟩≡ (141a)
```
int[] queryMetricRequestDistanceDetourTotal() throws SQLException {
  return queryMetricRequestDistanceDetourTotal(true);
}
```
Uses queryMetricRequestDistanceDetourTotal 108d.

### DBQueryMetricRequestDistanceTransitTotal(1)

Method DBQueryMetricRequestDistanceTransitTotal(1) returns the total transit distance of all requests. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \sum_{r\in\mathcal{R}} D^{\text{transit}}(\mathcal{X}, r)}$

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

109b ⟨*Read: DBQueryMetricRequestDistanceTransitTotal(1)* 109b⟩≡ (131c)
```
int[] DBQueryMetricRequestDistanceTransitTotal(boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    final int[] output = new int[] { 0 };
    this.distance_requests_transit.forEach((rid, val) -> output[0] += val);
    return output;
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S115", 1);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```
Defines:
  DBQueryMetricRequestDistanceTransitTotal, used in chunk 109.
Uses PSQuery 56a and S115 51i.

Method DBQueryMetricRequestDistanceTransitTotal(0) calls DBQueryMetricRequestDistanceTransitTotal(1) with a default parameter.

109c ⟨*Read: DBQueryMetricRequestDistanceTransitTotal(0)* 109c⟩≡ (132)
```
int[] DBQueryMetricRequestDistanceTransitTotal() throws SQLException {
  return DBQueryMetricRequestDistanceTransitTotal(true);
}
```
Uses DBQueryMetricRequestDistanceTransitTotal 109b.

Method queryMetricRequestDistanceTransitTotal(1) wraps DBQueryMetricRequestDistanceTransitTotal(1).

109d ⟨*Read: queryMetricRequestDistanceTransitTotal(1)* 109d⟩≡ (140c)
```
int[] queryMetricRequestDistanceTransitTotal(boolean flag_usecache) throws SQLException {
  int[] output = storage.DBQueryMetricRequestDistanceTransitTotal(flag_usecache);
  return output;
}
```
Defines:
  queryMetricRequestDistanceTransitTotal, used in chunks 110a and 189a.
Uses DBQueryMetricRequestDistanceTransitTotal 109b.

Method queryMetricRequestDistanceTransitTotal(0) calls queryMetricRequestDistanceTransitTotal(1) with a default parameter.

110a ⟨*Read: queryMetricRequestDistanceTransitTotal(0)* 110a⟩≡ (141a)
```
  int[] queryMetricRequestDistanceTransitTotal() throws SQLException {
    return queryMetricRequestDistanceTransitTotal(true);
  }
```
Uses queryMetricRequestDistanceTransitTotal 109d.

DBQueryMetricRequestDurationPickupTotal**(1)**

Method DBQueryMetricRequestDurationPickupTotal(1) returns the total pickup delay of all requests. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \sum_{r\in\mathcal{R}} \delta^{\text{pickup}}(\mathcal{X}, r)}$

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

110b ⟨*Read: DBQueryMetricRequestDurationPickupTotal(1)* 110b⟩≡ (131c)
```
  int[] DBQueryMetricRequestDurationPickupTotal(boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      final int[] output = new int[] { 0 };
      this.duration_requests_pickup.forEach((rid, val) -> output[0] += val);
      return output;
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S119", 1);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```
Defines:
  DBQueryMetricRequestDurationPickupTotal, used in chunk 110.
Uses PSQuery 56a and S119 51m.

Method DBQueryMetricRequestDurationPickupTotal(0) calls DBQueryMetricRequestDurationPickupTotal(1) with a default parameter.

110c ⟨*Read: DBQueryMetricRequestDurationPickupTotal(0)* 110c⟩≡ (132)
```
  int[] DBQueryMetricRequestDurationPickupTotal() throws SQLException {
    return DBQueryMetricRequestDurationPickupTotal(true);
  }
```
Uses DBQueryMetricRequestDurationPickupTotal 110b.

Method queryMetricRequestDurationPickupTotal(1) wraps DBQueryMetricRequestDurationPickupTotal(1).

110d ⟨*Read: queryMetricRequestDurationPickupTotal(1)* 110d⟩≡ (140c)
```
  int[] queryMetricRequestDurationPickupTotal(boolean flag_usecache) throws SQLException {
    int[] output = storage.DBQueryMetricRequestDurationPickupTotal(flag_usecache);
    return output;
  }
```
Defines:
  queryMetricRequestDurationPickupTotal, used in chunks 110e and 189f.
Uses DBQueryMetricRequestDurationPickupTotal 110b.

Method queryMetricRequestDurationPickupTotal(0) calls queryMetricRequestDurationPickupTotal(1) with a default parameter.

110e ⟨*Read: queryMetricRequestDurationPickupTotal(0)* 110e⟩≡ (141a)
```
  int[] queryMetricRequestDurationPickupTotal() throws SQLException {
    return queryMetricRequestDurationPickupTotal(true);
  }
```
Uses queryMetricRequestDurationPickupTotal 110d.

`DBQueryMetricRequestDurationTransitTotal`**(1)**

Method `DBQueryMetricRequestDurationTransitTotal`(1) returns the total transit duration of all requests. A `SQLException` is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or `null` if no results.

$\boxed{0 : \sum_{r \in \mathcal{R}} \delta^{\text{transit}}(\mathcal{X}, r)}$

**Side Effects:** none.
**Throws:** `SQLException` if database failure is encountered.

111a ⟨*Read: DBQueryMetricRequestDurationTransitTotal(1)* 111a⟩≡ (131c)
```
int[] DBQueryMetricRequestDurationTransitTotal(boolean flag_usecache) throws SQLException {
  if (flag_usecache) {
    final int[] output = new int[] { 0 };
    this.duration_requests_transit.forEach((rid, val) -> output[0] += val);
    return output;
  } else {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S121", 1);
    } catch (SQLException e) {
      throw e;
    }
  }
}
```
Defines:
`DBQueryMetricRequestDurationTransitTotal`, used in chunk 111.
Uses `PSQuery` 56a and `S121` 52a.

Method `DBQueryMetricRequestDurationTransitTotal`(0) calls `DBQueryMetricRequestDurationTransitTotal`(1) with a default parameter.

111b ⟨*Read: DBQueryMetricRequestDurationTransitTotal(0)* 111b⟩≡ (132)
```
int[] DBQueryMetricRequestDurationTransitTotal() throws SQLException {
  return DBQueryMetricRequestDurationTransitTotal(true);
}
```
Uses `DBQueryMetricRequestDurationTransitTotal` 111a.

Method `queryMetricRequestDurationTransitTotal`(1) wraps `DBQueryMetricRequestDurationTransitTotal`(1).

111c ⟨*Read: queryMetricRequestDurationTransitTotal(1)* 111c⟩≡ (140c)
```
int[] queryMetricRequestDurationTransitTotal(boolean flag_usecache) throws SQLException {
  int[] output = storage.DBQueryMetricRequestDurationTransitTotal(flag_usecache);
  return output;
}
```
Defines:
`queryMetricRequestDurationTransitTotal`, used in chunks 111d and 189c.
Uses `DBQueryMetricRequestDurationTransitTotal` 111a.

Method `queryMetricRequestDurationTransitTotal`(0) calls `queryMetricRequestDurationTransitTotal`(1) with a default parameter.

111d ⟨*Read: queryMetricRequestDurationTransitTotal(0)* 111d⟩≡ (141a)
```
int[] queryMetricRequestDurationTransitTotal() throws SQLException {
  return queryMetricRequestDurationTransitTotal(true);
}
```
Uses `queryMetricRequestDurationTransitTotal` 111c.

## DBQueryMetricRequestDurationTravelTotal(1)

Method DBQueryMetricRequestDurationTravelTotal(1) returns the total travel duration of all requests. A SQLException is thrown in case of database failure.

**Parameters:** none.
**Returns:** results of the query flattened into an integer array, or null if no results.

$\boxed{0 : \sum_{r\in\mathcal{R}} \delta^{\text{travel}}(\mathcal{X}, r)}$

**Side Effects:** none.
**Throws:** SQLException if database failure is encountered.

112a ⟨*Read: DBQueryMetricRequestDurationTravelTotal(1)* 112a⟩≡ (131c)
```
  int[] DBQueryMetricRequestDurationTravelTotal(boolean flag_usecache) throws SQLException {
    if (flag_usecache) {
      final int[] output = new int[] { 0 };
      this.duration_requests_travel.forEach((rid, val) -> output[0] += val);
      return output;
    } else {
      try (⟨Open conn 35c⟩) {
        return PSQuery(conn, "S123", 1);
      } catch (SQLException e) {
        throw e;
      }
    }
  }
```
Defines:
  DBQueryMetricRequestDurationTravelTotal, used in chunk 112.
Uses PSQuery 56a and S123 52c.

Method DBQueryMetricRequestDurationTravelTotal(0) calls DBQueryMetricRequestDurationTravelTotal(1) with a default parameter.

112b ⟨*Read: DBQueryMetricRequestDurationTravelTotal(0)* 112b⟩≡ (132)
```
  int[] DBQueryMetricRequestDurationTravelTotal() throws SQLException {
    return DBQueryMetricRequestDurationTravelTotal(true);
  }
```
Uses DBQueryMetricRequestDurationTravelTotal 112a.

Method queryMetricRequestDurationTravelTotal(1) wraps DBQueryMetricRequestDurationTravelTotal(1).

112c ⟨*Read: queryMetricRequestDurationTravelTotal(1)* 112c⟩≡ (140c)
```
  int[] queryMetricRequestDurationTravelTotal(boolean flag_usecache) throws SQLException {
    int[] output = storage.DBQueryMetricRequestDurationTravelTotal(flag_usecache);
    return output;
  }
```
Defines:
  queryMetricRequestDurationTravelTotal, used in chunks 112d and 189e.
Uses DBQueryMetricRequestDurationTravelTotal 112a.

Method queryMetricRequestDurationTravelTotal(0) calls queryMetricRequestDurationTravelTotal(1) with a default parameter.

112d ⟨*Read: queryMetricRequestDurationTravelTotal(0)* 112d⟩≡ (141a)
```
  int[] queryMetricRequestDurationTravelTotal() throws SQLException {
    return queryMetricRequestDurationTravelTotal(true);
  }
```
Uses queryMetricRequestDurationTravelTotal 112c.

## DBQueryMetricRequestTWViolationsTotal(0)

112e ⟨*Read: DBQueryMetricRequestTWViolationsTotal(0)* 112e⟩≡ (131c)
```
  int[] DBQueryMetricRequestTWViolationsTotal() throws SQLException {
    try (⟨Open conn 35c⟩) {
      return PSQuery(conn, "S151", 1);
    } catch (SQLException e) {
      throw e;
    }
  }
```

Defines:
`DBQueryMetricRequestTWViolationsTotal`, used in chunk 113.
Uses `PSQuery` 56a and `S151` 53k.

113 ⟨*Read: queryMetricRequestTWViolationsTotal(0)* 113⟩≡ (140c)

```
  int[] queryMetricRequestTWViolationsTotal() throws SQLException {
    int[] output = storage.DBQueryMetricRequestTWViolationsTotal();
    return output;
  }
```

Defines:
`queryMetricRequestTWViolationsTotal`, used in chunk 190c.
Uses `DBQueryMetricRequestTWViolationsTotal` 112e.

## 3.4 Write Operations

### 3.4.1 Chunks

#### Apply traffic to route, sched

114 ⟨*Apply traffic to route, sched* 114⟩≡ (125b)
```
int[] mutroute = route.clone();
int[] mutsched = sched.clone();
if (this.traffic != null) {
  for (int k = 0; k < (mutroute.length - 3); k += 4) {
    final int t1 = mutroute[k];
    final int v1 = mutroute[(k + 1)];
    final int t2 = mutroute[(k + 2)];
    final int v2 = mutroute[(k + 3)];
    int[] ddnu = this.storage.DBQueryEdge(v1, v2);
    final int dd = ddnu[0];
    final int nu_old = ddnu[1];
    final int nu_new = Math.max(1,
        (int) Math.round(this.traffic.apply(
            v1, v2, (1000*t1 + this.controller.getClockReferenceMs())
        )*nu_old));
    final int diff = ((dd/(t2 - t1)) > nu_new
        ? ((int) Math.ceil((dd/(float) nu_new + t1))) - t2
        : 0);
    if (diff != 0) {
      for (int p = 0; p < (mutsched.length - 3); p += 4) {
        if (mutsched[p] >= mutroute[(k + 2)]) {
          mutsched[p] += diff;
        }
      }
      for (int q = (k + 2); q < (mutroute.length - 1); q += 2) {
        mutroute[q] += diff;
      }
    }
  }
}
```

Uses apply 156a, DBQueryEdge 71b, and getClockReferenceMs 57c.

### Check time window violation

115a ⟨*Check time window violation* 115a⟩≡ (125b)

```
  for (int k = 0; k < (sched.length - 2); k += 3) {
    final int tl = this.storage.DBQueryUser(sched[(k + 2)])[3];
    if (sched[k] > tl) {
      throw new TimeWindowException("Waypoint time (t="+sched[k]+") "
          +"after late window (t="+tl+", uid="+sched[(k + 2)]+")");
    }
  }
```

Uses DBQueryUser 73c and TimeWindowException 64d.

### Delete from W remaining route

115b ⟨*Delete from W remaining route* 115b⟩≡ (118b)

```
  PreparedStatement pS76 = this.PSCreate(conn, "S76");
  this.PSAdd(pS76, sid, route[0]);
  this.PSSubmit(pS76);
```

Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, and S76 48f.

### Delete from PD, CPD jobs

115c ⟨*Delete from PD, CPD jobs* 115c⟩≡ (124c)

```
  PreparedStatement pS42 = this.PSCreate(conn, "S42");
  PreparedStatement pS43 = this.PSCreate(conn, "S43");
  for (final int r : ridneg) {
    this.PSAdd(pS42, r);
    this.PSAdd(pS43, r);
  }
  this.PSSubmit(pS42, pS43);
```

Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, S42 48g, and S43 48h.

### Delete from CQ remaining schedule

115d ⟨*Delete from CQ remaining schedule* 115d⟩≡ (118c)

```
  PreparedStatement pS80 = this.PSCreate(conn, "S80");
  this.PSAdd(pS80, sid, route[0]);
  this.PSSubmit(pS80);
```

Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, and S80 48i.

## Insert into user tables new user

116a ⟨*Insert into user tables new user* 116a⟩≡ (122b 123c)
```
PreparedStatement pS2 = this.PSCreate(conn, "S2");
PreparedStatement pS3 = this.PSCreate(conn, "S3");
PreparedStatement pS4 = this.PSCreate(conn, "S4");
PreparedStatement pS5 = this.PSCreate(conn, "S5");
PreparedStatement pS6 = this.PSCreate(conn, "S6");
PreparedStatement pS7 = this.PSCreate(conn, "S7");
this.PSAdd(pS2, uid, u[1]);
this.PSAdd(pS3, uid, u[2]);
this.PSAdd(pS4, uid, u[3]);
this.PSAdd(pS5, uid, u[4]);
this.PSAdd(pS6, uid, u[5]);
this.PSAdd(pS7, uid, u[6]);
this.PSSubmit(pS2, pS3, pS4, pS5, pS6, pS7);
```
Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, S2 46c, S3 47a, S4 47b, S5 47c, S6 47d, and S7 47e.

## Insert into R new request

116b ⟨*Insert into R new request* 116b⟩≡ (122b)
```
PreparedStatement pS9 = this.PSCreate(conn, "S9");
this.PSAdd(pS9, uid, u[1], u[2], u[3], u[4], u[5], u[6]);
this.PSSubmit(pS9);
```
Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, and S9 47g.

## Insert into S new server

116c ⟨*Insert into S new server* 116c⟩≡ (123c)
```
PreparedStatement pS8 = this.PSCreate(conn, "S8");
this.PSAdd(pS8, uid, u[1], u[2], u[3], u[4], u[5], u[6]);
this.PSSubmit(pS8);
```
Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, and S8 47f.

## Insert into W new server route

116d ⟨*Insert into W new server route* 116d⟩≡ (123c)
```
⟨Procedure to insert route 118a⟩
pS10 = this.PSCreate(conn, "S10");
this.PSAdd(pS10, uid, se, null, null, route[0], route[1], null, null);
this.PSSubmit(pS10);
```
Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, and S10 47h.

## Insert into W new remaining route

116e ⟨*Insert into W new remaining route* 116e⟩≡ (118b)
```
final int uid = sid;
⟨Procedure to insert route 118a⟩
```

## Insert into CW new server route

116f ⟨*Insert into CW new server route* 116f⟩≡ (123c)
```
PreparedStatement pS11 = this.PSCreate(conn, "S11");
final int te = route[(route.length - 2)];
this.PSAdd(pS11, uid, u[2], u[3], u[4], u[5], u[2], u[4], te, u[5]);
this.PSSubmit(pS11);
```
Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, and S11 47j.

### Insert into CQ new server

117a ⟨*Insert into CQ new server* 117a⟩≡ (123c)
```
  PreparedStatement pS14 = this.PSCreate(conn, "S14");
  this.PSAdd(pS14, uid, u[1], u[2], null, u[2], u[4], null, u[1],
      null, null, null, null, null, 1);
  this.PSSubmit(pS14);
```
Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, and S14 47m.

### Insert into CQ new remaining schedule

117b ⟨*Insert into CQ new remaining schedule* 117b⟩≡ (118c)
```
  PreparedStatement pS14 = PSCreate(conn, "S14");
  for (int j = 0; j < (sched.length - 3); j += 4) {
    final int t2 = sched[(j + 0)];
    final int v2 = sched[(j + 1)];
    final int Lj = sched[(j + 3)];
    if (Lj != 0) {
      // if only origin or only destination is in sched, cache will
      // not contain key Lj.
      if (cache.containsKey(Lj)) {
        final int[] qpd = cache.get(Lj);
        final int q2 = (t2 == qpd[1] ? q1 + qpd[0] : q1 - qpd[0]);
        final int o2 = o1 + 1;
        this.PSAdd(pS14, sid, sq, se, t1, t2, v2, q1, q2, Lj,
              qpd[0], qpd[1], qpd[2], o1, o2);
        t1 = t2;
        q1 = q2;
        o1 = o2;
      } else {
        throw new UserNotFoundException("User "+Lj+" not found in schedule!");
      }
    }
  }
  this.PSSubmit(pS14);
```
Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, S14 47m, and UserNotFoundException 65a.

### Insert into PD, CPD new jobs

117c ⟨*Insert into PD, CPD new jobs* 117c⟩≡ (118c)
```
  PreparedStatement pS12 = this.PSCreate(conn, "S12");
  PreparedStatement pS13 = this.PSCreate(conn, "S13");
  for (final int r : ridpos) {
    final int[] output2 = this.PSQuery(conn, "S51", 5, r);
    final int rq = output2[0];
    final int re = output2[1];
    final int rl = output2[2];
    final int ro = output2[3];
    final int rd = output2[4];
    // if r not in sched, cache and cache2 will not contain key r
    if (cache.containsKey(r) && cache2.containsKey(r)) {
      final int[] qpd = cache.get(r);
      final int[]  pd = cache2.get(r);
      this.PSAdd(pS12, sid, qpd[1], pd[0], r);
      this.PSAdd(pS12, sid, qpd[2], pd[1], r);
      this.PSAdd(pS13, sid, se, route[(route.length - 2)], qpd[1], pd[0], qpd[2], pd[1],
            r, re, rl, ro, rd);
    } else {
      throw new UserNotFoundException("User "+r+" not found in schedule!");
    }
  }
  this.PSSubmit(pS12, pS13);
```
Uses PSAdd 55b, PSCreate 55a, PSQuery 56a, PSSubmit 55c, S12 47k, S13 47l, S51 49e, and UserNotFoundException 65a.

## Procedure to insert route

118a ⟨*Procedure to insert route* 118a⟩≡ (116)
```
  PreparedStatement pS10 = this.PSCreate(conn, "S10");
  for (int i = 0; i < (route.length - 3); i += 2) {
    final int t1 = route[(i + 0)];
    final int v1 = route[(i + 1)];
    final int t2 = route[(i + 2)];
    final int v2 = route[(i + 3)];
    if (!(this.lu_edges.containsKey(v1) && this.lu_edges.get(v1).containsKey(v2))) {
      ⟨Debug: EdgeNotFoundException 127⟩
      throw new EdgeNotFoundException("Edge ("+v1+", "+v2+") not found.");
    }
    final int dd = this.lu_edges.get(v1).get(v2)[0];
    final int nu = this.lu_edges.get(v1).get(v2)[1];
    this.PSAdd(pS10, uid, se, t1, v1, t2, v2, dd, nu);
  }
  this.PSSubmit(pS10);
```
Uses `EdgeNotFoundException` 63d, `PSAdd` 55b, `PSCreate` 55a, `PSSubmit` 55c, and `S10` 47h.

## Procedure to update route

118b ⟨*Procedure to update route* 118b⟩≡ (124c)
```
  ⟨Delete from W remaining route 115b⟩
  ⟨Insert into W new remaining route 116e⟩
  ⟨Update CW, CPD route endpoint 119b⟩
```

## Procedure to update and add to schedule

118c ⟨*Procedure to update and add to schedule* 118c⟩≡ (124c)
```
  ⟨Update PD, CPD arrival and departure times 119c⟩
  ⟨Populate the tp, td cache and vp, vd cache and update CQ 118d⟩
  ⟨Select from CQ latest order number 119a⟩
  ⟨Delete from CQ remaining schedule 115d⟩
  ⟨Insert into CQ new remaining schedule 117b⟩
  ⟨Insert into PD, CPD new jobs 117c⟩
```

## Populate the tp, td cache and vp, vd cache and update CQ

118d ⟨*Populate the tp, td cache and vp, vd cache and update CQ* 118d⟩≡ (118c)
```
  PreparedStatement pS140 = this.PSCreate(conn, "S140");
  for (int j = 0; j < (sched.length - 3); j += 4) {
    final int Lj = sched[(j + 3)];
    if (Lj != 0 && !cache.containsKey(Lj)) {
      final int rq = lu_users.get(Lj)[1];
      boolean flagged = false;
      for (final int r : ridpos) {
        if (Lj == r) {
          flagged = true;
          break;
        }
      }
      if (flagged) {
        final int tp = sched[(j + 0)];
        final int vp = sched[(j + 1)];
        for (int k = (j + 4); k < (sched.length - 3); k += 4) {
          if (Lj == sched[(k + 3)]) {
            final int td = sched[(k + 0)];
            final int vd = sched[(k + 1)];
            cache. put(Lj, new int[] { rq, tp, td });
            cache2.put(Lj, new int[] { vp, vd });
            break;
          }
        }
      } else {
        final int[] output = this.PSQuery(conn, "S86", 2, Lj);
        if (output.length == 0) {
          throw new UserNotFoundException("Request "+Lj+" not in pickups/dropoffs!");
        }
        final int tp = output[0];
        final int td = output[1];
        // Here is first time we've seen Lj in the schedule
        // If tp, td both greater than route[0], it means sched should
        // provide two Lj waypoints. If only one is found, then
        // Lj is "dangling".
        if (tp > route[0] && td > route[0]) {
          boolean dangling = true;
          for (int k = (j + 4); k < (sched.length - 3); k += 4) {
            if (Lj == sched[(k + 3)]) {
              dangling = false;
              break;
            }
          }
          if (dangling) {
            throw new UserNotFoundException("Request "+Lj+" is dangling!");
          }
        }
        this.PSAdd(pS140, tp, td, Lj);
        cache.put(Lj, new int[] { rq, tp, td });
      }
    }
  }
  this.PSSubmit(pS140);
```

Uses PSAdd 55b, PSCreate 55a, PSQuery 56a, PSSubmit 55c, S140 53a, S86 50c, and UserNotFoundException 65a.

### Select from CQ latest order number

119a ⟨*Select from CQ latest order number* 119a⟩≡ (118c)

```
  final int[] output = (route[0] == 0 ? null : this.PSQuery(conn, "S87", 3, sid, route[0]));
  int t1 = (route[0] == 0 ?  0 : output[0]);
  int q1 = (route[0] == 0 ? sq : output[1]);
  int o1 = (route[0] == 0 ?  1 : output[2]);
```

Uses PSQuery 56a and S87 50e.

### Update CW, CPD route endpoint

119b ⟨*Update CW, CPD route endpoint* 119b⟩≡ (118b)

```
  PreparedStatement pS77 = this.PSCreate(conn, "S77");
  PreparedStatement pS139 = this.PSCreate(conn, "S139");
  final int te = sched[(sched.length - 4)];
  final int ve = sched[(sched.length - 3)];
  this.PSAdd(pS77, te, ve, sid);
  this.PSAdd(pS139, te, sid);
  this.PSSubmit(pS77, pS139);
```

Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, S139 52n, and S77 48b.

### Update PD, CPD arrival and departure times

119c ⟨*Update PD, CPD arrival and departure times* 119c⟩≡ (118c)

```
  PreparedStatement pS82 = this.PSCreate(conn, "S82");
  PreparedStatement pS83 = this.PSCreate(conn, "S83");
  PreparedStatement pS84 = this.PSCreate(conn, "S84");
  for (int j = 0; j < (sched.length - 3); j += 4) {
    final int tj = sched[(j + 0)];
    final int vj = sched[(j + 1)];
    final int Lj = sched[(j + 3)];
    if (Lj != 0) {
      this.PSAdd(pS82, tj, vj, Lj);
      this.PSAdd(pS83, tj, vj, Lj);
      this.PSAdd(pS84, tj, vj, Lj);
    }
  }
  this.PSSubmit(pS83, pS82, pS84);
```

Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, S82 48d, S83 48e, and S84 48c.

### 3.4.2 Methods: Write Road Network

DBInsertVertex(3)

Method DBInsertVertex(3) inserts a vertex into Table V and into lu_vertices if all succeeds. If the vertex attemping to be inserted already exists, a DuplicateVertexException is thrown. A SQLException is thrown for other database failures.

**Parameters:**

| | |
|---|---|
| Integer v (param. 1): | vertex identifier. |
| Integer lng (param. 2): | longitude, written to an *integer precision*, *e.g.* for longitude 123.456789, pass 123456789 for $10^6$ precision. **The caller is responsible for remembering the precision.** |
| Integer lat (param. 3): | latitude, written to an *integer precision* as above. |

**Returns:** nothing.
**Side Effects:** inserts a row into Table V, puts an entry into lu_vertices.
**Throws:** DuplicateVertexException if vertex already exists, or SQLException for other database failures.

120a ⟨*Write: DBInsertVertex(3)* 120a⟩≡ (133a) 120b ▷

```
  void DBInsertVertex(final int v, final int lng, final int lat)
  throws DuplicateVertexException, SQLException {
```

Defines:
DBInsertVertex, used in chunk 143f.
Uses DuplicateVertexException 63c.

If only DBInsertVertex(3) is ever used to write vertices into Table V, we can be sure that any vertex appearing in Table V also appears in lu_vertices. To check if the vertex in param. 1 is a duplicate entry, it is sufficient to check lu_vertices.

120b ⟨*Write: DBInsertVertex(3)* 120a⟩+≡ (133a) ◁ 120a 120c ▷

```
    if (this.lu_vertices.containsKey(v)) {
      throw new DuplicateVertexException("Vertex "+v+" already exists.");
    }
```

Uses DuplicateVertexException 63c.

All we do is use statement S0 to submit the insert statement against Table V. By putting conn in the resources of the outer try, we ensure conn gets closed in the end no matter what happens. This pattern will appear in other write methods. If all succeeds, we put the vertex into lu_vertices.

120c ⟨*Write: DBInsertVertex(3)* 120a⟩+≡ (133a) ◁ 120b

```
    try (⟨Open conn 35c⟩) {
      try {
        PreparedStatement pS0 = this.PSCreate(conn, "S0");
        this.PSAdd(pS0, v, lng, lat);
        this.PSSubmit(pS0);
        conn.commit();
      } catch (SQLException e) {
        conn.rollback();
        throw e;
      }
    } catch (SQLException e) {
      throw e;
    }
    this.lu_vertices.put(v, new int[] { lng, lat });
```

```
}
```

Uses PSAdd 55b, PSCreate 55a, PSSubmit 55c, and S0 46a.

### DBInsertEdge(4)

Method DBInsertEdge(4) inserts an edge into Table E and into lu_edges if all succeeds. If the edge attempting to be inserted already exists, a DuplicateEdgeException is thrown. A SQLException is thrown for other database failures.

**Parameters:**

| | |
|---|---|
| Integer v1 (param. 1): | source vertex identifier. |
| Integer v2 (param. 2): | target vertex identifier. |
| Integer dd (param. 3): | distance along the edge, in meters. |
| Integer nu (param. 4): | maximum free-flow speed along the edge, in meters per second. |

**Returns:** nothing.
**Side Effects:** inserts a row into Table E, puts an entry into lu_edges.
**Throws:** DuplicateEdgeException if edge already exists, or SQLException for other database failures.

121 ⟨*Write: DBInsertEdge(4)* 121⟩≡ (133a)

```
void DBInsertEdge(final int v1, final int v2, final int dd, final int nu)
throws DuplicateEdgeException, SQLException {
  if (this.lu_edges.containsKey(v1) && this.lu_edges.get(v1).containsKey(v2)) {
    throw new DuplicateEdgeException("Edge ("+v1+", "+v2+") already exists.");
  }
  if (!this.lu_edges.containsKey(v1)) {
    this.lu_edges.put(v1, new ConcurrentHashMap<Integer, int[]>());
  }
  try (⟨Open conn 35c⟩) {
    try {
      PreparedStatement pS1 = this.PSCreate(conn, "S1");
      this.PSAdd(pS1, v1, v2, dd, nu);
      this.PSSubmit(pS1);
      conn.commit();
    } catch (SQLException e) {
      conn.rollback();
      throw e;
    }
  } catch (SQLException e) {
    throw e;
  }
  this.lu_edges.get(v1).put(v2, new int[] { dd, nu });
}
```

Defines:
DBInsertEdge, used in chunk 144b.
Uses DuplicateEdgeException 63a, PSAdd 55b, PSCreate 55a, PSSubmit 55c, and S1 46b.

### DBUpdateEdgeSpeed(3)

Method DBUpdateEdgeSpeed(3) updates the maximum free-flow speed of an edge in the road network. If the edge attempting to be updated does not exist, an EdgeNotFoundException is throw. A SQLException is thrown for other database failures.

**Parameters:**

| | |
|---|---|
| Integer v1 (param. 1): | source vertex identifier. |
| Integer v2 (param. 2): | target vertex identifier. |
| Integer nu (param. 3): | new maximum free-flow speed, in meters per second. |

**Returns:** nothing.
**Side Effects:** updates a row in Table E, updates an entry in lu_edges. **May update rows in Table W if edge belongs to any server route. This update may cause C56 violations if waypoint times (columns t1, t2) are not updated accordingly!**
**Throws:** EdgeNotFoundException if edge does not exist, or SQLException for other database failures.

122a ⟨*Write: DBUpdateEdgeSpeed(3)* 122a⟩≡ (133a)
```
  void DBUpdateEdgeSpeed(final int v1, final int v2, final int nu)
  throws EdgeNotFoundException, SQLException {
    if (!(this.lu_edges.containsKey(v1) && this.lu_edges.get(v1).containsKey(v2))) {
      throw new EdgeNotFoundException("Edge ("+v1+", "+v2+") not found.");
    }
    try (⟨Open conn 35c⟩) {
      try {
        PreparedStatement pS15 = this.PSCreate(conn, "S15");
        PreparedStatement pS131 = this.PSCreate(conn, "S131");
        this.PSAdd(pS15, nu, v1, v2);
        this.PSAdd(pS131, nu, v1, v2);
        this.PSSubmit(pS15, pS131);
        conn.commit();
      } catch (SQLException e) {
        conn.rollback();
        throw e;
      }
    } catch (SQLException e) {
      throw e;
    }
    this.lu_edges.get(v1).get(v2)[1] = nu;
  }
```
Defines:
  `DBUpdateEdgeSpeed`, never used.
Uses `EdgeNotFoundException` 63d, `PSAdd` 55b, `PSCreate` 55a, `PSSubmit` 55c, `S131` 48a, and `S15` 47n.

### 3.4.3 Methods: Write Users

`DBInsertRequest`**(1)**

Method `DBInsertRequest`(1) inserts a new request into the user tables and into `lu_users` and `lu_rstatus` if all succeeds. If the request attempting to be inserted already exists, a `DuplicateUserException` is thrown. A `SQLException` is thrown for other database failures.

**Parameters:**
Array `u` (param. 1): 7-element integer array storing values of request $r$'s components.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 : identifier | $1 : r_q$ | $2 : r_e$ | $3 : r_1$ | $4 : r_o$ | $5 : r_d$ | $6 : d_r$ |

**Returns:** nothing.
**Side Effects:** inserts a row into each of the user tables, insert a row into Table R, puts an entry into `lu_users` and into `lu_rstatus`.
**Throws:** `DuplicateUserException` if request already exists, or `SQLException` for other database failures.

122b ⟨*Write: DBInsertRequest(1)* 122b⟩≡ (133a) 123a▷
```
  void DBInsertRequest(final int[] u)
  throws DuplicateUserException, SQLException {
    final int uid = u[0];
    if (this.lu_users.containsKey(uid)) {
      throw new DuplicateUserException("User "+uid+" already exists.");
    }
    try (⟨Open conn 35c⟩) {
      try {
        ⟨Insert into user tables new user 116a⟩
        ⟨Insert into R new request 116b⟩
        conn.commit();
      } catch (SQLException e) {
        conn.rollback();
        throw e;
      }
    } catch (SQLException e) {
      throw e;
    }
```

Defines:
  `DBInsertRequest`, used in chunk 123b.
Uses `DuplicateUserException` 63b.

In the last step, we put $r$ into `lu_users` and put it into `lu_rstatus` with the value set to `false` to indicate that it is unassigned. When we put it into `lu_users`, we store a cloned array `u` as the value because we don't want any changes to `u` on the caller side showing up in our cache (we are considering users to be immutable).

123a ⟨*Write: DBInsertRequest(1)* 122b⟩+≡ (133a) ◁122b

```
    this.lu_users.put(u[0], u.clone());
    this.lu_rstatus.put(u[0], false);
    this.count_requests++;
    this.sum_distance_unassigned += u[6];
    this.sum_distance_base_requests += u[6];
  }
```

Method `insertRequest`(1) wraps `DBInsertRequest`(1).

123b ⟨*Write: insertRequest(1)* 123b⟩≡ (141b)

```
  void insertRequest(final int[] u) throws DuplicateUserException, SQLException {
    this.storage.DBInsertRequest(u);
  }
```

Defines:
  `insertRequest`, used in chunk 144d.
Uses `DBInsertRequest` 122b and `DuplicateUserException` 63b.

## `DBInsertServer`(2)

Method `DBInsertServer`(2) inserts a new server into the user tables and into `lu_users` if all succeeds. If the server attempting to be inserted already exists, a `DuplicateUserException` is thrown. The method requires the server's initial route be given in the second parameter. If the supplied route contains an edge that does not exist in Table E, an `EdgeNotFoundException` is thrown. A `SQLException` is thrown for other database failures.

**Parameters:**

Array `u` (param. 1): 7-element integer array storing values of server $s$'s components.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 : identifier | $1 : s_\mathrm{q}$ | $2 : s_\mathrm{e}$ | $3 : s_1$ | $4 : s_\mathrm{o}$ | $5 : s_\mathrm{d}$ | $6 : d_s$ |

Array `route` (param. 2): $(2|w|)$-element integer array storing values of waypoint components in the server's route $w$.

| | | | | |
|---|---|---|---|---|
| $0 : t_1$ | $1 : v_1$ | $\cdots$ | $(2|w|-2) : t_{|w|}$ | $(2|w|-1) : v_{|w|}$ |

**Returns:** nothing.
**Side Effects:** inserts a row into each of the user tables, insert a row into Table S, inserts at least two rows into Table W, inserts a row into Table CW, inserts a row into Table CQ, puts an entry into `lu_users`.
**Throws:** `DuplicateUserException` if server already exists, `EdgeNotFoundException` if `route` contains an edge that does not exist in Table E, or `SQLException` for other database failures.

123c ⟨*Write: DBInsertServer(2)* 123c⟩≡ (133a) 124a ▷

```
  void DBInsertServer(final int[] u, final int[] route)
  throws DuplicateUserException, EdgeNotFoundException, SQLException {
    final int uid = u[0];
    if (this.lu_users.containsKey(uid)) {
      throw new DuplicateUserException("User "+uid+" already exists.");
    }
    try (⟨Open conn 35c⟩) {
      try {
        final int se = u[2];
        ⟨Insert into user tables new user 116a⟩
        ⟨Insert into S new server 116c⟩
        ⟨Insert into W new server route 116d⟩
        ⟨Insert into CW new server route 116f⟩
        ⟨Insert into CQ new server 117a⟩
        conn.commit();
      } catch (SQLException e) {
        conn.rollback();
        throw e;
      }
    } catch (SQLException e) {
      throw e;
    }
```
Uses DuplicateUserException 63b and EdgeNotFoundException 63d.

In the last step, we put *s* into lu_users.

124a ⟨*Write: DBInsertServer(2)* 123c⟩+≡ (133a) ◁123c

```
    this.lu_users.put(uid, u.clone());
    this.lu_lvt.put(uid, 0);
    this.sum_distance_base_servers += u[6];
    this.distance_servers.put(uid, u[6]);
    this.distance_servers_cruising.put(uid, u[6]);
    this.duration_servers.put(uid, (route[(route.length - 2)] - route[0]));
    this.duration_servers_cruising.put(uid, (route[(route.length - 2)] - route[0]));
  }
```

Method insertServer(2) wraps insertServer(2).

124b ⟨*Write: insertServer(2)* 124b⟩≡ (141b)

```
  void insertServer(final int[] u)
  throws DuplicateUserException, EdgeNotFoundException, SQLException,
         GtreeNotLoadedException, GtreeIllegalSourceException, GtreeIllegalTargetException {
    this.storage.DBInsertServer(u, this.tools.computeRoute(u[4], u[5], u[2]));
  }
```
Defines:
insertServer, used in chunk 144d.
Uses computeRoute 161b, DuplicateUserException 63b, EdgeNotFoundException 63d, GtreeIllegalSourceException 63e, GtreeIllegalTargetException 64a, and GtreeNotLoadedException 64b.

### 3.4.4 Methods: Write Server Properties

#### DBUpdateServerService(5)

124c ⟨*Write: DBUpdateServerService(5)* 124c⟩≡ (133a) 125a▷

```
  void DBUpdateServerService(final int sid, final int[] route, final int[] sched,
      final int[] ridpos, final int[] ridneg)
  throws UserNotFoundException, EdgeNotFoundException, SQLException {
    ⟨Debug: DBUpdateServerService parameters 126a⟩
    if (!this.lu_users.containsKey(sid)) {
      throw new UserNotFoundException("User "+sid+" not found.");
    }
    for (final int r : ridpos) {
      if (!this.lu_users.containsKey(r)) {
        throw new UserNotFoundException("User "+r+" not found.");
      }
    }
    for (final int r : ridneg) {
      if (!this.lu_users.containsKey(r)) {
        throw new UserNotFoundException("User "+r+" not found.");
      }
    }
    {
      int count = 0;
      for (int i = 0; i < (sched.length - 3); i += 4) {
        if (sched[(i + 2)] == sid) {
          count++;
        }
      }
      if (count != 1) {
        throw new UserNotFoundException("Server "+sid+" not found in schedule!");
      }
    }
    Map<Integer, int[]> cache  = new HashMap<>();
    Map<Integer, int[]> cache2 = new HashMap<>();
    try (⟨Open conn 35c⟩) {
      conn.setTransactionIsolation(Connection.TRANSACTION_READ_UNCOMMITTED);
      Statement temp = conn.createStatement();
      temp.execute("LOCK TABLE CQ  IN EXCLUSIVE MODE");
      temp.execute("LOCK TABLE CW  IN EXCLUSIVE MODE");
      temp.execute("LOCK TABLE W   IN EXCLUSIVE MODE");
      temp.execute("LOCK TABLE PD  IN EXCLUSIVE MODE");
      temp.execute("LOCK TABLE CPD IN EXCLUSIVE MODE");
      try {
        final int sq = lu_users.get(sid)[1];
        final int se = lu_users.get(sid)[2];
        ⟨Procedure to update route 118b⟩
        ⟨Procedure to update and add to schedule 118c⟩
        ⟨Delete from PD, CPD jobs 115c⟩
        conn.commit();
      } catch (SQLException e) {
        conn.rollback();
        throw e;
      }
    } catch (SQLException e) {
      throw e;
    }
```

Defines:
  DBUpdateServerService, used in chunks 125b and 126a.
Uses EdgeNotFoundException 63d and UserNotFoundException 65a.

125a ⟨*Write: DBUpdateServerService(5)* 124c⟩+≡ (133a) ◁124c

```
    for (int i = 0; i < (sched.length - 3); i += 4) {
      final int r = sched[(i + 3)];
      if (r != 0) {
        ⟨Cache request transit distance and duration 36b⟩
      }
    }
    for (final int r : ridpos) {
      this.lu_rstatus.put(r, true);
      this.count_assigned++;
      this.sum_distance_unassigned -= this.lu_users.get(r)[6];
    }
    for (final int r : ridneg) {
      this.lu_rstatus.put(r, false);
      this.count_assigned--;
      this.sum_distance_unassigned += this.lu_users.get(r)[6];
      this.distance_requests_transit.put(r, 0);
      this.duration_requests_transit.put(r, 0);
    }
    ⟨Cache server distance 35e⟩
    ⟨Cache cruising distance and duration 36a⟩
  }
```

125b ⟨*Write: updateServerService(5)* 125b⟩≡ (149b)

```
  void updateServerService(final int sid, final int[] route, final int[] sched,
      final int[] ridpos, final int[] ridneg)
  throws RouteIllegalOverwriteException, UserNotFoundException,
         EdgeNotFoundException, TimeWindowException, SQLException {
    final int[] current = this.storage.DBQueryServerRoute(sid);
    int t_now  = this.retrieveClock();
    int t_next = t_now;
    for (int i = 0; i < (current.length - 1); i += 2) {
      if (current[i] > t_next) {
        t_next = current[i];
        break;
      }
    }
    int i = 0;
    while (i < current.length && current[i] != route[0]) {
      i += 2;
    }
    if (i == current.length) {
      ⟨Debug: RouteIllegalOverwriteException, missing branch point 126b⟩
      throw new RouteIllegalOverwriteException("Missing branch point!");
    }
    int j = 0;
    while (i < current.length && (current[i] <= t_next && current[(i + 1)] != 0)) {
      if ((current[(i + 1)] != route[(j + 1)])
       || (current[i] != route[j] && current[i] <= t_now)) {
        ⟨Debug: RouteIllegalOverwriteException, overwrite occurred 126c⟩
        throw new RouteIllegalOverwriteException("Overwrite occurred!");
      }
      i += 2;
      j += 2;
    }
    /*⟨Check time window violation 115a⟩*/
    ⟨Apply traffic to route, sched 114⟩
    this.storage.DBUpdateServerService(sid, mutroute, mutsched, ridpos, ridneg);
  }
```

Defines:
`updateServerAddToSchedule`, never used.
Uses `DBQueryServerRoute` 82c, `DBUpdateServerService` 124c, `EdgeNotFoundException` 63d, `retrieveClock` 58f, `RouteIllegalOverwriteException` 64c, `TimeWindowException` 64d, and `UserNotFoundException` 65a.

### 3.4.5 Debug

#### Debug: DBUpdateServerService Parameters

126a ⟨*Debug: DBUpdateServerService parameters* 126a⟩≡ (124c)

```
  if (DEBUG) {
    System.out.printf("DBUpdateServerService(5), sid=%d, route=[#=%d], sched=[#=%d], ridpos=[#=%d], ridneg=[#=%
        sid, route.length, sched.length, ridpos.length, ridneg.length);
  }
```

Uses `DBUpdateServerService` 124c.

#### Debug: RouteIllegalOverwriteException, Missing Branch Point

126b ⟨*Debug: RouteIllegalOverwriteException, missing branch point* 126b⟩≡ (125b)

```
  if (DEBUG) {
    for (i = 0; i < current.length - 1; i+=2) {
      System.out.printf("debug wold[%d..%d]={ %d, %d }\n", i, (i + 1),
          current[i], current[i+1]);
    }
    for (i = 0; i < route.length - 1; i+=2) {
      System.out.printf("debug wnew[%d..%d]={ %d, %d }\n", i, (i + 1),
          route[i], route[i+1]);
    }
  }
```

#### Debug: RouteIllegalOverwriteException, Overwrite Occurred

126c ⟨*Debug: RouteIllegalOverwriteException, overwrite occurred* 126c⟩≡ (125b)

```
  if (DEBUG) {
    System.out.printf("overwrite, current[%d] != route[%d] or current[%d] != route[%d]\n",
        i, j, (i + 1), (j + 1));
    for (i = 0; i < current.length - 1; i+=2) {
      System.out.printf("debug wold[%d..%d]={ %d, %d }\n", i, (i + 1),
          current[i], current[i+1]);
    }
    for (i = 0; i < route.length - 1; i+=2) {
      System.out.printf("debug wnew[%d..%d]={ %d, %d }\n", i, (i + 1),
          route[i], route[i+1]);
    }
  }
```

## Debug: EdgeNotFoundException

127 ⟨*Debug: EdgeNotFoundException* 127⟩≡ (118a)

```
  if (DEBUG) {
    System.out.printf("edge (%d, %d) not found; print route until now\n", v1, v2);
    for (int j = 0; j <= i; j += 2) {
      System.out.printf("debug route[%d..%d]~>route[%d..%d]={ %d, %d }~>{ %d, %d }\n",
        j, (j + 1), (j + 2), (j + 3), route[j], route[j+1], route[j+2], route[j+3]);
    }
  }
```

## 3.5 G-tree Operations

### 3.5.1 Chunks

### 3.5.2 Methods

#### GTGtreeLoad(1)

128a ⟨*Gtree: GTGtreeLoad(1)* 128a⟩≡ (158e)

```
void GTGtreeLoad(final String p) throws FileNotFoundException {
  try {
    System.loadLibrary("gtree");
  } catch (UnsatisfiedLinkError e) {
    System.err.println("Native code library failed to load: "+e);
    System.exit(1);
  }
  if (p.length() > 0) {
    gtreeJNI.setIndex_path(p);
    this.gtree = new GTree();
    gtreeJNI.read_GTree(gtree);
    this.flag_gtree_loaded = true;
  } else {
    throw new FileNotFoundException("Bad path to gtree");
  }
}
```

Defines:
`GTGtreeLoad`, used in chunk 128b.

Method gtreeLoad(1) wraps GTGtreeLoad(1).

128b ⟨*Gtree: gtreeLoad(1)* 128b⟩≡ (142a 151c)

```
void gtreeLoad(String p) throws FileNotFoundException {
  this.tools.GTGtreeLoad(p);
}
```

Defines:
`gtreeLoad`, used in chunks 167b, 191b, and 196.
Uses `GTGtreeLoad` 128a.

#### GTGtreeClose(0)

128c ⟨*Gtree: GTGtreeClose(0)* 128c⟩≡ (158e)

```
void GTGtreeClose() {
  this.gtree = null;
  this.flag_gtree_loaded = false;
}
```

Defines:
`GTGtreeClose`, used in chunk 128d.

Method gtreeClose(0) wraps GTGtreeClose(0).

128d ⟨*Gtree: gtreeClose(0)* 128d⟩≡ (142a 151c)

```
void gtreeClose() {
  this.tools.GTGtreeClose();
}
```

Defines:
`gtreeClose`, used in chunk 203.
Uses `GTGtreeClose` 128c.

## 3.6 Class: Storage

The `Storage` class is a data-access layer for the schema in the Derby database.

```
129a  ⟨Storage.java 129a⟩≡
        ⟨Package: sim 35a⟩
        ⟨Storage.java preamble 129b⟩
        public class Storage {
          ⟨Storage member variables 130b⟩
          ⟨Storage constructor 131b⟩
          ⟨Storage methods 131c⟩
        }
```

### 3.6.1 Preamble

We import:

- various Jargo exceptions for exception handling;

```
129b  ⟨Storage.java preamble 129b⟩≡                                   (129a) 129c▷
        import com.github.jargors.sim.DuplicateVertexException;
        import com.github.jargors.sim.DuplicateEdgeException;
        import com.github.jargors.sim.DuplicateUserException;
        import com.github.jargors.sim.EdgeNotFoundException;
        import com.github.jargors.sim.UserNotFoundException;
        import com.github.jargors.sim.VertexNotFoundException;
        import com.github.jargors.sim.TimeWindowException;
```

Uses `DuplicateEdgeException` 63a, `DuplicateUserException` 63b, `DuplicateVertexException` 63c, `EdgeNotFoundException` 63d, `TimeWindowException` 64d, `UserNotFoundException` 65a, and `VertexNotFoundException` 65b.

- parts of the JDBC API from `java.sql`, for communication with Derby;

```
129c  ⟨Storage.java preamble 129b⟩+≡                          (129a) ◁129b 129d▷
        import java.sql.CallableStatement;   import java.sql.Connection;
        import java.sql.DriverManager;       import java.sql.PreparedStatement;
        import java.sql.ResultSet;           import java.sql.SQLException;
        import java.sql.Statement;           import java.sql.Types;
```

- Apache DBCP2 and Pool2, for connection pooling;

```
129d  ⟨Storage.java preamble 129b⟩+≡                          (129a) ◁129c 130a▷
        import org.apache.commons.dbcp2.ConnectionFactory;
        import org.apache.commons.dbcp2.DriverManagerConnectionFactory;
        import org.apache.commons.dbcp2.PoolableConnection;
        import org.apache.commons.dbcp2.PoolableConnectionFactory;
        import org.apache.commons.dbcp2.PoolingDriver;
        import org.apache.commons.pool2.ObjectPool;
        import org.apache.commons.pool2.impl.GenericObjectPool;
        import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
```

- standard map classes for caching various items.

130a ⟨*Storage.java preamble* 129b⟩+≡ (129a) ◁129d

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Map;
import java.util.HashMap;
import java.util.concurrent.ConcurrentHashMap;
```

### 3.6.2 Member Variables

The storage interface caches static data to avoid unnecessary database queries.

- The `lu_rstatus` map stores a boolean flag for each request indicating whether the request is assigned or not. While the map elements are not static, the elements do not change often. Guaranteeing consistency of the map is easy because only `Storage` can update assignment changes to the database. Whenever it successfully does an update, we simply update the map at the same time.

130b ⟨`Storage` *member variables* 130b⟩≡ (129a) 130c ▷

```
private Map<Integer, Boolean> lu_rstatus = new HashMap<>();  //*
```

- The other maps store static data values. References to these maps may be held by other threads. To avoid accidental corruption due to concurrent access, we use `ConcurrentHashMap`.

130c ⟨`Storage` *member variables* 130b⟩+≡ (129a) ◁130b 130d ▷

```
private ConcurrentHashMap<String, String> lu_pstr     = new ConcurrentHashMap<String, String>();
private ConcurrentHashMap<Integer, int[]> lu_vertices = new ConcurrentHashMap<Integer, int[]>();
private ConcurrentHashMap<Integer,
    ConcurrentHashMap<Integer, int[]>>    lu_edges    = new ConcurrentHashMap<Integer, ConcurrentHashMap
private ConcurrentHashMap<Integer, int[]> lu_users    = new ConcurrentHashMap<Integer, int[]>();
private Map<Integer, Integer>             lu_lvt      = new HashMap<Integer, Integer>();
```

Cache some stats for metrics:

130d ⟨`Storage` *member variables* 130b⟩+≡ (129a) ◁130c 130e ▷

```
private int count_requests = 0;
private int count_assigned = 0;
private int sum_distance_unassigned = 0;
private int sum_distance_base_requests = 0;
private int sum_distance_base_servers = 0;
private Map<Integer, Integer> distance_servers = new HashMap<Integer, Integer>();
private Map<Integer, Integer> distance_servers_cruising = new HashMap<Integer, Integer>();
private Map<Integer, Integer> distance_requests_transit = new HashMap<Integer, Integer>();
private Map<Integer, Integer> duration_servers = new HashMap<Integer, Integer>();
private Map<Integer, Integer> duration_servers_cruising = new HashMap<Integer, Integer>();
private Map<Integer, Integer> duration_requests_transit = new HashMap<Integer, Integer>();
private Map<Integer, Integer> duration_requests_travel = new HashMap<Integer, Integer>();
private Map<Integer, Integer> duration_requests_pickup = new HashMap<Integer, Integer>();
```

The storage interface also contains configuration settings and JDBC objects.

- Parameter `STATEMENTS_MAX_COUNT` gives maximum number of simultaneous prepared statements. Parameter `REQUEST_TIMEOUT` sets how long a client is allowed to try to match a request, in other words if request $r$ is not assigned within $\pi_{\mathsf{e}}(r) + \texttt{REQUEST\_TIMEOUT}$, then it is not tried again. The remaining parameters configure the Derby database connection.

130e ⟨`Storage` *member variables* 130b⟩+≡ (129a) ◁130d 131a ▷

```
private final int    STATEMENTS_MAX_COUNT   = 20;
private       int    REQUEST_TIMEOUT        = 30;
private       String CONNECTIONS_URL        = "jdbc:derby:memory:jargo;create=true";
private final String CONNECTIONS_DRIVER_URL = "jdbc:apache:commons:dbcp:";
private final String CONNECTIONS_POOL_NAME  = "jargo";
private final String CONNECTIONS_POOL_URL   = (CONNECTIONS_DRIVER_URL + CONNECTIONS_POOL_NAME);
public static final double CSHIFT           = 10000000.0;
private final boolean DEBUG =
    "true".equals(System.getProperty("jargors.storage.debug"));
```

- The connection_factory is an object that returns new connections. The poolableconnection_factory registers connection_factory and then can return new poolable connections. The pool is an object containing the available poolable connections, and it is registered by the poolableconnection_factory. The driver is the JDBC DriverManager. We get a reference to DriverManager in order to register the pool.

131a ⟨Storage *member variables* 130b⟩+≡ (129a) ◁130e

```
  private ConnectionFactory                   connection_factory;
  private PoolableConnectionFactory           poolableconnection_factory;
  private ObjectPool<PoolableConnection>      pool;
  private PoolingDriver                       driver;
```

### 3.6.3 Constructor

Constructor Storage(0) simply initializes the prepared statement strings into lu_pstr by using JargoSetupPreparedStatements(0).

**Parameters:** none.
**Returns:** nothing.
**Side Effects:** adds entries into lu_pstr.
**Throws:** nothing.

131b ⟨Storage *constructor* 131b⟩≡ (129a)

```
  public Storage() {
    this.JargoSetupPreparedStatements();
  }
```

Uses JargoSetupPreparedStatements 45.

### 3.6.4 Methods

#### Read Methods

131c ⟨Storage *methods* 131c⟩≡ (129a) 132▷

```
  public ⟨Read: DBQuery(2) 67b⟩
  public ⟨Read: DBQueryQuick(3) 68a⟩
  public ⟨Read: DBQueryEdge(2) 71b⟩
  public ⟨Read: DBQueryEdgeStatistics(0) 73a⟩
  public ⟨Read: DBQueryEdges(0) 72a⟩
  public ⟨Read: DBQueryEdgesCount(0) 72c⟩
  public ⟨Read: DBQueryMBR(0) 69a⟩
  public ⟨Read: DBQueryMetricRequestDistanceBaseTotal(0) 106c⟩
  public ⟨Read: DBQueryMetricRequestDistanceBaseUnassignedTotal(1) 107a⟩
  public ⟨Read: DBQueryMetricRequestDistanceBaseUnassignedRunning(0) 107e⟩
  public ⟨Read: DBQueryMetricRequestDistanceDetourTotal(1) 108b⟩
  public ⟨Read: DBQueryMetricRequestDistanceTransitTotal(1) 109b⟩
  public ⟨Read: DBQueryMetricRequestDurationPickupTotal(1) 110b⟩
  public ⟨Read: DBQueryMetricRequestDurationTransitTotal(1) 111a⟩
  public ⟨Read: DBQueryMetricRequestDurationTravelTotal(1) 112a⟩
  public ⟨Read: DBQueryMetricRequestTWViolationsTotal(0) 112e⟩
  public ⟨Read: DBQueryMetricServerDistanceBaseTotal(0) 101b⟩
  public ⟨Read: DBQueryMetricServerDistanceCruisingTotal(1) 102a⟩
  public ⟨Read: DBQueryMetricServerDistanceRunning(0) 100e⟩
  public ⟨Read: DBQueryMetricServerDistanceServiceTotal(1) 102e⟩
  public ⟨Read: DBQueryMetricServerDistanceTotal(1) 100a⟩
  public ⟨Read: DBQueryMetricServerDurationCruisingTotal(1) 104d⟩
  public ⟨Read: DBQueryMetricServerDurationServiceTotal(1) 105c⟩
  public ⟨Read: DBQueryMetricServerDurationTravelTotal(1) 103d⟩
  public ⟨Read: DBQueryMetricServerTWViolationsTotal(0) 106a⟩
  public ⟨Read: DBQueryMetricServiceRate(1) 97b⟩
  public ⟨Read: DBQueryMetricUserDistanceBaseTotal(1) 98e⟩
  public ⟨Read: DBQueryMetricUserDistanceBaseRunning(0) 99d⟩
  public ⟨Read: DBQueryRequestDistanceDetour(2) 75b⟩
  public ⟨Read: DBQueryRequestDistanceTransit(2) 76a⟩
  public ⟨Read: DBQueryRequestDurationPickup(2) 76b⟩
  public ⟨Read: DBQueryRequestDurationTransit(2) 77a⟩
  public ⟨Read: DBQueryRequestDurationTravel(2) 77b⟩
  public ⟨Read: DBQueryRequestIsAssigned(2) 75a⟩
  public ⟨Read: DBQueryRequestStatus(2) 74c⟩
  public ⟨Read: DBQueryRequestTimeOfArrival(1) 78c⟩
  public ⟨Read: DBQueryRequestTimeOfDeparture(1) 78a⟩
  public ⟨Read: DBQueryRequestsCount(0) 79b⟩
  public ⟨Read: DBQueryRequestsCountActive(1) 79d⟩
  public ⟨Read: DBQueryRequestsCountAssigned(0) 80d⟩
  public ⟨Read: DBQueryRequestsCountCompleted(1) 80f⟩
  public ⟨Read: DBQueryRequestsQueued(1) 81b⟩
  public ⟨Read: DBQueryRequestsWaiting(1) 82a⟩
  public ⟨Read: DBQueryServerAssignmentsCompleted(2) 93a⟩
  public ⟨Read: DBQueryServerAssignmentsPending(2) 92c⟩
  public ⟨Read: DBQueryServerCapacityViolations(4) 86b⟩
  public ⟨Read: DBQueryServerDistance(2) 87a⟩
  public ⟨Read: DBQueryServerDistanceCruising(2) 88b⟩
  public ⟨Read: DBQueryServerDistanceRemaining(2) 87c⟩
  public ⟨Read: DBQueryServerDistanceService(2) 88c⟩
  public ⟨Read: DBQueryServerDurationCruising(2) 90c⟩
  public ⟨Read: DBQueryServerDurationService(2) 90d⟩
  public ⟨Read: DBQueryServerDurationRemaining(2) 89a⟩
  public ⟨Read: DBQueryServerDurationTravel(2) 90a⟩
  public ⟨Read: DBQueryServerLoadMax(2) 85b⟩
  public ⟨Read: DBQueryServerRoute(1) 82c⟩
  public ⟨Read: DBQueryServerRouteActive(1) 83c⟩
  public ⟨Read: DBQueryServerRouteRemaining(2) 83a⟩
  public ⟨Read: DBQueryServerSchedule(1) 84a⟩
  public ⟨Read: DBQueryServerScheduleRemaining(2) 84c⟩
  public ⟨Read: DBQueryServerTimeOfArrival(1) 92a⟩
  public ⟨Read: DBQueryServerTimeOfDeparture(1) 91a⟩
  public ⟨Read: DBQueryServersActive(1) 93b⟩
  public ⟨Read: DBQueryServersCount(0) 94a⟩
  public ⟨Read: DBQueryServersCountActive(1) 94c⟩
  public ⟨Read: DBQueryServersLocations(1) 95c⟩
  public ⟨Read: DBQueryServersLocationsActive(1) 96a⟩
  public ⟨Read: DBQueryUser(1) 73c⟩
  public ⟨Read: DBQueryUsers(0) 74b⟩
  public ⟨Read: DBQueryVertex(1) 69c⟩
  public ⟨Read: DBQueryVertices(0) 70b⟩
  public ⟨Read: DBQueryVerticesCount(0) 70d⟩
```

## Cached Read Methods

132 ⟨Storage *methods* 131c⟩+≡ (129a) ◁131c 133a▷

```
  public ⟨Read: DBQueryMetricRequestDistanceBaseUnassignedTotal(0) 107b⟩
  public ⟨Read: DBQueryMetricRequestDistanceDetourTotal(0) 108c⟩
  public ⟨Read: DBQueryMetricRequestDistanceTransitTotal(0) 109c⟩
  public ⟨Read: DBQueryMetricRequestDurationPickupTotal(0) 110c⟩
  public ⟨Read: DBQueryMetricRequestDurationTransitTotal(0) 111b⟩
  public ⟨Read: DBQueryMetricRequestDurationTravelTotal(0) 112b⟩
  public ⟨Read: DBQueryMetricServerDistanceCruisingTotal(0) 102b⟩
  public ⟨Read: DBQueryMetricServerDistanceServiceTotal(0) 103a⟩
  public ⟨Read: DBQueryMetricServerDistanceTotal(0) 100b⟩
  public ⟨Read: DBQueryMetricServerDurationCruisingTotal(0) 104e⟩
  public ⟨Read: DBQueryMetricServerDurationServiceTotal(0) 105d⟩
  public ⟨Read: DBQueryMetricServerDurationTravelTotal(0) 104a⟩
  public ⟨Read: DBQueryMetricServiceRate(0) 97c⟩
  public ⟨Read: DBQueryMetricUserDistanceBaseTotal(0) 99a⟩
```

## Write Methods

133a ⟨Storage *methods* 131c⟩+≡ (129a) ◁132 133b▷

```
  public ⟨Write: DBInsertEdge(4) 121⟩
  public ⟨Write: DBInsertRequest(1) 122b⟩
  public ⟨Write: DBInsertServer(2) 123c⟩
  public ⟨Write: DBInsertVertex(3) 120a⟩
  public ⟨Write: DBUpdateEdgeSpeed(3) 122a⟩
  public ⟨Write: DBUpdateServerService(5) 124c⟩
```

## Administration

133b ⟨Storage *methods* 131c⟩+≡ (129a) ◁133a

```
  public ⟨Admin: JargoCacheRoadNetworkFromDB(0) 41b⟩
  public ⟨Admin: JargoCacheUsersFromDB(0) 42e⟩
  public ⟨Admin: JargoInstanceClose(0) 40c⟩
  public ⟨Admin: JargoInstanceExport(1) 40a⟩
  public ⟨Admin: JargoInstanceInitialize(0) 37c⟩
  public ⟨Admin: JargoInstanceLoad(1) 39b⟩
  public ⟨Admin: JargoInstanceNew(0) 37a⟩
  public ⟨Admin: getRefCacheEdges(0) 57d⟩
  public ⟨Admin: getRefCacheUsers(0) 57e⟩
  public ⟨Admin: getRefCacheVertices(0) 58a⟩
  public ⟨Admin: setRequestTimeout(1) 60d⟩
  private ⟨Admin: JargoSetupDriver(0) 44b⟩
  private ⟨Admin: JargoSetupPreparedStatements(0) 45⟩
  private ⟨Admin: PSAdd(2..) 55b⟩
  private ⟨Admin: PSCreate(2) 55a⟩
  private ⟨Admin: PSQuery(3..) 56a⟩
  private ⟨Admin: PSSubmit(1..) 55c⟩
```

## 3.7 Class: Controller

134a ⟨*Controller.java* 134a⟩≡
```
  ⟨Package: sim 35a⟩
  ⟨Controller.java preamble 134b⟩
  public class Controller {
    ⟨Controller member variables 135e⟩
    ⟨Controller constructor 140a⟩
    ⟨Controller methods 140c⟩
  }
```

### 3.7.1 Preamble

We import:

- all parts of the Jargo stack;

134b ⟨*Controller.java preamble* 134b⟩≡ (134a) 135a ▷
```
  import com.github.jargors.sim.Storage;
  import com.github.jargors.sim.Communicator;
  import com.github.jargors.sim.Client;
  import com.github.jargors.sim.Tools;
  import com.github.jargors.sim.ClientException;
  import com.github.jargors.sim.ClientFatalException;
  import com.github.jargors.sim.DuplicateVertexException;
  import com.github.jargors.sim.DuplicateEdgeException;
  import com.github.jargors.sim.DuplicateUserException;
  import com.github.jargors.sim.EdgeNotFoundException;
  import com.github.jargors.sim.UserNotFoundException;
  import com.github.jargors.sim.VertexNotFoundException;
  import com.github.jargors.sim.GtreeNotLoadedException;
  import com.github.jargors.sim.GtreeIllegalSourceException;
```

```
    import com.github.jargors.sim.GtreeIllegalTargetException;
```

Uses DuplicateEdgeException 63a, DuplicateUserException 63b, DuplicateVertexException 63c, EdgeNotFoundException 63d, GtreeIllegalSourceException 63e, GtreeIllegalTargetException 64a, GtreeNotLoadedException 64b, UserNotFoundException 65a, and VertexNotFoundException 65b.

- standard utilities for concurrent execution;

135a ⟨*Controller.java preamble* 134b⟩+≡ (134a) ◁134b 135b▷

```
    import java.util.ArrayList;
    import java.util.concurrent.CompletableFuture;
    import java.util.concurrent.Executors;
    import java.util.concurrent.ConcurrentHashMap;
    import java.util.concurrent.ScheduledExecutorService;
    import java.util.concurrent.ScheduledFuture;
    import java.util.concurrent.TimeUnit;
    import java.util.function.Consumer;
```

- standard classes for file operations;

135b ⟨*Controller.java preamble* 134b⟩+≡ (134a) ◁135a 135c▷

```
    import java.util.Scanner;
    import java.io.File;
    import java.io.FileNotFoundException;
```

- standard map classes for caching various items.

135c ⟨*Controller.java preamble* 134b⟩+≡ (134a) ◁135b 135d▷

```
    import java.util.Map;
    import java.util.HashMap;
```

135d ⟨*Controller.java preamble* 134b⟩+≡ (134a) ◁135c

```
    import java.sql.SQLException;
```

### 3.7.2 Member Variables

Member variables are grouped into *containers*, *settings*, and *loops*.

135e ⟨Controller *member variables* 135e⟩≡ (134a)
  ⟨*Container objects* 135f⟩
  ⟨*Settings objects* 136a⟩
  ⟨*Loop objects* 136e⟩

**Containers.**

135f ⟨*Container objects* 135f⟩≡ (135e)

```
  private Storage storage;
  private Communicator communicator;
  private Tools tools = new Tools();
  private Client client;
  private Map<Integer, Boolean> lu_rseen = new HashMap<Integer, Boolean>();
  private Map<Integer, Boolean> lu_sseen = new HashMap<Integer, Boolean>();
  private String refTimeStr = "";
  private long refTimeMs = 0;
  private int simClock = 0;
  private int simClockReferenceDay = 0;
  private int simClockReferenceMinute = 0;
  private int simClockReferenceHour = 0;
  private int simClockReferenceSecond = 0;
  private long dur_query = 0;
```

**Settings.** Settings objects configure various aspects of the simulation.

136a ⟨*Settings objects* 136a⟩≡ (135e) 136b ▷
```
  private int CLOCK_START =
      Integer.parseInt(System.getProperty("jargors.controller.clock_start", "0"));
  private int CLOCK_END =
      Integer.parseInt(System.getProperty("jargors.controller.clock_end", "1800"));
  private int REQUEST_TIMEOUT =
      Integer.parseInt(System.getProperty("jargors.controller.request_timeout", "30"));
  private int QUEUE_TIMEOUT =
      Integer.parseInt(System.getProperty("jargors.controller.queue_timeout", "30"));
  private int REQUEST_COLLECTION_PERIOD =
      Integer.parseInt(System.getProperty("jargors.controller.request_collection_period", "1"));
  private int REQUEST_HANDLING_PERIOD =
      Integer.parseInt(System.getProperty("jargors.controller.request_handling_period", "1"));
  private int SERVER_COLLECTION_PERIOD =
      Integer.parseInt(System.getProperty("jargors.controller.server_collection_period", "1"));
```

The `loop_delay` configures how many seconds to wait until the controller loops start. The update periods configure how often particular loops should execute, in seconds.

136b ⟨*Settings objects* 136a⟩+≡ (135e) ◁ 136a 136c ▷
```
  private int loop_delay = 0;
  // private int deviation_rate = 0.02;
  // private int breakdown_rate = 0.005;
```

The `CSHIFT` setting configures the precision for longitude and latitude coordiates (see `Storage.DBInsertVertex`(3)).

136c ⟨*Settings objects* 136a⟩+≡ (135e) ◁ 136b 136d ▷
```
  private final double CSHIFT = Storage.CSHIFT;
  private boolean kill = false;
  private boolean working = false;
  private ScheduledExecutorService exe = null;
  private ScheduledFuture<?> cb1 = null;
  private ScheduledFuture<?> cb2 = null;
  private ScheduledFuture<?> cb3 = null;
  private ScheduledFuture<?> cb4 = null;
  private ScheduledFuture<?> cb5 = null;
```

The `DEBUG` setting controls whether certain messages are print to screen. Pass `-Djargors.controller.debug=true` to the `java` command to set `DEBUG` to `true`.

136d ⟨*Settings objects* 136a⟩+≡ (135e) ◁ 136c
```
  private final boolean DEBUG =
      "true".equals(System.getProperty("jargors.controller.debug"));
```

**Loops.** Jargo's simulation environment comprises four "loops", defined here, running in parallel. They are executed using Java's `ScheduledExecutorService` to control timing.

136e ⟨*Loop objects* 136e⟩≡ (135e)
```
  ⟨Definition of clock loop 136f⟩
  ⟨Definition of request collection loop 137⟩
  ⟨Definition of request handling loop 138⟩
  ⟨Definition of server collection loop 139⟩
```

### Clock Loop

Member `ClockLoop` is a `Runnable` that does two things. First it advances the simulation world time, and then it tells `communicator` about the new time.

**Parameters:** none.
**Returns:** nothing.
**Side Effects:** increments `simClock` by 1, may modify `communicator` by changing `Communicator.simClock`, may print to standard error if `DEBUG` is `true`.
**Throws:** nothing.

136f ⟨*Definition of clock loop* 136f⟩≡ (136e)
```
  private Runnable ClockLoop = () -> {
    // TODO: The speed of the updateServer.. methods is about 50ms, meaning we
    // can do ~20 updates per second. If a problem instance has more than 20
    // requests per second and an algo is fast enough to do more than 20 updates
    // per second, the updates will become the bottleneck. It might be unfair
    // to the algo if we advance the clock while waiting for updates to finish.
    // So in this case we only advance the clock after the updates finish.
    // How to implement? We just measure the time it takes to do an update and
    // add that duration onto the clock. We can output a "clock rate" to show
    // the user the current simulation rate, i.e. clock_rate=1x means real-time,
    // clock_rate=0.5x means 1 simulated second takes 2 real seconds, etc.
    this.simClock++;
    this.simClockReferenceSecond++;
    if (this.simClockReferenceSecond > 59) {
      this.simClockReferenceSecond = 0;
      this.simClockReferenceMinute++;
      if (this.simClockReferenceMinute > 59) {
        this.simClockReferenceMinute = 0;
        this.simClockReferenceHour++;
        if (this.simClockReferenceHour > 23) {
          this.simClockReferenceHour = 0;
          this.simClockReferenceDay++;
        }
      }
    }
    if (DEBUG) {
      System.out.printf("t=%d (day %d, %02d:%02d:%02d)\n",
          this.simClock,
          this.simClockReferenceDay,
          this.simClockReferenceHour,
          this.simClockReferenceMinute,
          this.simClockReferenceSecond);
    }
  };
```

### Request Collection Loop

Member `RequestCollectionLoop` is a `Runnable` that collects requests eligible for assignment at the current world time. A request $r$ is "eligible" if it is not assigned at the current world time, and if the world time is between the request's early time $r_e$ and ($r_e$ + `REQUEST_TIMEOUT`) (see `Storage.DBQueryRequestsQueued`(1)). If the eligible requests cannot be collected, we consider this failure to be fatal and exit immediately. A possible reason may be database failure in `storage`.

**Parameters:** none.
**Returns:** nothing.
**Side Effects:** may modify `client` by adding objects into `Client.queue`, may put new entries or modify existing entries in `lu_rseen`, may print to standard error if `DEBUG` is `true`, or exits the JVM if failure occurs.
**Throws:** nothing.

137 ⟨*Definition of request collection loop* 137⟩≡ (136e)

```
  private Runnable RequestCollectionLoop = () -> {
    int  A1 = 0;
    int  A2 = 0;
    final int now = this.simClock;
    try {
      A2 = this.client.dropRequests(now - QUEUE_TIMEOUT);
      if (DEBUG) {
        System.out.printf("drop %d requests\n", A2);
      }
      int[] output = this.storage.DBQueryRequestsQueued(now);
      if (DEBUG) {
        System.out.printf("query %d unassigned requests\n", (output.length/7));
      }
      for (int i = 0; i < (output.length - 6); i += 7) {
        if (!this.lu_rseen.containsKey(output[i]) || this.lu_rseen.get(output[i]) == false) {
          this.client.addRequest(new int[] {
            output[(i + 0)],
            output[(i + 1)],
            output[(i + 2)],
            output[(i + 3)],
            output[(i + 4)],
            output[(i + 5)],
            output[(i + 6)] });
          this.lu_rseen.put(output[i], true);
          A1++;
        }
      }
      if (DEBUG) {
        System.out.printf("add %d new requests\n", A1);
      }
    } catch (SQLException e) {
      if (e.getErrorCode() == 40000) {
        System.err.println("Warning: database connection interrupted");
      } else {
        System.err.println("Encountered fatal error");
        try {
          instanceExport("crash-db");
          System.err.println(e.toString());
          System.err.println(e.getErrorCode());
          e.printStackTrace();
        } catch (Exception ee) {
          // ..
        } finally {
          System.exit(1);
        }
      }
    }
  };
```

Uses addRequest 152a, DBQueryRequestsQueued 81b, dropRequests 152b, instanceExport 40b, and query 68b.

### Request Handling Loop

Member RequestHandlingLoop is a Runnable that notifies the client algorithm to check for and process new requests.

**Parameters:** none.
**Returns:** nothing.
**Side Effects:** may indirectly modify the database underlying storage depending on the body of Client.notifyNew(0). May print to standard error if a ClientException occurs or ClientFatalException occurs or DEBUG is true, or exits the JVM if ClientFatalException occurs.
**Throws:** nothing.

138 ⟨*Definition of request handling loop* 138⟩≡ (136e)

```
  private Runnable RequestHandlingLoop = () -> {
    try {
      this.client.notifyNew();  // blocks this thread until queue is empty
    } catch (ClientException e) {
      System.err.printf("[t=%d] Controller.RequestHandlingLoop caught a ClientException: %s\n",
          this.simClock, e.toString());
      // try {
      //   instanceExport("debug-db");
      // } catch (Exception ee) { }
      e.printStackTrace();
    } catch (ClientFatalException e) {
      System.err.printf("[t=%d] Controller.RequestHandlingLoop caught a ClientFatalException: %s\n",
          this.simClock, e.toString());
      e.printStackTrace();
      System.exit(1);
    } catch (Exception e) {
      System.err.printf("[t=%d] Controller.RequestHandlingLoop caught a unspecified Exception: %s\n",
          this.simClock, e.toString());
      e.printStackTrace();
      System.exit(1);
    }
  };
```
Uses `instanceExport` 40b and `notifyNew` 151e.

## Server Loop

Member `ServerLoop` is a `Runnable` that collects last-known locations of all active servers at the current word time. A server is "active" if its service has not ended, in other words it has not arrived at its own destination. The "last-known location" is the waypoint in the server's route $w$ with a time component closest to but not exceeding the given time, in other words $w_{\leq t_{|w_{\leq t}|}}$ (see `Storage.DBQueryServersLocationsActive`(1)). If the last-known locations cannot be collected, we consider this failure to be fatal and exit immediately. A possible reason may be database failure in `storage`.

**Parameters:** none.
**Returns:** nothing.
**Side Effects:** may indirectly modify the database underlying `storage` depending on the body of `Client.collectServerLocations`(1). May print to standard error if `DEBUG` is `true`, or exits the JVM if failure occurs.
**Throws:** nothing.

139 ⟨*Definition of server collection loop* 139⟩≡ (136e)
```
  private Runnable ServerLoop = () -> {
    try {
      int[] output = this.storage.DBQueryServersLocationsActive(this.simClock);
      if (DEBUG) {
        System.out.printf("got %d servers\n", (output.length/3));
      }
      for (int i = 0; i < (output.length - 2); i += 3) {
        if (!this.lu_sseen.containsKey(output[i])) {
          this.lu_sseen.put(output[i], true);
        }
      }
      this.client.collectServerLocations(output);
    } catch (SQLException e) {
      if (e.getErrorCode() == 40000) {
        System.err.println("Warning: database connection interrupted");
      } else {
        System.err.println("Encountered fatal error");
        System.err.println(e.toString());
        System.err.println(e.getErrorCode());
        e.printStackTrace();
        System.exit(1);
      }
    } catch (Exception e) {
      System.err.printf("[t=%d] Controller.ServerLoop caught a unspecified Exception: %s\n",
          this.simClock, e.toString());
      e.printStackTrace();
      System.exit(1);
    }
  };
```
Uses `collectServerLocations` 152c and `DBQueryServersLocationsActive` 96a.

### 3.7.3 Constructor

Constructor `Controller`(0) registers a new `Storage` to the `storage` member variable. It also registers a new `Communicator` to the `communicator` member variable. It then registers itself and the new `Storage` to `communicator`.

**Parameters:** none.
**Returns:** nothing.
**Side Effects:** creates a new `Storage` and `Communicator` on the memory heap, modifies `storage` and `communicator`.
**Throws:** nothing.

140a ⟨`Controller` *constructor* 140a⟩≡ (134a)

```
public Controller() {
  this.storage = new Storage();
  this.communicator = new Communicator();
  this.communicator.setRefStorage(this.storage);
  this.communicator.setRefController(this);
}
```

Uses `setRefController` 61e and `setRefStorage` 61f.

### 3.7.4 Chunks

140b ⟨*Read header rows* 140b⟩≡ (144d)

```
System.out.println(sc.next());
System.out.println(sc.next());
System.out.println(sc.next());
System.out.println(sc.next());
System.out.println(sc.next());
this.setClockReference(sc.next());
System.out.println(sc.next());
System.out.println(sc.next());
System.out.println(sc.next());
System.out.println(sc.next());
System.out.println(sc.next());
System.out.println(sc.next());
```

Uses `setClockReference` 59d.

### 3.7.5 Methods

#### Read Methods

140c ⟨`Controller` *methods* 140c⟩≡ (134a) 141a ▷

```
public ⟨Read: query(2) 68b⟩
public ⟨Read: queryQuick(3) 68c⟩
public ⟨Read: queryEdge(2) 71c⟩
public ⟨Read: queryEdgeStatistics(0) 73b⟩
public ⟨Read: queryEdges(0) 72b⟩
public ⟨Read: queryEdgesCount(0) 72d⟩
public ⟨Read: queryMBR(0) 69b⟩
public ⟨Read: queryMetricRequestDistanceBaseTotal(0) 106d⟩
public ⟨Read: queryMetricRequestDistanceBaseUnassignedTotal(1) 107c⟩
public ⟨Read: queryMetricRequestDistanceBaseUnassignedRunning(0) 108a⟩
public ⟨Read: queryMetricRequestDistanceDetourTotal(1) 108d⟩
public ⟨Read: queryMetricRequestDistanceTransitTotal(1) 109d⟩
public ⟨Read: queryMetricRequestDurationPickupTotal(1) 110d⟩
public ⟨Read: queryMetricRequestDurationTransitTotal(1) 111c⟩
public ⟨Read: queryMetricRequestDurationTravelTotal(1) 112c⟩
public ⟨Read: queryMetricRequestTWViolationsTotal(0) 113⟩
public ⟨Read: queryMetricServerDistanceBaseTotal(0) 101c⟩
public ⟨Read: queryMetricServerDistanceCruisingTotal(1) 102c⟩
public ⟨Read: queryMetricServerDistanceRunning(0) 101a⟩
public ⟨Read: queryMetricServerDistanceServiceTotal(1) 103b⟩
```

```
  public ⟨Read: queryMetricServerDistanceTotal(1) 100c⟩
  public ⟨Read: queryMetricServerDurationCruisingTotal(1) 105a⟩
  public ⟨Read: queryMetricServerDurationServiceTotal(1) 105e⟩
  public ⟨Read: queryMetricServerDurationTravelTotal(1) 104b⟩
  public ⟨Read: queryMetricServerTWViolationsTotal(0) 106b⟩
  public ⟨Read: queryMetricServiceRate(1) 98a⟩
  public ⟨Read: queryMetricServiceRateRunning(0) 98d⟩
  public ⟨Read: queryMetricUserDistanceBaseTotal(1) 99b⟩
  public ⟨Read: queryMetricUserDistanceBaseRunning(0) 99e⟩
  public ⟨Read: queryRequestTimeOfArrival(1) 79a⟩
  public ⟨Read: queryRequestTimeOfDeparture(1) 78b⟩
  public ⟨Read: queryRequestsCount(0) 79c⟩
  public ⟨Read: queryRequestsCountActive(1) 80a⟩
  public ⟨Read: queryRequestsCountAppeared(0) 80c⟩
  public ⟨Read: queryRequestsCountAssigned(0) 80e⟩
  public ⟨Read: queryRequestsCountCompleted(1) 81a⟩
  public ⟨Read: queryRequestsQueued(1) 81e⟩
  public ⟨Read: queryRequestsWaiting(1) 82b⟩
  public ⟨Read: queryServerDistance(2) 87b⟩
  public ⟨Read: queryServerRoute(1) 82d⟩
  public ⟨Read: queryServerRouteActive(1) 83d⟩
  public ⟨Read: queryServerRouteRemaining(2) 83b⟩
  public ⟨Read: queryServerSchedule(1) 84b⟩
  public ⟨Read: queryServerTimeOfDeparture(1) 91b⟩
  public ⟨Read: queryServersActive(1) 93c⟩
  public ⟨Read: queryServersCount(0) 94b⟩
  public ⟨Read: queryServersCountActive(1) 94d⟩
  public ⟨Read: queryServersCountAppeared(0) 95b⟩
  public ⟨Read: queryServersLocationsActive(1) 97a⟩
  public ⟨Read: queryUser(1) 74a⟩
  public ⟨Read: queryVertex(1) 70a⟩
  public ⟨Read: queryVertices(0) 70c⟩
  public ⟨Read: queryVerticesCount(0) 71a⟩
```

### Cached Read Methods

```
141a  ⟨Controller methods 140c⟩+≡                    (134a) ◁140c 141b▷
  public ⟨Read: queryMetricRequestDistanceBaseUnassignedTotal(0) 107d⟩
  public ⟨Read: queryMetricRequestDistanceDetourTotal(0) 109a⟩
  public ⟨Read: queryMetricRequestDistanceTransitTotal(0) 110a⟩
  public ⟨Read: queryMetricRequestDurationPickupTotal(0) 110e⟩
  public ⟨Read: queryMetricRequestDurationTransitTotal(0) 111d⟩
  public ⟨Read: queryMetricRequestDurationTravelTotal(0) 112d⟩
  public ⟨Read: queryMetricServerDistanceCruisingTotal(0) 102d⟩
  public ⟨Read: queryMetricServerDistanceServiceTotal(0) 103c⟩
  public ⟨Read: queryMetricServerDistanceTotal(0) 100d⟩
  public ⟨Read: queryMetricServerDurationCruisingTotal(0) 105b⟩
  public ⟨Read: queryMetricServerDurationServiceTotal(0) 105f⟩
  public ⟨Read: queryMetricServerDurationTravelTotal(0) 104c⟩
  public ⟨Read: queryMetricServiceRate(0) 98b⟩
  public ⟨Read: queryMetricUserDistanceBaseTotal(0) 99c⟩
```

### Write Methods

```
141b  ⟨Controller methods 140c⟩+≡                    (134a) ◁141a 141c▷
  public ⟨Write: insertRequest(1) 123b⟩
  public ⟨Write: insertServer(2) 124b⟩
```

### Administration

```
141c  ⟨Controller methods 140c⟩+≡                    (134a) ◁141b 142a▷
  public ⟨Admin: cacheRoadNetworkFromDB(0) 42d⟩
  public ⟨Admin: cacheUsersFromDB(0) 44a⟩
  public ⟨Admin: instanceClose(0) 41a⟩
  public ⟨Admin: instanceExport(1) 40b⟩
  public ⟨Admin: instanceInitialize(0) 39a⟩
  public ⟨Admin: instanceLoad(1) 39c⟩
  public ⟨Admin: instanceNew(0) 37b⟩
  public ⟨Admin: getClock(0) 56b⟩
  public ⟨Admin: getClockStart(0) 57a⟩
  public ⟨Admin: getClockReference(0) 57b⟩
  public ⟨Admin: getClockReferenceMs(0) 57c⟩
  public ⟨Admin: getRefCommunicator(0) 58b⟩
  public ⟨Admin: getRefStorage(0) 58c⟩
  public ⟨Admin: retrieveQueueSize(0) 58d⟩
  public ⟨Admin: retrieveHandleRequestDur(0) 58e⟩
  public ⟨Admin: retrieveRefCacheEdges(0) 59b⟩
  public ⟨Admin: retrieveRefCacheUsers(0) 59c⟩
  public ⟨Admin: retrieveRefCacheVertices(0) 59a⟩
  public ⟨Admin: forwardRefCommunicator(1) 62c⟩
  public ⟨Admin: forwardRefTraffic(1) 62b⟩
  public ⟨Admin: setClockEnd(1) 60b⟩
  public ⟨Admin: setClockReference(1) 59d⟩
  public ⟨Admin: setClockStart(1) 60a⟩
  public ⟨Admin: setQueueTimeout(1) 60c⟩
  public ⟨Admin: setRefClient(1) 61c⟩
```

### G-tree Methods

142a ⟨Controller *methods* 140c⟩+≡ (134a) ◁141c 142b▷

```
  public ⟨Gtree: gtreeClose(0) 128d⟩
  public ⟨Gtree: gtreeLoad(1) 128b⟩
```

### Special Methods

142b ⟨Controller *methods* 140c⟩+≡ (134a) ◁142a

```
  public ⟨Controller: getClockReferenceDay(0) 142c⟩
  public ⟨Controller: getClockReferenceHour(0) 142d⟩
  public ⟨Controller: getClockReferenceMinute(0) 143a⟩
  public ⟨Controller: getClockReferenceSecond(0) 143b⟩
  public ⟨Controller: getQueryDur(0) 143c⟩
  public ⟨Controller: loadProblem(1) 144d⟩
  public ⟨Controller: loadRoadNetworkFromFile(1) 143d⟩
  public ⟨Controller: isKilled(0) 145a⟩
  public ⟨Controller: returnRequest(1) 145b⟩
  public ⟨Controller: startRealtime(1) 145c⟩
  public ⟨Controller: startSequential(1) 146a⟩
  public ⟨Controller: step(0) 146b⟩
  public ⟨Controller: stop(1) 146c⟩
```

### getClockReferenceDay(0)

142c ⟨*Controller: getClockReferenceDay(0)* 142c⟩≡ (142b)

```
  int getClockReferenceDay() {
    return this.simClockReferenceDay;
  }
```

### getClockReferenceHour(0)

142d ⟨*Controller: getClockReferenceHour(0)* 142d⟩≡ (142b)

```
  int getClockReferenceHour() {
    return this.simClockReferenceHour;
  }
```

## getClockReferenceMinute(0)

143a ⟨*Controller: getClockReferenceMinute(0)* 143a⟩≡ (142b)
```
  int getClockReferenceMinute() {
    return this.simClockReferenceMinute;
  }
```

## getClockReferenceSecond(0)

143b ⟨*Controller: getClockReferenceSecond(0)* 143b⟩≡ (142b)
```
  int getClockReferenceSecond() {
    return this.simClockReferenceSecond;
  }
```

## getQueryDur(0)

143c ⟨*Controller: getQueryDur(0)* 143c⟩≡ (142b)
```
  long getQueryDur() {
    return this.dur_query;
  }
```

## loadRoadNetworkFromFile(1)

143d ⟨*Controller: loadRoadNetworkFromFile(1)* 143d⟩≡ (142b) 143e ▷
```
  void loadRoadNetworkFromFile(final String f_rnet) throws FileNotFoundException, SQLException {
    if (DEBUG) {
      System.out.printf("loadRoadNetworkFileFile(1), arg1=%s\n", f_rnet);
    }
    Scanner sc = new Scanner(new File(f_rnet));
    while (sc.hasNext()) {
```
Defines:
`loadRoadNetworkFromFile`, used in chunks 167b and 194.

If a vertex identifier is 0, then we store its coordinates as $(0, 0)$. We still call `Scanner.nextDouble`(0) because we need to advance to the next token.

143e ⟨*Controller: loadRoadNetworkFromFile(1)* 143d⟩+≡ (142b) ◁ 143d 143f ▷
```
  final int col0 = sc.nextInt();
  final int col1 = sc.nextInt();
  final int col2 = sc.nextInt();
  final int col3 = (col1 == 0 ? (int) (0*sc.nextDouble()) : (int) Math.round(sc.nextDouble()*CSHIFT));
  final int col4 = (col1 == 0 ? (int) (0*sc.nextDouble()) : (int) Math.round(sc.nextDouble()*CSHIFT));
  final int col5 = (col2 == 0 ? (int) (0*sc.nextDouble()) : (int) Math.round(sc.nextDouble()*CSHIFT));
  final int col6 = (col2 == 0 ? (int) (0*sc.nextDouble()) : (int) Math.round(sc.nextDouble()*CSHIFT));
```

Now we insert the vertices into the database.

143f ⟨*Controller: loadRoadNetworkFromFile(1)* 143d⟩+≡ (142b) ◁ 143e 144a ▷
```
  try {
    this.storage.DBInsertVertex(col1, col3, col4);
  } catch (DuplicateVertexException e) {
    if (DEBUG) {
      // System.err.println("Warning! Duplicate vertex ignored.");
    }
  }
  try {
    this.storage.DBInsertVertex(col2, col5, col6);
  } catch (DuplicateVertexException e) {
    if (DEBUG) {
      // System.err.println("Warning! Duplicate vertex ignored.");
    }
  }
```
Uses `DBInsertVertex` 120a and `DuplicateVertexException` 63c.

We use haversine to compute edge weights[1]. If one of the vertices in the edge is a dummy vertex, we set the weight to 0[2].

144a ⟨*Controller: loadRoadNetworkFromFile(1)* 143d⟩+≡ (142b) ◁143f 144b▷
```
  final int dist = ((col1 != 0 && col2 != 0)
    ? this.tools.computeHaversine(
          col3/CSHIFT, col4/CSHIFT,
          col5/CSHIFT, col6/CSHIFT) : 0);
```
Uses `computeHaversine` 159b.

The fifth parameter is the *initial speed* on all the edges [3].

144b ⟨*Controller: loadRoadNetworkFromFile(1)* 143d⟩+≡ (142b) ◁144a 144c▷
```
  try {
    this.storage.DBInsertEdge(col1, col2, dist, 10);
  } catch (DuplicateEdgeException e) {
    if (DEBUG) {
      // System.err.println("Warning! Duplicate edge ignored.");
    }
  }
```
Uses `DBInsertEdge` 121 and `DuplicateEdgeException` 63a.

Now we close the `while` loop and register the caches to Tools.

144c ⟨*Controller: loadRoadNetworkFromFile(1)* 143d⟩+≡ (142b) ◁144b
```
    }
    this.tools.setRefCacheVertices(this.storage.getRefCacheVertices());
    this.tools.setRefCacheEdges(this.storage.getRefCacheEdges());
  }
```
Uses `getRefCacheEdges` 57d, `getRefCacheVertices` 58a, `setRefCacheEdges` 61a, and `setRefCacheVertices` 60e.

## `loadProblem(1)`

144d ⟨*Controller: loadProblem(1)* 144d⟩≡ (142b)
```
  void loadProblem(String p)
  throws FileNotFoundException, DuplicateUserException, EdgeNotFoundException, SQLException,
         GtreeNotLoadedException, GtreeIllegalSourceException, GtreeIllegalTargetException {
    Scanner sc = new Scanner(new File(p));
    ⟨Read header rows 140b⟩
    if (DEBUG) {
      System.out.printf("loadProblem(1), arg1=%s\n", p);
      System.out.printf("Set reference string '%s'\n", this.refTimeStr);
      System.out.printf("Set reference milliseconds from midnight '%d'\n", this.refTimeMs);
    }
    while (sc.hasNext()) {
      final int uid = sc.nextInt();
      final int  uo = sc.nextInt();
      final int  ud = sc.nextInt();
      final int  uq = sc.nextInt();
      final int  ue = sc.nextInt();
      final int  ul = sc.nextInt();
      final int  ub = this.tools.computeShortestPathDistance(uo, ud);
      if (uq < 0) {
        this.insertServer(new int[] { uid, uq, ue, ul, uo, ud, ub });
      } else {
        this.insertRequest(new int[] { uid, uq, ue, ul, uo, ud, ub });
      }
    }
  }
```
Defines:
`loadProblem`, used in chunks 167b and 195.

[1]If the distance between two vertices is 0 due to rounding, then we round it up to 1.
[2]The dummy vertex should only terminate and never begin an edge in the road network, otherwise a shortest path could take a shortcut through the dummy vertex to reach any other vertex with 0 weight!
[3]In the future, the speed on each edge may be recorded directly in the road network file instead of hardcoded here.

Uses computeShortestPathDistance 161a, DuplicateUserException 63b, EdgeNotFoundException 63d, GtreeIllegalSourceException 63e, GtreeIllegalTargetException 64a, GtreeNotLoadedException 64b, insertRequest 123b, and insertServer 124b.

## isKilled(0)

145a ⟨*Controller: isKilled(0)* 145a⟩≡ (142b)

```
final boolean isKilled() {
  return this.kill;
}
```

Defines:
isKilled, never used.

## returnRequest(1)

145b ⟨*Controller: returnRequest(1)* 145b⟩≡ (142b)

```
void returnRequest(final int[] r) {
  if (this.simClock - r[2] < QUEUE_TIMEOUT) {
    this.lu_rseen.put(r[0], false);
  }
}
```

Defines:
returnRequest, used in chunk 149e.

## startRealtime(1)

145c ⟨*Controller: startRealtime(1)* 145c⟩≡ (142b)

```
void startRealtime(final Consumer<Boolean> app_cb) {
  if (DEBUG) {
    System.out.printf("startRealtime(1)\n");
  }

  this.storage.setRequestTimeout(REQUEST_TIMEOUT);
  if (DEBUG) {
    System.out.printf("setRequestTimeout(1), arg1=%d\n", REQUEST_TIMEOUT);
  }

  this.simClock = CLOCK_START;
  if (DEBUG) {
    System.out.printf("simClock=%d\n", CLOCK_START);
  }

  int simulation_duration = (CLOCK_END - CLOCK_START);
  if (DEBUG) {
    System.out.printf("simulation_duration=%d\n", simulation_duration);
  }

  this.exe = Executors.newScheduledThreadPool(5);
  if (DEBUG) {
    System.out.printf("newScheduledThreadPool(1), arg1=5\n");
  }

  this.cb1 = exe.scheduleAtFixedRate(
    this.ClockLoop, 0, 1, TimeUnit.SECONDS);
  if (DEBUG) {
    System.out.printf("exe ClockLoop, delay=0, int=1\n");
  }

  this.cb2 = exe.scheduleAtFixedRate(
    this.RequestCollectionLoop, this.loop_delay, REQUEST_COLLECTION_PERIOD, TimeUnit.SECONDS);
  if (DEBUG) {
    System.out.printf("exe RequestCollectionLoop, delay=%d, int=%d\n",
        this.loop_delay, REQUEST_COLLECTION_PERIOD);
  }

  this.cb3 = exe.scheduleAtFixedRate(
    this.RequestHandlingLoop, this.loop_delay, REQUEST_HANDLING_PERIOD, TimeUnit.MILLISECONDS);
  if (DEBUG) {
    System.out.printf("exe RequestHandlingLoop, delay=%d, int=%d\n",
        this.loop_delay, REQUEST_HANDLING_PERIOD);
  }

  this.cb4 = exe.scheduleAtFixedRate(
    this.ServerLoop, this.loop_delay, SERVER_COLLECTION_PERIOD, TimeUnit.SECONDS);
  if (DEBUG) {
    System.out.printf("exe ServerLoop, delay=%d, int=%d\n",
        this.loop_delay, SERVER_COLLECTION_PERIOD);
  }

  this.exe.schedule(() -> {
    this.stop(app_cb);
  }, simulation_duration, TimeUnit.SECONDS);
  if (DEBUG) {
    System.out.printf("exe stop, delay=%d\n",
        simulation_duration);
  }
}
```

Defines:
  startRealtime, used in chunks 167b and 202b.
Uses setRequestTimeout 60d and stop 146c.

### startSequential(1)

146a ⟨*Controller: startSequential(1)* 146a⟩≡ (142b)

```
  void startSequential(final Consumer<Boolean> app_cb) throws Exception {
    if (DEBUG) {
      System.out.printf("startSequuential(1)\n");
    }
    this.storage.setRequestTimeout(REQUEST_TIMEOUT);
    this.simClock = CLOCK_START;
    while (!kill && this.simClock < CLOCK_END) {
      this.working = true;
      this.step();
      this.working = false;
    }
    this.stop(app_cb);
  }
```

Defines:
  startSequential, used in chunks 167b and 202a.
Uses setRequestTimeout 60d and stop 146c.

### step(0)

146b ⟨*Controller: step(0)* 146b⟩≡ (142b)

```
  void step() {
    this.ClockLoop.run();
    this.ServerLoop.run();
    this.RequestCollectionLoop.run();
    this.RequestHandlingLoop.run();
  }
```

### stop(1)

146c ⟨*Controller: stop(1)* 146c⟩≡ (142b)

```
  void stop(final Consumer<Boolean> app_cb) {
    if (DEBUG) {
      System.out.printf("stop(1)\n");
    }
    if (this.exe == null) {  // sequential mode
      this.kill = true;
      while (this.working) {
        try {
          Thread.sleep(100);
        } catch (InterruptedException e) {
          // ...
        }
      }
    } else {  // realtime mode
      this.cb1.cancel(true);
      this.cb2.cancel(true);
      this.cb3.cancel(true);
      this.cb4.cancel(true);
      this.exe.shutdown();
    }
    try {
      if (this.client != null) {
        this.client.end();
      }
      app_cb.accept(true);
    } catch (Exception e) {
      System.err.println("Error in ending callback");
      System.err.println(e.toString());
      e.printStackTrace();
      return;
    }
  }
```

Defines:
  `stop`, used in chunks 145c, 146a, and 203.
Uses `end` 152d.

## 3.8 Class: Communicator

148a ⟨*Communicator.java* 148a⟩≡
```
  ⟨Package: sim 35a⟩
  ⟨Communicator.java preamble 148b⟩
  public class Communicator {
    ⟨Communicator member variables 148c⟩
    ⟨Communicator constructor 148d⟩
    ⟨Communicator methods 149a⟩
  }
```

### 3.8.1 Preamble

148b ⟨*Communicator.java preamble* 148b⟩≡ (148a)
```
  import com.github.jargors.sim.Storage;
  import com.github.jargors.sim.Controller;
  import com.github.jargors.sim.Traffic;
  import com.github.jargors.sim.ClientException;
  import com.github.jargors.sim.ClientFatalException;
  import com.github.jargors.sim.DuplicateVertexException;
  import com.github.jargors.sim.DuplicateEdgeException;
  import com.github.jargors.sim.DuplicateUserException;
  import com.github.jargors.sim.EdgeNotFoundException;
  import com.github.jargors.sim.UserNotFoundException;
  import com.github.jargors.sim.VertexNotFoundException;
  import com.github.jargors.sim.RouteIllegalOverwriteException;
  import com.github.jargors.sim.TimeWindowException;
  import java.util.Map;
  import java.util.concurrent.ConcurrentHashMap;
  import java.sql.SQLException;
```
Uses DuplicateEdgeException 63a, DuplicateUserException 63b, DuplicateVertexException 63c, EdgeNotFoundException 63d, RouteIllegalOverwriteException 64c, TimeWindowException 64d, UserNotFoundException 65a, and VertexNotFoundException 65b.

### 3.8.2 Member Variables

148c ⟨Communicator *member variables* 148c⟩≡ (148a)
```
  private Storage storage;
  private Controller controller;
  private Traffic traffic = null;
  private final boolean DEBUG = "true".equals(System.getProperty("jargors.communicator.debug"));
```

### 3.8.3 Constructor

148d ⟨Communicator *constructor* 148d⟩≡ (148a)
```
  public Communicator() { }
```

### 3.8.4 Methods

#### Read Methods

149a ⟨Communicator *methods* 149a⟩≡ (148a) 149b ▷

```
  public ⟨Read: queryEdge(2) 71c⟩
  public ⟨Read: queryServerCapacityViolations(4) 86c⟩
  public ⟨Read: queryServerDistanceRemaining(2) 88a⟩
  public ⟨Read: queryServerDurationRemaining(2) 89b⟩
  public ⟨Read: queryServerDurationTravel(2) 90b⟩
  public ⟨Read: queryServerLoadMax(2) 86a⟩
  public ⟨Read: queryServerRouteActive(1) 83d⟩
  public ⟨Read: queryServerRouteRemaining(2) 83b⟩
  public ⟨Read: queryServerScheduleRemaining(2) 85a⟩
  public ⟨Read: queryServersLocationsActive(1) 97a⟩
  public ⟨Read: queryUser(1) 74a⟩
  public ⟨Read: queryVertex(1) 70a⟩
```

#### Write Methods

149b ⟨Communicator *methods* 149a⟩+≡ (148a) ◁ 149a 149c ▷

```
  public ⟨Write: updateServerService(5) 125b⟩
```

#### Adminstration

149c ⟨Communicator *methods* 149a⟩+≡ (148a) ◁ 149b 149d ▷

```
  public ⟨Admin: retrieveClock(0) 58f⟩
  public ⟨Admin: retrieveRefCacheEdges(0) 59b⟩
  public ⟨Admin: retrieveRefCacheUsers(0) 59c⟩
  public ⟨Admin: retrieveRefCacheVertices(0) 59a⟩
  public ⟨Admin: setRefController(1) 61e⟩
  public ⟨Admin: setRefStorage(1) 61f⟩
  public ⟨Admin: setRefTraffic(1) 62a⟩
```

#### Special Methods

149d ⟨Communicator *methods* 149a⟩+≡ (148a) ◁ 149c

```
  public ⟨Communicator: forwardReturnRequest(1) 149e⟩
```

#### forwardReturnRequest(1)

149e ⟨*Communicator: forwardReturnRequest(1)* 149e⟩≡ (149d)

```
  void forwardReturnRequest(final int[] r) {
    this.controller.returnRequest(r);
  }
```

Defines:
`forwardReturnRequest`, never used.
Uses `returnRequest` 145b.

## 3.9 Class: Client

150a ⟨*Client.java* 150a⟩≡
```
  ⟨Package: sim 35a⟩
  ⟨Client.java preamble 150b⟩
  public abstract class Client {
    ⟨Client member variables 150c⟩
    ⟨Client constructor 151a⟩
    ⟨Client methods 151b⟩
  }
```

### 3.9.1 Preamble

150b ⟨*Client.java preamble* 150b⟩≡ (150a)
```
  import com.github.jargors.sim.Communicator;
  import com.github.jargors.sim.Tools;
  import com.github.jargors.sim.ClientException;
  import com.github.jargors.sim.ClientFatalException;
  import java.util.concurrent.ConcurrentLinkedQueue;
  import java.util.concurrent.ConcurrentHashMap;
  import java.io.FileNotFoundException;
```

### 3.9.2 Member Variables

150c ⟨Client *member variables* 150c⟩≡ (150a)
```
  protected ConcurrentLinkedQueue<int[]> queue = new ConcurrentLinkedQueue<int[]>();
  protected Communicator communicator;
  protected Tools tools = new Tools();
  protected final boolean DEBUG =
      "true".equals(System.getProperty("jargors.client.debug"));
  protected ConcurrentHashMap<Integer, Integer> lut = new ConcurrentHashMap<Integer, Integer>();
  protected ConcurrentHashMap<Integer, Integer> luv = new ConcurrentHashMap<Integer, Integer>();
  protected long dur_handle_request = 0;
```

### 3.9.3 Constructor

151a ⟨Client *constructor* 151a⟩≡ (150a)
```
  public Client() {
    if (DEBUG) {
      System.out.printf("create Client\n");
    }
  }
```

### 3.9.4 Methods

#### Administration

151b ⟨Client *methods* 151b⟩≡ (150a) 151c ▷
```
  public ⟨Admin: forwardRefCacheEdges(1) 62d⟩
  public ⟨Admin: forwardRefCacheUsers(1) 62e⟩
  public ⟨Admin: forwardRefCacheVertices(1) 62f⟩
  public ⟨Admin: setRefCommunicator(1) 61d⟩
```

#### G-tree Methods

151c ⟨Client *methods* 151b⟩+≡ (150a) ◁ 151b 151d ▷
```
  public ⟨Gtree: gtreeLoad(1) 128b⟩
  public ⟨Gtree: gtreeClose(0) 128d⟩
```

#### Special Methods

151d ⟨Client *methods* 151b⟩+≡ (150a) ◁ 151c
```
  public ⟨Client: addRequest(1) 152a⟩
  public ⟨Client: collectServerLocations(1) 152c⟩
  public ⟨Client: dropRequests(1) 152b⟩
  public ⟨Client: getQueueSize(0) 152f⟩
  public ⟨Client: getHandleRequestDur(0) 152e⟩
  public ⟨Client: init(0) 153c⟩
  public ⟨Client: notifyNew(0) 151e⟩
  protected ⟨Client: end(0) 152d⟩
  protected ⟨Client: handleRequest(1) 153a⟩
  protected ⟨Client: handleServerLocation(1) 153b⟩
```

#### notifyNew(0)

151e ⟨*Client: notifyNew(0)* 151e⟩≡ (151d)
```
  void notifyNew() throws ClientException, ClientFatalException {
    while (!this.queue.isEmpty()) {
      long A0 = System.currentTimeMillis();
      this.handleRequest(this.queue.remove());
      this.dur_handle_request = System.currentTimeMillis() - A0;
      if (DEBUG) {
        System.out.printf("handleRequest(1), arg1=[#]\n");
      }
    }
  }
```
Defines:
  `notifyNew`, used in chunk 138.
Uses `handleRequest` 153a.

### addRequest(1)

152a ⟨*Client: addRequest(1)* 152a⟩≡ (151d)
```
  void addRequest(final int[] r) {
    this.queue.add(r);
  }
```
Defines:
  addRequest, used in chunk 137.

### dropRequests(1)

152b ⟨*Client: dropRequests(1)* 152b⟩≡ (151d)
```
  int dropRequests(final int deadline) {
    final int temp = this.queue.size();
    this.queue.removeIf((r) -> { return r[2] < deadline; });
    return Math.max(0, temp - this.queue.size());
  }
```
Defines:
  dropRequests, used in chunk 137.

### collectServerLocations(1)

Array src =

| | | | |
|---|---|---|---|
| 0 : sid of server $s$ | 1 : time of $s$'s last location | 2 : vertex of $s$'s last location | ⋯ |

152c ⟨*Client: collectServerLocations(1)* 152c⟩≡ (151d)
```
  void collectServerLocations(final int[] src) {
    for (int i = 0; i < (src.length - 2); i += 3) {
      this.handleServerLocation(new int[] {
        src[i],
        src[(i + 1)],
        src[(i + 2)]
      });
    }
  }
```
Defines:
  collectServerLocations, used in chunk 139.
Uses handleServerLocation 153b.

### end(0)

152d ⟨*Client: end(0)* 152d⟩≡ (151d)
```
  void end() { }
```
Defines:
  end, used in chunks 146c and 165–67.

### getHandleRequestDur(0)

152e ⟨*Client: getHandleRequestDur(0)* 152e⟩≡ (151d)
```
  long getHandleRequestDur() {
    return this.dur_handle_request;
  }
```

### getQueueSize(0)

152f ⟨*Client: getQueueSize(0)* 152f⟩≡ (151d)
```
  int getQueueSize() {
    return this.queue.size();
  }
```
Defines:
  getQueueSize, used in chunk 58d.

`handleRequest(1)`

Array `r` =

| 0 : `rid` of request $r$ | 1 : $r_q$ | 2 : $r_e$ | 3 : $r_l$ | 4 : $r_o$ | 5 : $r_d$ | 6 : $d_r$ |
|---|---|---|---|---|---|---|

153a ⟨*Client: handleRequest(1)* 153a⟩≡ (151d)
```
  void handleRequest(final int[] r) throws ClientException, ClientFatalException { }
```
Defines:
`handleRequest`, used in chunk 151e.

`handleServerLocation(1)`

Array `loc` =

| 0 : `sid` of server $s$ | 1 : time of $s$'s last location | 2 : vertex of $s$'s last location |
|---|---|---|

153b ⟨*Client: handleServerLocation(1)* 153b⟩≡ (151d)
```
  void handleServerLocation(final int[] loc) {
    lut.put(loc[0], loc[1]);
    luv.put(loc[0], loc[2]);
  }
```
Defines:
`handleServerLocation`, used in chunk 152c.

`init(0)`

153c ⟨*Client: init(0)* 153c⟩≡ (151d)
```
  void init() { }
```

### 3.9.5 Exceptions

`ClientException`

153d ⟨*ClientException.java* 153d⟩≡
```
  ⟨Package: sim 35a⟩
  public class ClientException extends Exception {
    public ClientException() { }
    public ClientException(String message) {
      super(message);
    }
    public ClientException(Throwable cause) {
      super(cause);
    }
    public ClientException(String message, Throwable cause) {
      super(message, cause);
    }
  }
```

`ClientFatalException`

153e ⟨*ClientFatalException.java* 153e⟩≡
```
  ⟨Package: sim 35a⟩
  public class ClientFatalException extends Exception {
    public ClientFatalException() { }
    public ClientFatalException(String message) {
      super(message);
    }
    public ClientFatalException(Throwable cause) {
      super(cause);
    }
    public ClientFatalException(String message, Throwable cause) {
      super(message, cause);
    }
  }
```

### 3.9.6 Debug

## 3.10 Class: Traffic

```
155a  ⟨Traffic.java 155a⟩≡
        ⟨Package: sim 35a⟩
        ⟨Traffic.java preamble 155b⟩
        public abstract class Traffic {
          ⟨Traffic member variables 155c⟩
          ⟨Traffic constructor 155d⟩
          ⟨Traffic methods 155e⟩
        }
```

### 3.10.1 Preamble

The preamble declares the package and imports dependencies.

```
155b  ⟨Traffic.java preamble 155b⟩≡                                    (155a)
        import com.github.jargors.sim.Tools;
        import java.util.Map;
        import java.util.HashMap;
        import java.util.concurrent.ConcurrentHashMap;
```

### 3.10.2 Member Variables

```
155c  ⟨Traffic member variables 155c⟩≡                                  (155a)
        protected Tools tools = new Tools();
        protected final boolean DEBUG = "true".equals(System.getProperty("jargors.traffic.debug"));
```

### 3.10.3 Constructor

```
155d  ⟨Traffic constructor 155d⟩≡                                       (155a)
        public Traffic() {
          if (DEBUG) {
            System.out.printf("create Traffic\n");
          }
        }
```

### 3.10.4 Methods

### 3.10.5 Traffic

#### Administration

```
155e  ⟨Traffic methods 155e⟩≡                                 (155a)  155f▷
        public ⟨Admin: forwardRefCacheEdges(1) 62d⟩
        public ⟨Admin: forwardRefCacheVertices(1) 62f⟩
```

#### Special Methods

```
155f  ⟨Traffic methods 155e⟩+≡                                (155a)  ◁155e
        public ⟨Traffic: apply(3) 156a⟩
        public ⟨Traffic: init(0) 156b⟩
```

### apply(3)

Param 3 is seconds since start of day (12 AM midnight).

156a ⟨*Traffic: apply(3)* 156a⟩≡ (155f)
```
double apply(int v1, int v2, long t) {
  return 1.0;
}
```
Defines:
`apply`, used in chunks 114 and 178.

### init(0)

156b ⟨*Traffic: init(0)* 156b⟩≡ (155f)
```
void init() { }
```

## 3.11 Class: Tools

157a ⟨*Tools.java* 157a⟩≡
```
⟨Package: sim 35a⟩
⟨Tools.java preamble 157b⟩
public class Tools {
  ⟨Tools member variables 158a⟩
  ⟨Tools constructor 158b⟩
  ⟨Tools methods 158c⟩
}
```

### 3.11.1 Preamble

The preamble declares the package and imports dependencies.

157b ⟨*Tools.java preamble* 157b⟩≡ (157a)
```
import com.github.jargors.sim.EdgeNotFoundException;
import com.github.jargors.sim.VertexNotFoundException;
import com.github.jargors.sim.GtreeNotLoadedException;
import com.github.jargors.sim.GtreeIllegalSourceException;
import com.github.jargors.sim.GtreeIllegalTargetException;
import com.github.jamjpan.gtree.GTree;
import com.github.jamjpan.gtree.gtreeJNI;
import com.github.jamjpan.gtree.IntVector;
import java.io.FileNotFoundException;
import java.sql.SQLException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDateTime;
import java.util.Arrays;
import java.util.Date;
import java.util.concurrent.ConcurrentHashMap;
```
Uses EdgeNotFoundException 63d, GtreeIllegalSourceException 63e, GtreeIllegalTargetException 64a, GtreeNotLoadedException 64b, and VertexNotFoundException 65b.

### 3.11.2 Member Variables

158a ⟨Tools *member variables* 158a⟩≡ (157a)
```
  private GTree gtree;
  private boolean flag_gtree_loaded = false;
  private ConcurrentHashMap<Integer, int[]> lu_vertices = new ConcurrentHashMap<Integer, int[]>();
  private ConcurrentHashMap<Integer,
      ConcurrentHashMap<Integer, int[]>>    lu_edges    = new ConcurrentHashMap<Integer, ConcurrentHashMap<Inte
  private ConcurrentHashMap<Integer, int[]> lu_users    = new ConcurrentHashMap<Integer, int[]>();
  private int[] bbox = new int[] { };
  private final double CSHIFT = Storage.CSHIFT;
  private final boolean DEBUG = "true".equals(System.getProperty("jargors.tools.debug"));
```

### 3.11.3 Constructor

158b ⟨Tools *constructor* 158b⟩≡ (157a)
```
  public Tools() {
    if (DEBUG) {
      System.out.printf("create Tools\n");
    }
  }
```

### 3.11.4 Methods

#### Read Methods

158c ⟨Tools *methods* 158c⟩≡ (157a) 158d ▷
```
  public ⟨Read: DBQueryEdge(2) 71b⟩
  public ⟨Read: DBQueryVertex(1) 69c⟩
```

#### Administration

158d ⟨Tools *methods* 158c⟩+≡ (157a) ◁ 158c 158e ▷
```
  public ⟨Admin: setRefCacheEdges(1) 61a⟩
  public ⟨Admin: setRefCacheUsers(1) 61b⟩
  public ⟨Admin: setRefCacheVertices(1) 60e⟩
```

#### G-tree Methods

158e ⟨Tools *methods* 158c⟩+≡ (157a) ◁ 158d 158f ▷
```
  public ⟨Gtree: GTGtreeLoad(1) 128a⟩
  public ⟨Gtree: GTGtreeClose(0) 128c⟩
```

#### Special Methods

158f ⟨Tools *methods* 158c⟩+≡ (157a) ◁ 158e
```
  public ⟨Tools: computeBoundingBox(0) 159a⟩
  public ⟨Tools: computeDuration(2) 160a⟩
  public ⟨Tools: computeHaversine(2) 159d⟩
  public ⟨Tools: computeHaversine(4) 159b⟩
  public ⟨Tools: computeRoute(3) 161b⟩
  public ⟨Tools: computeShortestPath(2) 160b⟩
  public ⟨Tools: computeShortestPathDistance(2) 161a⟩
  public ⟨Tools: filterByHaversine(3) 162a⟩
  public ⟨Tools: parseClockReference(1) 162b⟩
  public ⟨Tools: printPath(1) 162d⟩
  public ⟨Tools: printRoute(1) 163a⟩
  public ⟨Tools: printSchedule(1) 163b⟩
  public ⟨Tools: printUser(1) 162c⟩
  public static ⟨Tools: Print(1) 163d⟩
  public static ⟨Tools: Print(2) 163c⟩
  public static ⟨Tools: PrintSQLException(1) 163e⟩
```

### computeBoundingBox(0)

Even though there is already DBQueryMBR(0), the Tools class doesn't have access to it. So compute-BoundingBox(0) is kind of a filler method.

159a ⟨*Tools: computeBoundingBox(0)* 159a⟩≡ (158f)
```
int[] computeBoundingBox() {
  if (this.bbox.length == 0) {
    int x_min = Integer.MAX_VALUE;
    int y_min = Integer.MAX_VALUE;
    int x_max = Integer.MIN_VALUE;
    int y_max = Integer.MIN_VALUE;
    for (int i : this.lu_vertices.keySet()) {
      if (i == 0) {
        continue;
      }
      final int[] coord = this.lu_vertices.get(i);
      x_min = Math.min(x_min, coord[0]);
      y_min = Math.min(y_min, coord[1]);
      x_max = Math.max(x_max, coord[0]);
      y_max = Math.max(y_max, coord[1]);
    }
    this.bbox = new int[] { x_min, x_max, y_min, y_max };
  }
  return this.bbox.clone();
}
```
Defines:
`computeBoundingBox`, never used.

### computeHaversine(4)

159b ⟨*Tools: computeHaversine(4)* 159b⟩≡ (158f)
```
int computeHaversine(
    final double lng1, final double lat1, final double lng2, final double lat2) {
  final double  dlat = Math.toRadians((lat2 - lat1));
  final double  dlng = Math.toRadians((lng2 - lng1));
  final double rlat1 = Math.toRadians(lat1);
  final double rlat2 = Math.toRadians(lat2);
  final double a = Math.pow(Math.sin((dlat/2)), 2)
    + (Math.pow(Math.sin((dlng/2)), 2)*Math.cos(rlat1)*Math.cos(rlat2));
  final double c = 2*Math.asin(Math.sqrt(a));
  int d = (int) Math.round(c*6371000);
  ⟨..round to nearest meter 159c⟩
  return d;
}
```
Defines:
`computeHaversine`, used in chunks 144a, 159d, and 162a.

If the nearest meter is 0 and the two points are not equal, then the distance is rounded up to 1 meter.

159c ⟨*..round to nearest meter* 159c⟩≡ (159b)
```
if (d == 0 && (lng1 != lng2 || lat1 != lat2)) {
  d = 1;
}
```

### computeHaversine(2)

159d ⟨*Tools: computeHaversine(2)* 159d⟩≡ (158f)
```
int computeHaversine(final int u, final int v) throws VertexNotFoundException {
  if (!this.lu_vertices.containsKey(u)) {
    throw new VertexNotFoundException("Vertex "+u+" not found.");
  }
  if (!this.lu_vertices.containsKey(v)) {
    throw new VertexNotFoundException("Vertex "+v+" not found.");
  }
```

```
    return this.computeHaversine(
      this.lu_vertices.get(u)[0]/CSHIFT, this.lu_vertices.get(u)[1]/CSHIFT,
      this.lu_vertices.get(v)[0]/CSHIFT, this.lu_vertices.get(v)[1]/CSHIFT);
  }
```

Uses `computeHaversine` 159b and `VertexNotFoundException` 65b.

### `computeDuration`(2)

160a ⟨*Tools: computeDuration(2)* 160a⟩≡ (158f)

```
  int computeDuration(final int dd, final int nu) {
    int d = (int) Math.ceil(dd/(float) nu);
    return (d == 0 ? 1 : d);
  }
```

### `computeShortestPath`(2)

Beware, two important notes:

- The vertices in G-tree are 0-indexed while they are 1-indexed in Jargo. To compensate, at location `L1` we subtract 1 from `u` and `v`, and at location `L2` we add 1 to the vertices returned in the path.
- We consider vertex 0 to be a dummy vertex. The path to this vertex from any other vertex $v$ is always $\{v, 0\}$. The path from 0 to any other vertex is undefined and throws a `RuntimeException`.

160b ⟨*Tools: computeShortestPath(2)* 160b⟩≡ (158f)

```
  int[] computeShortestPath(final int u, final int v)
  throws GtreeNotLoadedException, GtreeIllegalSourceException {
    int[] output = null;
    if (!this.flag_gtree_loaded) {
      throw new GtreeNotLoadedException("Gtree not loaded!");
    } else if (u == 0) {
      throw new GtreeIllegalSourceException("Source cannot be 0!");
    } else if (v == 0) {
      output = new int[] { u, v };
    } else if (u == v) {
      output = new int[] { u };
    } else {
      IntVector path = new IntVector();
      gtree.shortest_path_querying((u - 1), (v - 1)); // L1
      gtree.path_recovery((u - 1), (v - 1), path);
      if (path != null) {
        output = new int[path.size()];
        for (int i = 0; i < path.size(); i++) {
          output[i] = path.get(i) + 1;                // L2
        }
      }
    }
    return output;
  }
```

Defines:
  `computeShortestPath`, used in chunk 161b.
Uses `GtreeIllegalSourceException` 63e and `GtreeNotLoadedException` 64b.

### `computeShortestPathDistance`(2)

Beware, two important notes:

- The vertices in G-tree are 0-indexed while they are 1-indexed in Jargo. To compensate, at we subtract 1 from `u` and `v` when calling `gtree.search`(2).
- We consider vertex 0 to be a dummy vertex. The distance to this vertex is always 0.

161a ⟨*Tools: computeShortestPathDistance(2)* 161a⟩≡ (158f)
```
int computeShortestPathDistance(final int u, final int v)
throws GtreeNotLoadedException, GtreeIllegalSourceException {
  int d = 0;
  if (!this.flag_gtree_loaded) {
    throw new GtreeNotLoadedException("GTree not loaded!");
  } else if (u == 0) {
    throw new GtreeIllegalSourceException("Source cannot be 0!");
  } else if (u != v && v != 0) {
    d = gtree.shortest_path_querying((u - 1), (v - 1));
  }
  return d;
}
```
Defines:
  `computeShortestPathDistance`, used in chunk 144d.
Uses `GtreeIllegalSourceException` 63e and `GtreeNotLoadedException` 64b.

### `computeRoute`(3)

161b ⟨*Tools: computeRoute(3)* 161b⟩≡ (158f)
```
int[] computeRoute(final int source, final int target, final int starttime)
throws GtreeNotLoadedException, GtreeIllegalSourceException, GtreeIllegalTargetException {
  int[] output = null;
  if (source == 0) {
    throw new GtreeIllegalSourceException("Source cannot be 0!");
  } else if (target == 0) {
    output = new int[] { starttime, source, starttime + 1, target };
  } else {
    int[] path = this.computeShortestPath(source, target);
    if (path == null) {
      throw new GtreeIllegalTargetException("No path from source to target!");
    } else {
      output = new int[(path.length*2)];
      output[0] = starttime;
      output[1] = source;
      int t = starttime;
      int j = 2;
      for (int i = 0; i < (path.length - 1); i++) {
        int u = path[(i + 0)];
        int v = path[(i + 1)];
        int[] edge = this.lu_edges.get(u).get(v);
        output[(j + 0)] = (t += computeDuration(edge[0], edge[1]));
        output[(j + 1)] = v;
        j += 2;
      }
    }
  }
  return output;
}
```
Defines:
  `computeRoute`, used in chunk 124b.
Uses `computeShortestPath` 160b, `GtreeIllegalSourceException` 63e, `GtreeIllegalTargetException` 64a,
  and `GtreeNotLoadedException` 64b.

### `filterByHaversine`(3)

This function takes a request origin `ro`, a locations array `locs`, and a distance threshold `threshold` as input, and returns a copy of `locs` that keeps only those location triplets where the haversine distance between the location in the triplet and `ro` is within `threshold`.

Locations array `locs` =

| | | |
|---|---|---|
| 0 : `sid` of server $s_1$ | 1 : $\pi_t(w_{\lvert w_{\leq t}\rvert})$ of server $s_1$ | 2 : $\pi_v(w_{\lvert w_{\leq t}\rvert})$ of server $s_1$ |
| 3 : `sid` of server $s_2$ | 4 : $\pi_t(w_{\lvert w_{\leq t}\rvert})$ of server $s_2$ | 5 : $\pi_v(w_{\lvert w_{\leq t}\rvert})$ of server $s_2$ |

... and so on,

where $\pi_t(w_{|w_{\leq t}|})$ gives the time component of a server's last-visited waypoint, and $\pi_v(w_{|w_{\leq t}|})$ gives the vertex component.

162a ⟨*Tools: filterByHaversine(3)* 162a⟩≡ (158f)
```
  int[] filterByHaversine(final int ro, final int[] locs, final int threshold)
  throws VertexNotFoundException {
    final int n = (locs.length/3);
    int[] temp = new int[n];
    int i = 0;
    for (int k = 0; k < n; k++) {
      if (this.computeHaversine(ro, locs[((3*k) + 2)]) < threshold) {
        temp[i++] = 3*k;
      }
    }
    return Arrays.copyOf(temp, i);
  }
```
Defines:
  `filterByHaversine`, never used.
Uses `computeHaversine` 159b and `VertexNotFoundException` 65b.

## `parseClockReference`(1)

Given a time string in "hhmm" format, returns milliseconds since Epoch. We force the year to 1971 so the return is always positive value. Due to timezone, if left default 1970, the return value can be negative if timezone is China and time is early in the morning.

162b ⟨*Tools: parseClockReference(1)* 162b⟩≡ (158f)
```
  long parseClockReference (final String refTimeStr) throws ParseException {
    SimpleDateFormat sdf = new SimpleDateFormat("HHmmyyyy");
    Date d = sdf.parse(refTimeStr+"1971");
    System.out.printf("hr=%d\n", d.getHours());
    System.out.printf("mn=%d\n", d.getMinutes());
    return d.getHours()*60*60*1000 + d.getMinutes()*60*1000;
  }
```

## `printUser`(1)

162c ⟨*Tools: printUser(1)* 162c⟩≡ (158f)
```
  void printUser(final int[] u) {
    System.out.println("User {uid="+u[0]+", q="+u[1]+", e="+u[2]+", l="+u[3]
      +", o="+u[4]+", d="+u[5]+", b="+u[6]+"}");
  }
```
Defines:
  `printUser`, never used.

## `printPath`(1)

162d ⟨*Tools: printPath(1)* 162d⟩≡ (158f)
```
  void printPath(final int[] p) {
    for (Integer i : p) {
      System.out.print(i+" ");
    }
    System.out.println();
  }
```
Defines:
  `printPath`, never used.

### printRoute(1)

163a ⟨*Tools: printRoute(1)* 163a⟩≡ (158f)
```
void printRoute(final int[] w) {
  for (int i = 0; i < (w.length - 1); i += 2) {
    System.out.print("("+w[i]+", "+w[(i + 1)]+") ");
  }
  System.out.println();
}
```
Defines:
  `printRoute`, never used.

### printSchedule(1)

163b ⟨*Tools: printSchedule(1)* 163b⟩≡ (158f)
```
void printSchedule(final int[] b) {
  for (int i = 0; i < (b.length - 3); i += 4) {
    System.out.print("("+b[i]+", "+b[(i + 1)]
      + ", "+b[(i + 2)]+", "+b[(i + 3)]+") ");
  }
  System.out.println();
}
```
Defines:
  `printSchedule`, never used.

### Print(2)

163c ⟨*Tools: Print(2)* 163c⟩≡ (158f)
```
void Print(final String a, final String b) {
  System.out.println(String.format("[%s][%s] %s", a, LocalDateTime.now(), b));
}
```
Defines:
  `Print`, used in chunk 163d.

### Print(1)

163d ⟨*Tools: Print(1)* 163d⟩≡ (158f)
```
void Print(final String b) {
  System.out.println(String.format("[*][%s] %s", LocalDateTime.now(), b));
}
```
Uses `Print` 163c.

### PrintSQLException(1)

163e ⟨*Tools: PrintSQLException(1)* 163e⟩≡ (158f)
```
void PrintSQLException(SQLException e) {
  while (e != null) {
    System.err.println("\n----- SQLException -----");
    System.err.println("  SQL State:  " + e.getSQLState());
    System.err.println("  Error Code: " + e.getErrorCode());
    System.err.println("  Message:    " + e.getMessage());
    e.printStackTrace(System.err);
    e = e.getNextException();
  }
}
```
Defines:
  `PrintSQLException`, used in chunks 193 and 205c.

# Chapter 4

# Evaluators

This chapter presents the command-line and graphical evaluators.

## 4.1 Class: Command

165a ⟨*Command.java* 165a⟩≡
```
  ⟨Package: ui 35b⟩
  ⟨Command.java preamble 165b⟩
  public class Command {
    ⟨Command methods 167a⟩
  }
```

### 4.1.1 Preamble

165b ⟨*Command.java preamble* 165b⟩≡ (165a) 165c ▷
```
  import com.github.jargors.sim.*;
```

165c ⟨*Command.java preamble* 165b⟩+≡ (165a) ◁ 165b
```
  import java.lang.reflect.Constructor;
  import java.net.URL;
  import java.net.URLClassLoader;
```

### 4.1.2 Chunks

165d ⟨*Simulation callback* 165d⟩≡ (167b)
```
  (status) -> {
    try {
      ctrl.instanceExport("export");
    } catch (Exception e) {
      System.out.println("Export failed.");
      e.printStackTrace();
    }
  }
```
Uses instanceExport 40b.

165e ⟨*Help string: usage* 165e⟩≡ (167b)
```
  String.join("\n",
    "Jargo, a real-time stochastic ridesharing simulator.",
    "Usage: ./launch-cli [OPTION...] MODE ROAD GTREE PROB CLIENT CLASSNAME",
    "",
    "Mandatory arguments:",
    "  MODE       runtime mode, either 'seq' or 'real'",
    "  ROAD       road network *.rnet file",
    "  GTREE      gtree *.gtree file to the road network",
    "  PROB       problem *.instance file (see FORMATS section)",
    "  CLIENT     client *.jar file",
    "  CLASSNAME  client classname",
    "",
    "Options:",
    "  -h       show help",
    "  -r       client *.gtree file (default: GTREE)",
    "  -x       traffic *.jar file (default: none)",
    "  -y       traffic classname (default: '')",
    "  -s       start time (see TIME section)",
    "  -e       end time (see TIME section)",
    ""
  );
```
Uses end 152d.

166 ⟨*Help string: details 166*⟩≡ (167b)
```
  String.join("\n",
    "FORMATS",
    "",
    "The ROAD file should be a plain-text file with seven *space-delimited*",
    "numerical columns. The column values should be:",
    "  Column 1: unique identifier for one edge of the road network",
    "  Column 2: identifier of the from-vertex of the edge",
    "  Column 3: identifier of the to-vertex of the edge",
    "  Column 4: longitude coordinate of the from-vertex",
    "  Column 5: latitude coordinate of the from-vertex",
    "  Column 6: longitude coordinate of the to-verex",
    "  Column 7: latitude coordinate of the to-vertex",
    "",
    "If the same value appears more than once in Column 1, then a",
    "DuplicateEdgeException is thrown. If the same values appears in Columns 2 and",
    "3, then a SQL exception is thrown because Jargo's data model does not allow",
    "self-referencing edges. Directed edges are allowed, for example the values in",
    "Columns 2 and 3 for one row are reversed are reversed in a different row of the",
    "file. Columns 4--7 should be in WGS84 coordinate system because Jargo uses",
    "haversine on the Earth's surface to calculate certain distances. Using WGS84",
    "also facilitates offline visualization of vehicle routes and other spatial",
    "items in geospatial software such as QGIS.",
    "",
    "The PROB file should be a plain-text file with three header rows and six",
    "*tab-delimited* numerical columns. The first header row is unused and can",
    "contain any text (e.g. notes to yourself). The second header row should have",
    "three space-delimited numbers, indicating the number of vehicles, customers,",
    "and the reference time, respectively. The reference time should be formatted as",
    "a four-digit military time, e.g. 1835 for 6:35 PM, or 0013 for 12:13 AM. The",
    "third header row is unused and can contain any text. I like to put the column",
    "headers in this row. The remaining rows have the following format:",
    "  Column 1: unique identifier of the vehicle or customer (the 'participant')",
    "  Column 2: identifier of the participant's origin vertex",
    "  Column 3: identifier of the participant's destination vertex",
    "  Column 4: the participant's 'load', position to indicate seating requirement",
    "            and negative to indicate seating capacity",
    "  Column 5: 'early' time, or time the participant appears on the network",
    "  Column 6: 'late' time, or latest acceptable time participant should arrive",
    "            at destination",
    "",
    "The GTREE and client g-tree (-r) should be in g-tree format (see",
    "https://github.com/jamjpan/GTree).",
    "",
    "The CLIENT and traffic (-x) should be Java jar archives containing a Client",
    "or Traffic class, respectively.",
    "",
    "TIME",
    "",
    "The start (-s) and end (-e) times are in seconds, relative to the problem",
    "reference time. For example, if the reference time is 0013 (12:13 AM) and the",
    "options '-s 0 -e 1800' are passed, then Jargo will simulate the 30 minutes",
    "between 12:13 AM and 12:43 AM. If no start and end times are passed, the",
    "default start time is 0 and the default end time is maximum 'early' time",
    "(Column 5 in the problem instance) plus 30.",
    "",
    "EXAMPLE",
    "",
    "  ./launch-cli \\",
    "    -r broadway.gtree \\",
    "    -x NormalTraffic.jar \\",
    "    -y com.example.NormalTraffic \\",
    "    -s 0 \\",
    "    -e 3600 \\",
    "    real \\",
    "    manhattan.rnet \\",
    "    manhattan.gtree \\",
    "    manhattan.instance \\",
    "    NearestNeighbor.jar \\",
    "    com.example.NearestNeighbor",
    "",
    "See the manual for more detail https://github.com/jargors/Jargo",
    ""
  );
```

Uses `DuplicateEdgeException` 63a and `end` 152d.

## 4.1.3 Methods

167a ⟨`Command` *methods* 167a⟩≡ (165a)

```
  public static ⟨Command: main(1) 167b⟩
```

### `main`(1)

167b ⟨*Command: main(1)* 167b⟩≡ (167a)

```
  void main(String[] args) throws Exception {
    final int REQUIRED_ARGS = 6;
    String arg1  = "";  // runtime mode
    String arg2  = "";  // road network *.rnet file
    String arg3  = "";  // gtree *.gtree file
    String arg4  = "";  // problem *.instance file
    String arg5  = "";  // client *.jar file
    String arg6  = "";  // client *.class classname
    String opt_r = "";  // client *.gtree file
    String opt_x = "";  // traffic *.jar file
    String opt_y = "";  // traffic *.class classname
    String opt_s = "";  // start time in seconds, relative to problem time
    String opt_e = "";  // end time in seconds, relative to problem time

    String help1 = ⟨Help string: usage 165e⟩
    String help2 = ⟨Help string: details 166⟩

    if (args.length == 1 && args[0].equals("-h")) {
      System.out.print(help1);
      System.out.print(help2);
      System.exit(0);
    } else if (args.length < REQUIRED_ARGS) {
      System.out.print(help1);
    } else {
      // Extract required arguments
      int j = (args.length - REQUIRED_ARGS);
      arg1 = args[(j + 0)];
      arg2 = args[(j + 1)];
      arg3 = args[(j + 2)];
      arg4 = args[(j + 3)];
      arg5 = args[(j + 4)];
      arg6 = args[(j + 5)];

      // Extract optional arguments
      int i = -1;
      while (i++ < j) {
        if (args[i].equals("-r")) {
          opt_r = args[(i + 1)];
        } else if (args[i].equals("-x")) {
          opt_x = args[(i + 1)];
        } else if (args[i].equals("-y")) {
```

```
      opt_y = args[(i + 1)];
    } else if (args[i].equals("-s")) {
      opt_s = args[(i + 1)];
    } else if (args[i].equals("-e")) {
      opt_e = args[(i + 1)];
    }
  }

  // Initialize road, problem, g-tree
  Controller ctrl = new Controller();
  ctrl.instanceNew();
  ctrl.instanceInitialize();
  System.out.printf("set road '%s'\n", arg2);
  ctrl.loadRoadNetworkFromFile(arg2);
  System.out.printf("set gtree '%s'\n", arg3);
  ctrl.gtreeLoad(arg3);
  System.out.printf("set problem '%s'\n", arg4);
  ctrl.loadProblem(arg4);

  Client client = null;
  Traffic traffic = null;

  URLClassLoader tmploader = null;
  Class<?> tmpclass = null;
  Constructor<?> tmpcstor = null;

  // Load Client
  System.out.printf("set client '%s'\n", arg5);
  System.out.printf("set client classname '%s'\n", arg6);
  tmploader = new URLClassLoader(new URL[] { new URL("file://" + arg5) },
      Class.forName("com.github.jargors.ui.Command").getClassLoader());
  tmpclass = Class.forName(arg6, true, tmploader);
  tmpcstor = tmpclass.getDeclaredConstructor();
  client = (Client) tmpcstor.newInstance();

  // Load Traffic
  System.out.printf("opt traffic '%s'\n", opt_x);
  System.out.printf("opt traffic classname '%s'\n", opt_y);
  if (!opt_x.equals("")) {
    tmploader = new URLClassLoader(new URL[] { new URL("file://" + opt_x) },
      Class.forName("com.github.jargors.ui.Command").getClassLoader());
    tmpclass = Class.forName(opt_y, true, tmploader);
    tmpcstor = tmpclass.getDeclaredConstructor();
    traffic = (Traffic) tmpcstor.newInstance();
  }

  // Initialize Client
  ctrl.setRefClient(client);
  ctrl.forwardRefCommunicator(ctrl.getRefCommunicator());
  client.forwardRefCacheVertices(ctrl.retrieveRefCacheVertices());
  client.forwardRefCacheEdges(ctrl.retrieveRefCacheEdges());
  client.forwardRefCacheUsers(ctrl.retrieveRefCacheUsers());
  opt_r = (opt_r.equals("") ? arg3 : opt_r);
  System.out.printf("opt client gtree '%s'\n", opt_r);
  client.gtreeLoad(opt_r);
  client.init();

  // Initialize Traffic
  if (traffic != null) {
    ctrl.forwardRefTraffic(traffic);
    traffic.forwardRefCacheVertices(ctrl.retrieveRefCacheVertices());
    traffic.forwardRefCacheEdges(ctrl.retrieveRefCacheEdges());
    traffic.init();
```

```
      }

      // Set start time
      opt_s = (opt_s.equals("") ? "0" : opt_s);
      System.out.printf("opt start '%s'\n", opt_s);
      ctrl.setClockStart(Integer.parseInt(opt_s));

      // Set end time
      opt_e = (opt_e.equals("")
          ? Integer.toString(ctrl.query("select max (re) from R", 1)[0])
          : opt_e);
      System.out.printf("opt end '%s'\n", opt_e);
      ctrl.setClockEnd(Integer.parseInt(opt_e));

      // Start simulation
      System.out.printf("set mode '%s'\n", arg1);
      if (arg1.equals("seq")) {
        ctrl.startSequential(⟨Simulation callback 165d⟩);
      } else if (arg1.equals("real")) {
        ctrl.startRealtime(⟨Simulation callback 165d⟩);
      } else {
        System.out.printf("Unrecognized mode; Exiting.\n");
        System.exit(1);
      }
    }
  }
```

Defines:
  `main`, never used.

Uses `end` 152d, `forwardRefCacheEdges` 62d, `forwardRefCacheUsers` 62e, `forwardRefCacheVertices` 62f, `forwardRefCommunicator` 62c, `forwardRefTraffic` 62b, `getRefCommunicator` 58b, `gtreeLoad` 128b, `instanceInitialize` 39a, `instanceNew` 37b, `loadProblem` 144d, `loadRoadNetworkFromFile` 143d, `query` 68b, `retrieveRefCacheEdges` 59b, `retrieveRefCacheUsers` 59c, `retrieveRefCacheVertices` 59a, `setClockEnd` 60b, `setClockStart` 60a, `setRefClient` 61c, `startRealtime` 145c, and `startSequential` 146a.

## 4.2 Class: Desktop

170a ⟨*Desktop.java* 170a⟩≡
```
  ⟨Package: ui 35b⟩
  ⟨Desktop.java preamble 170b⟩
  public class Desktop extends Application {
    ⟨Desktop methods 170c⟩
  }
```

### 4.2.1 Preamble

170b ⟨*Desktop.java preamble* 170b⟩≡ (170a)
```
  import com.github.jargors.ui.DesktopController;
  import java.io.IOException;
  import javafx.application.Application;
  import javafx.fxml.FXMLLoader;
  import javafx.scene.Scene;
  import javafx.stage.Stage;
```

### 4.2.2 Methods

170c ⟨Desktop *methods* 170c⟩≡ (170a)
```
  public void ⟨Desktop: start(1) 170d⟩
```

#### start(1)

170d ⟨*Desktop: start(1)* 170d⟩≡ (170c)
```
  start(Stage stage) throws IOException {
    System.out.println("JavaFX "+System.getProperties().get("javafx.runtime.version"));
    FXMLLoader fxmll = new FXMLLoader(Desktop.class.getResource("/fxml/Desktop.fxml"));
    Scene scene = new Scene(fxmll.load());
    DesktopController dc = fxmll.getController();
    dc.setStage(stage);
    scene.widthProperty().addListener((a, b, c) -> {
      dc.setWindowWidth((double) c);
    });
    scene.heightProperty().addListener((a, b, c) -> {
      dc.setWindowHeight((double) c);
    });
    stage.setTitle("Jargo Desktop");
    stage.setScene(scene);
    stage.show();
  }
```
Uses `setStage` 206c, `setWindowHeight` 206b, and `setWindowWidth` 206a.

## 4.3 Class: DesktopController

172a ⟨*DesktopController.java* 172a⟩≡
```
  ⟨Package: ui 35b⟩
  ⟨DesktopController.java preamble 172b⟩
  public class DesktopController {
    ⟨DesktopController member variables 173c⟩
    ⟨DesktopController methods 191c⟩
    public void initialize() {
      Image image = new Image("res/icon.gif");
      this.logo = new ImageView();
      this.logo.setImage(image);
      this.logo.setFitWidth(64);
      this.logo.setPreserveRatio(true);
      this.logo.setSmooth(true);
      this.logo.setCache(true);
    }
  }
```

### 4.3.1 Preamble

172b ⟨*DesktopController.java preamble* 172b⟩≡ (172a) 172c ▷
```
  import com.github.jargors.sim.*;
```

172c ⟨*DesktopController.java preamble* 172b⟩+≡ (172a) ◁ 172b 173a ▷
```
  import java.io.File;
  import java.io.FileInputStream;
  import java.io.FileNotFoundException;
  import java.io.IOException;
  import java.lang.StringBuilder;
  import java.lang.reflect.Constructor;
  import java.lang.reflect.InvocationTargetException;
  import java.net.MalformedURLException;
  import java.net.URL;
  import java.net.URLClassLoader;
  import java.sql.SQLException;
  import java.util.Arrays;
  import java.util.ArrayList;
  import java.util.Date;
  import java.util.HashMap;
  import java.util.List;
  import java.util.Map;
  import java.util.concurrent.CompletableFuture;
  import java.util.concurrent.ConcurrentHashMap;
  import java.util.concurrent.Executors;
  import java.util.concurrent.ScheduledExecutorService;
  import java.util.concurrent.ScheduledFuture;
  import java.util.concurrent.TimeUnit;
  import java.util.zip.ZipEntry;
  import java.util.zip.ZipInputStream;
```

173a ⟨*DesktopController.java preamble* 172b⟩+≡ (172a) ◁172c 173b▷
```
import javafx.animation.AnimationTimer;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.concurrent.Task;
import javafx.embed.swing.SwingNode;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Cursor;
import javafx.scene.Scene;
import javafx.scene.SnapshotParameters;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.CheckBox;
import javafx.scene.control.Label;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.Tab;
import javafx.scene.control.TabPane;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.image.WritableImage;
import javafx.scene.input.MouseEvent;
import javafx.scene.input.ScrollEvent;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Text;
import javafx.scene.transform.Rotate;
import javafx.stage.DirectoryChooser;
import javafx.stage.FileChooser.ExtensionFilter;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
```

173b ⟨*DesktopController.java preamble* 172b⟩+≡ (172a) ◁173a
```
import com.sun.tools.visualvm.charts.ChartFactory;
import com.sun.tools.visualvm.charts.SimpleXYChartDescriptor;
import com.sun.tools.visualvm.charts.SimpleXYChartSupport;
import javax.swing.JComponent;
import javax.swing.SwingUtilities;
```

## 4.3.2 Member Variables

### Constants

173c ⟨DesktopController *member variables* 173c⟩≡ (172a) 174a▷
```
private final Color C_ERROR   = Color.RED;
private final Color C_SUCCESS = Color.GREEN;
private final Color C_WARN    = Color.YELLOW;
private final String TITLE_STRING = "Jargo Desktop";
private final boolean DEBUG =
    "true".equals(System.getProperty("jargors.desktop.debug"));
```

### Interface

174a ⟨DesktopController *member variables* 173c⟩+≡ (172a) ◁173c 174b▷
```
  @FXML private AnchorPane container_lc_counts;
  @FXML private AnchorPane container_lc_distances;
  @FXML private AnchorPane container_lc_durations;
  @FXML private AnchorPane container_lc_rates;
  @FXML private AnchorPane container_lc_times;
  @FXML private Button btn_client;
  @FXML private Button btn_client_gtree;
  @FXML private Button btn_gtree;
  @FXML private Button btn_load;
  @FXML private Button btn_new;
  @FXML private Button btn_prob;
  @FXML private Button btn_query;
  @FXML private Button btn_road;
  @FXML private Button btn_startreal;
  @FXML private Button btn_startseq;
  @FXML private Button btn_stop;
  @FXML private Button btn_traffic;
  @FXML private CheckBox chk_continuous;
  @FXML private Circle circ_status;
  @FXML private Label lbl_status;
  @FXML private ScrollPane container_canvas;
  @FXML private Tab tab_map;
  @FXML private Tab tab_metrics;
  @FXML private TextArea txt_query;
  @FXML private TextArea txt_result;
  @FXML private TextField tf_client;
  @FXML private TextField tf_t0;
  @FXML private TextField tf_t1;
  @FXML private TextField tf_traffic;
  @FXML private VBox pane_info;
  private Canvas can_road;
  private Canvas can_servers;
  private Canvas can_requests;
  private Client client = null;
  private Controller controller = null;
  private Label lbl_fps;
  private Pane container_canvas_container;
  private Stage stage;
  private String clientclass = null;
  private String clientjar = null;
  private String db = null;
  private String gtree = null;
  private String gtree_client = null;
  private String prob = null;
  private String road = null;
  private String trafficclass = null;
  private String trafficjar = null;
  private Traffic traffic = null;
  private ImageView logo = null;
```

### Workspace

174b ⟨DesktopController *member variables* 173c⟩+≡ (172a) ◁174a 175▷
```
  private int access_path = 1;  // 1="New", 2="Load"
  private FetcherOfMapUnits muf = null;
  private GraphicsContext gc = null;
  private ScheduledFuture<?> cbFetcherOfMetrics = null;
  private RendererOfRoads ren_road = null;
  private RendererOfServers ren_servers = null;
  private RendererOfRequests ren_requests = null;
  private ScheduledExecutorService exe = null;
  private ScheduledExecutorService exe_query = null;
  private ScheduledFuture<?> cbFetcherOfLocations = null;
  private ScheduledFuture<?> cbFetcherOfRequests = null;
  private ScheduledFuture<?> cbSimulation = null;
  private double unit = 0;
  private double window_height = 0;
  private double window_width = 0;
  private double xunit = 0;
  private double yunit = 0;
  private int t0 = 0;
  private int t1 = 0;
  private int zoom = 1;
  private int[] edges = null;
  private int[] mbr = null;
  private int ns = 0;
  private int nr = 0;
```

## Renderers

175 ⟨DesktopController *member variables* 173c⟩+≡ (172a) ◁174b 176▷

```
  private class RendererOfRequests extends AnimationTimer {
    private final Color BG = Color.web("0xD7FFFF");
    private final Color REQUEST_FILL = Color.web("0xff5500");
    private final int REQUEST_WIDTH = 2;
    private final int REQUEST_HEIGHT = 2;
    private Canvas canvas = null;
    private ConcurrentHashMap<Integer, double[]> buffer =
        new ConcurrentHashMap<Integer, double[]>();
    private final ConcurrentHashMap<Integer, Image> bufimg =
        new ConcurrentHashMap<Integer, Image>();
    private FetcherOfMapUnits muf = null;
    private GraphicsContext gc = null;
    private Image image = null;
    private int zoom = 1;
    private long now = 0;
    private long prev = 0;
    public RendererOfRequests(final GraphicsContext gc, final FetcherOfMapUnits muf) {
      super();
      this.canvas = gc.getCanvas();
      this.gc = gc;
      this.gc.setStroke(REQUEST_FILL);
      this.muf = muf;
      this.setZoom(1);
      for (int i = 0; i <= 10; i++) {
        WritableImage wi = new WritableImage(80, 14);
        SnapshotParameters parameters = new SnapshotParameters();
        parameters.setFill(new Color(0, 0, 0, 0));
        Text txt = new Text(i == 10 ? "R" : ""+i);
        txt.setStroke(REQUEST_FILL);
        txt.snapshot(parameters, wi);
        this.bufimg.put(i, wi);
      }
    }
    public void setZoom(final int zoom) {
      this.zoom = zoom;
      Rectangle shape = new Rectangle(this.REQUEST_WIDTH*zoom, this.REQUEST_HEIGHT*zoom);
      shape.setFill(this.REQUEST_FILL);
      WritableImage wi = new WritableImage(this.REQUEST_WIDTH*zoom, this.REQUEST_HEIGHT*zoom);
      SnapshotParameters parameters = new SnapshotParameters();
      shape.snapshot(parameters, wi);
      this.image = wi;
    }
    public void fillBuffer(final int rid, final double[] buffer) {
      this.buffer.put(rid, buffer);
    }
    public void clearBuffer() {
      this.buffer.clear();
    }
    public void handle(final long now) {
      if (!this.muf.getMapVisible()) {
        return;
      }
      if (now - prev > 500_000_000) {  // render every 0.5 sec
        prev = now;
        this.gc.clearRect(0, 0, this.canvas.getWidth(), this.canvas.getHeight());
        for (final Map.Entry<Integer, double[]> kv : this.buffer.entrySet()) {
          final int rid = kv.getKey();
          final double[] buffer = kv.getValue();
          final double x = buffer[0]*this.muf.getUnit();
          final double y = this.canvas.getHeight() - buffer[1]*this.muf.getUnit();
          this.gc.drawImage(this.image, x, y);
          // this.gc.strokeText("R"+rid, x, y);
          /*int uid = rid;
          ⟨Draw identifier 186e⟩*/
        }
      }
    }
  }
```

176 ⟨DesktopController *member variables* 173c⟩+≡ (172a) ◁175 178▷

```
  private class RendererOfServers extends AnimationTimer {
    private final Color BG = Color.web("0xD7FFFF");
    private final Color SERVER_FILL = Color.web("0x555555");
    private final int SERVER_WIDTH = 5;
    private final int SERVER_HEIGHT = 3;
    private Canvas canvas = null;
    private final ConcurrentHashMap<Integer, Integer>  bufidx =
        new ConcurrentHashMap<Integer, Integer>();
    private final ConcurrentHashMap<Integer, double[]> buffer =
        new ConcurrentHashMap<Integer, double[]>();
    private final ConcurrentHashMap<Integer, Image> bufimg =
        new ConcurrentHashMap<Integer, Image>();
    private FetcherOfMapUnits muf = null;
    private GraphicsContext gc = null;
    private Image image = null;
    private Label lbl_fps = null;
    private boolean isRealtime = false;
    private int framecount = 0;
    private int zoom = 1;
    private long now = 0;
    private long prev = 0;
    public RendererOfServers(
        final GraphicsContext gc, final Label lbl_fps, final boolean isRealtime,
        final FetcherOfMapUnits muf) {
      super();
      this.canvas = gc.getCanvas();
      this.gc = gc;
      this.gc.setStroke(SERVER_FILL);
      this.isRealtime = isRealtime;
      this.lbl_fps = lbl_fps;
      this.muf = muf;
      this.setZoom(1);
      for (int i = 0; i <= 10; i++) {
        WritableImage wi = new WritableImage(80, 14);
        SnapshotParameters parameters = new SnapshotParameters();
        parameters.setFill(new Color(0, 0, 0, 0));
        Text txt = new Text(i == 10 ? "S" : ""+i);
        txt.setStroke(SERVER_FILL);
        txt.snapshot(parameters, wi);
        this.bufimg.put(i, wi);
      }
    }
    public void setZoom(final int zoom) {
      this.zoom = zoom;
      Rectangle shape = new Rectangle(this.SERVER_WIDTH*zoom, this.SERVER_HEIGHT*zoom);
      shape.setFill(this.SERVER_FILL);
      WritableImage wi = new WritableImage(this.SERVER_WIDTH*zoom, this.SERVER_HEIGHT*zoom);
      SnapshotParameters parameters = new SnapshotParameters();
      shape.snapshot(parameters, wi);
      this.image = wi;
    }
    public void fillBuffer(final int sid, final double[] buffer) {
      if (!this.bufidx.containsKey(sid)) {
        this.bufidx.put(sid, 0);
        this.buffer.put(sid, new double[] { 0,0,0, 0,0,0, 0,0,0 });
      }
      double[] ref = this.buffer.get(sid);
      int i = this.bufidx.get(sid);
      ref[(i + 0)] = this.now;
      ref[(i + 1)] = buffer[1];
      ref[(i + 2)] = buffer[2];
      i = (i + 3) % 9;
      ref[(i + 0)] = (this.now + (buffer[3] - buffer[0])*1_000_000_000);
      ref[(i + 1)] = buffer[4];
      ref[(i + 2)] = buffer[5];
      i = (i + 3) % 9;
      ref[(i + 0)] = (this.now + (buffer[6] - buffer[0])*1_000_000_000);
      ref[(i + 1)] = buffer[7];
      ref[(i + 2)] = buffer[8];
  /**
  if (sid == 1) System.out.printf("%.2f %.2f %.2f\n%.2f %.2f %.2f\n%.2f %.2f %.2f\n",
      buffer[0],
      buffer[1],
      buffer[2],
      buffer[3],
      buffer[4],
      buffer[5],
      buffer[6],
      buffer[7],
      buffer[8]);
  **/
    }
    public void handle(final long now) {
      if (!this.muf.getMapVisible()) {
        return;
      }
      this.now = now;
      // Count FPS
      if (now - prev > 1_000_000_000) {
        this.lbl_fps.setText(String.format("%d",this.framecount));
        prev = now;
        framecount = 0;
      } else {
        framecount++;
      }
      // Draw servers
      this.gc.clearRect(0, 0, this.canvas.getWidth(), this.canvas.getHeight());
      for (final Map.Entry<Integer, Integer> kv : this.bufidx.entrySet()) {
        final int sid = kv.getKey();
        final int i = kv.getValue();
        final double[] buffer = this.buffer.get(sid);
        if (buffer == null) {
          continue;
        }
        final double t1 = buffer[((i + 0) % 9)];
        final double x1 = buffer[((i + 1) % 9)];
        final double y1 = buffer[((i + 2) % 9)];
        final double t2 = buffer[((i + 3) % 9)];
        final double x2 = buffer[((i + 4) % 9)];
        final double y2 = buffer[((i + 5) % 9)];
        double x = 0;
        double y = 0;
        if (this.isRealtime) {
          double delta = ((double) (now - t1)/(t2 - t1));
          if (delta >= 1) {
            this.bufidx.put(sid, (i + 3) % 9);
            delta = 1;
          } else if (delta >= 0) {
            x = this.muf.getUnit()*(x1 + delta*(x2 - x1));
            y = this.canvas.getHeight() - this.muf.getUnit()*(y1 + delta*(y2 - y1));
          } else {
            // delta < 0 means we didn't get a buffer update
          }
  /**
  if (sid == 1) System.out.printf("S%d: (%.2f,%.2f) (%.2f,%.2f) bufidx=%d, delta=%.2f\n",
      sid, x1, y1, x2, y2, i, delta);
  **/
        } else {
          x = this.muf.getUnit()*x1;
          y = this.canvas.getHeight() - this.muf.getUnit()*y1;
        }
        this.gc.save();
        double angle = (360 - Math.toDegrees(Math.atan2((y2 - y1), (x2 - x1))));
        Rotate r = new Rotate(angle, x, y);
        this.gc.setTransform(r.getMxx(), r.getMyx(), r.getMxy(), r.getMyy(), r.getTx(), r.getTy());
        this.gc.drawImage(this.image, x, y);
        this.gc.restore();
        //this.gc.strokeText("S"+sid, x, y);
        /*int uid = sid;
        ⟨Draw identifier 186e⟩*/
      }
    }
  }
```

178 ⟨DesktopController *member variables* 173c⟩+≡ (172a) ◁176 180▷

```
  private class RendererOfRoads extends AnimationTimer {
    private long prev = 0;
    private GraphicsContext gc = null;
    private Canvas can_road = null;
    private Controller controller = null;
    private FetcherOfMapUnits muf = null;
    private Traffic traffic = null;
    private int[] edges = null;
    private final Color DEFAULT = Color.web("#BDBDBD");
    private final Color SPEED1 = Color.web("#FF0000");
    private final Color SPEED2 = Color.web("#FF2B00");
    private final Color SPEED3 = Color.web("#FF5600");
    private final Color SPEED4 = Color.web("#FF8100");
    private final Color SPEED5 = Color.web("#FFAC00");
    private final Color SPEED6 = Color.web("#FFD700");
    private final Color SPEED7 = Color.web("#D7D71B");
    private final Color SPEED8 = Color.web("#AFD737");
    private final Color SPEED9 = Color.web("#87D753");
    private final Color SPEED10 = Color.web("#5FD86F");
    private final double LINEWIDTH = 0.3;
    private final long PERIOD = 10000_000_000L;  // 10 seconds
    public RendererOfRoads(
        final GraphicsContext gc,
        final Controller controller,
        final FetcherOfMapUnits muf) {
      super();
      this.gc = gc;
      this.can_road = gc.getCanvas();
      this.controller = controller;
      try {
        this.edges = this.controller.queryEdges();
      } catch (SQLException se) {
        System.err.println("Couldn't get edges for road renderer");
        System.err.println(se.getMessage());
        se.printStackTrace();
      }
      this.muf = muf;
      this.gc.setLineWidth(LINEWIDTH);
    }
    public void setTraffic(final Traffic traffic) {
      this.traffic = traffic;
    }
    public void forceRender() {
      this.prev = 0;
    }
    public void handle(long now) {
      if (!this.muf.getMapVisible()) {
        return;
      }
      if ((now - this.prev) > PERIOD) {
        this.prev = now;
        // It is WAY faster to loop through and draw all the edges inside
        // handle(1) instead of drawing a single edge at a time inside handle(1).
        // My guess is because with the loop method, the internal graphics buffer
        // needs to be rendered to the screen only once for all edges but with
        // the single-edge method, it needs to be rendered once for each edge.
        this.gc.clearRect(0, 0, this.can_road.getWidth(), this.can_road.getHeight());
        for (int i = 0; i < (this.edges.length - 3); i += 4) {
          if (this.edges[(i + 0)] != 0 && this.edges[(i + 1)] != 0) {
            try {
              final int[] v1 = this.controller.queryVertex(this.edges[(i + 0)]);
              final int[] v2 = this.controller.queryVertex(this.edges[(i + 1)]);
              final double x1 = this.muf.getUnit()*(v1[0] - this.muf.getLngMin());
              final double x2 = this.muf.getUnit()*(v2[0] - this.muf.getLngMin());
              final double y1 = this.can_road.getHeight() - this.muf.getUnit()*(v1[1] - this.muf.getLatMin());
              final double y2 = this.can_road.getHeight() - this.muf.getUnit()*(v2[1] - this.muf.getLatMin());
              this.gc.setStroke(DEFAULT);
              if (this.traffic != null) {
                double x = this.traffic.apply(this.edges[i], this.edges[(i + 1)],
                    (1000*this.controller.getClock() + this.controller.getClockReferenceMs())
                );
                if (0.0 <= x && x <= 0.1) {
                  this.gc.setStroke(SPEED1);
                } else if (0.1 < x && x <= 0.2) {
                  this.gc.setStroke(SPEED2);
                } else if (0.2 < x && x <= 0.3) {
                  this.gc.setStroke(SPEED3);
                } else if (0.3 < x && x <= 0.4) {
                  this.gc.setStroke(SPEED4);
                } else if (0.4 < x && x <= 0.5) {
                  this.gc.setStroke(SPEED5);
                } else if (0.5 < x && x <= 0.6) {
                  this.gc.setStroke(SPEED6);
                } else if (0.6 < x && x <= 0.7) {
                  this.gc.setStroke(SPEED7);
                } else if (0.7 < x && x <= 0.8) {
                  this.gc.setStroke(SPEED8);
                } else if (0.8 < x && x <= 0.9) {
                  this.gc.setStroke(SPEED9);
                } else if (0.9 < x && x <= 1.0) {
                  this.gc.setStroke(SPEED10);
                }
              }
              // It seems to be less laggy if we call strokeLine(4) right here
              // compared to using Platform.runLater(1). Not sure if
              // strokePolyline(3) would be even faster, but we can't use
              // strokePolyline(3) anyway because each of our lines might be a
              // different color depending on traffic.
              this.gc.strokeLine(x1, y1, x2, y2);
            } catch (VertexNotFoundException ve) {
              System.err.println("Warning: "+ve.toString());
            } catch (SQLException se) {
              System.err.println("Warning: FetcherOfLocations failed to get locations");
              System.err.println(se.toString());
              se.printStackTrace();
            } catch (Exception ee) {
              ee.printStackTrace();
            }
          }
        }
      }
    }
  }
```

Uses apply 156a, getClock 56b, getClockReferenceMs 57c, queryEdges 72b, queryVertex 70a, and VertexNotFoundException 65b.

## Fetchers

180 ⟨DesktopController *member variables* 173c⟩+≡ (172a) ◁178 181▷

```
  private class FetcherOfMapUnits {
    private double unit = 0;
    private int lng_min = 0;
    private int lat_min = 0;
    private int lng_max = 0;
    private int lat_max = 0;
    private boolean mapVisible = true;
    public FetcherOfMapUnits() { }
    public double getUnit() {
      return this.unit;
    }
    public int getLngMin() {
      return this.lng_min;
    }
    public int getLngMax() {
      return this.lng_max;
    }
    public int getLatMin() {
      return this.lat_min;
    }
    public int getLatMax() {
      return this.lat_max;
    }
    public boolean getMapVisible() {
      return this.mapVisible;
    }
    public void setUnit(final double unit) {
      this.unit = unit;
    }
    public void setLngMin(final int lng_min) {
      this.lng_min = lng_min;
    }
    public void setLatMin(final int lat_min) {
      this.lat_min = lat_min;
    }
    public void setLngMax(final int lng_max) {
      this.lng_max = lng_max;
    }
    public void setLatMax(final int lat_max) {
      this.lat_max = lat_max;
    }
    public void setMapVisible(final boolean flag) {
      this.mapVisible = flag;
    }
  }
```

181 ⟨DesktopController *member variables* 173c⟩+≡ (172a) ◁180 182▷

```
  private class FetcherOfRequests implements Runnable {
    private Controller controller = null;
    private FetcherOfMapUnits muf = null;
    private RendererOfRequests renderer = null;
    public FetcherOfRequests(
        final Controller controller,
        final FetcherOfMapUnits muf,
        final RendererOfRequests renderer) {
      this.controller = controller;
      this.muf = muf;
      this.renderer = renderer;
    }
    public void run() {
      if (!this.muf.getMapVisible()) {
        return;
      }
      final int t = this.controller.getClock();
      try {
        this.renderer.clearBuffer();
        int[] waiting = this.controller.queryRequestsWaiting(t);
        for (int i = 0; i < (waiting.length - 1); i += 2) {
          final int rid = waiting[(i + 0)];
          final int  ro = waiting[(i + 1)];
          int[] coordinates = this.controller.queryVertex(ro);
          final double x = (coordinates[0] - this.muf.getLngMin());
          final double y = (coordinates[1] - this.muf.getLatMin());
          this.renderer.fillBuffer(rid, new double[] { x, y });
        }
      } catch (SQLException se) {
        System.err.println("Warning: FetcherOfRequests failed to get requests");
        System.err.println(se.toString());
        se.printStackTrace();
      } catch (VertexNotFoundException ve) {
        System.err.println("Warning: FetcherOfRequests got unknown location!");
        System.err.println(ve.toString());
        ve.printStackTrace();
      } catch (Exception ee) {
        ee.printStackTrace();
      }
    }
  }
```

Uses getClock 56b, queryRequestsWaiting 82b, queryVertex 70a, and VertexNotFoundException 65b.

182 ⟨DesktopController *member variables* 173c⟩+≡ (172a) ◁181 183▷
```
  private class FetcherOfLocations implements Runnable {
    private Controller controller = null;
    private FetcherOfMapUnits muf = null;
    private Map<Integer, double[]> buffer = new HashMap<Integer, double[]>();
    private RendererOfServers renderer = null;
    public FetcherOfLocations(
        final Controller controller,
        final FetcherOfMapUnits muf,
        final RendererOfServers renderer) {
      this.controller = controller;
      this.muf = muf;
      this.renderer = renderer;
    }
    public void run() {
      if (!this.muf.getMapVisible()) {
        return;
      }
      final int t = this.controller.getClock();
      try {
        int[] active = this.controller.queryServersActive(t);
        for (int i = 0; i < active.length; i++) {
          final int sid = active[i];
          if ((!this.buffer.containsKey(sid)) || (this.buffer.get(sid)[3] <= t)) {
            int[] route = this.controller.queryServerRouteActive(sid);
            if (route.length == 6) {  // ...
              final int t1 = route[0];
              final int v1 = route[1];
              final int t2 = route[2];
              final int v2 = route[3];
              final int t3 = route[4];
              final int v3 = route[5];
  if (this.buffer.containsKey(sid) && t1 == this.buffer.get(sid)[0]) {
    // we didn't get an update
    continue;
  }
  /**
  if (sid == 1) System.out.printf("%d %d\n%d %d\n%d %d\n",
      route[0],
      route[1],
      route[2],
      route[3],
      route[4],
      route[5]);
  **/
              int[] coordinates = this.controller.queryVertex(v1);
              final double x1 = (coordinates[0] - this.muf.getLngMin());
              final double y1 = (coordinates[1] - this.muf.getLatMin());
              coordinates = this.controller.queryVertex(v2);
              final double x2 = (coordinates[0] - this.muf.getLngMin());
              final double y2 = (coordinates[1] - this.muf.getLatMin());
              coordinates = this.controller.queryVertex(v3);
              final double x3 = (coordinates[0] - this.muf.getLngMin());
              final double y3 = (coordinates[1] - this.muf.getLatMin());
              double[] newbuf = new double[] { t1, x1, y1, t2, x2, y2, t3, x3, y3 };
              this.buffer.put(sid, newbuf);
              this.renderer.fillBuffer(sid, newbuf);
            }
          }
        }
      } catch (SQLException se) {
        System.err.println("Warning: FetcherOfLocations failed to get locations");
        System.err.println(se.toString());
        se.printStackTrace();
      } catch (VertexNotFoundException ve) {
        System.err.println("Warning: FetcherOfLocations got unknown location!");
        System.err.println(ve.toString());
        ve.printStackTrace();
      } catch (Exception ee) {
        ee.printStackTrace();
      }
    }
  }
```

Uses getClock 56b, queryServerRouteActive 83d, queryServersActive 93c, queryVertex 70a, and VertexNotFoundException 65b.

183 ⟨DesktopController *member variables* 173c⟩+≡ (172a) ◁182 185▷

```
  private class FetcherOfMetrics implements Runnable {
    private Controller controller = null;
    private Label lbl_status = null;
    private VBox pane_info = null;
    private ConcurrentHashMap<String, SimpleXYChartSupport> lu_series = null;
    private long A0 = 0;
    private long t_ref = 0;
    private int ns_total = 0;
    private int nr_total = 0;
    private int ns = 0;
    private int nr = 0;
    private Text txt1 = null;
    private Text txt2 = null;
    private Text txt3 = null;
    private Text txt4 = null;
    public FetcherOfMetrics(
        final Controller controller,
        final Label lbl_status,
        final VBox pane_info,
        final ConcurrentHashMap<String, SimpleXYChartSupport> lu_series,
        final int ns,
        final int nr) {
      this.controller = controller;
      this.lbl_status = lbl_status;
      this.pane_info = pane_info;
      this.lu_series = lu_series;
      this.ns_total = ns;
      this.nr_total = nr;
      this.t_ref = this.controller.getClockReferenceMs();
      this.pane_info.getChildren().add(new Text("Count Requests Queued "));
      this.pane_info.getChildren().add(new Text("Service Rate "));
      this.pane_info.getChildren().add(new Text("S-Violations "));
      this.pane_info.getChildren().add(new Text("R-Violations "));
      this.txt1 = (Text) this.pane_info.getChildren().get(0);
      this.txt2 = (Text) this.pane_info.getChildren().get(1);
      this.txt3 = (Text) this.pane_info.getChildren().get(2);
      this.txt4 = (Text) this.pane_info.getChildren().get(3);
    }
    public void run() {
      if (DEBUG) {
        this.A0 = System.currentTimeMillis();
      }
      final int t = this.controller.getClock();
      final int day = this.controller.getClockReferenceDay();
      final int hr = this.controller.getClockReferenceHour();
      final int min = this.controller.getClockReferenceMinute();
      final int sec = this.controller.getClockReferenceSecond();
      Platform.runLater(() -> {
        this.lbl_status.setText(String.format("Collect metrics (t=%d) (day %d %02d:%02d:%02d)",
          t, day, hr, min, sec));
      });
      try {
        this.ns = this.controller.queryServersCountAppeared()[0];
        this.nr = this.controller.queryRequestsCountAppeared()[0];
        int[] output = new int[] { };
        Number val = null;
        ⟨Retrieve serviceRate 187a⟩                final long y01 = val.longValue();
        ⟨Retrieve distanceSavings 187b⟩            final long y02 = val.longValue();
        ⟨Retrieve serverTravelDistance 188a⟩       final long y03 = val.longValue();
        ⟨Retrieve serverServiceDistance 188b⟩      final long y04 = val.longValue();
        ⟨Retrieve serverCruisingDistance 188c⟩     final long y05 = val.longValue();
//        ⟨Retrieve requestDistanceUnassigned 188g⟩ final long y06 = val.longValue();
        ⟨Retrieve requestTransitDistance 189a⟩     final long y07 = val.longValue();
        ⟨Retrieve requestDetourDistance 189b⟩      final long y08 = val.longValue();
        ⟨Retrieve serverTravelDuration 188d⟩       final long y09 = val.longValue();
        ⟨Retrieve serverServiceDuration 188e⟩      final long y10 = val.longValue();
        ⟨Retrieve serverCruisingDuration 188f⟩     final long y11 = val.longValue();
        ⟨Retrieve requestTransitDuration 189c⟩     final long y12 = val.longValue();
        ⟨Retrieve requestTravelDuration 189e⟩      final long y13 = val.longValue();
        ⟨Retrieve requestPickupDuration 189f⟩      final long y14 = val.longValue();
        ⟨Retrieve countRequestsQueue 189g⟩          final long y15 = val.longValue();
//        ⟨Retrieve countRequestsActive 189h⟩       final long y16 = val.longValue();
//        ⟨Retrieve countRequestsCompleted 189i⟩    final long y17 = val.longValue();
//        ⟨Retrieve countServersActive 190b⟩        final long y18 = val.longValue();
        ⟨Retrieve countRequestsViolations 190c⟩    final long y19 = val.longValue();
        ⟨Retrieve countServersViolations 190d⟩     final long y20 = val.longValue();
        ⟨Retrieve timeRequestHandling 190e⟩         final long y21 = val.longValue();
        SwingUtilities.invokeLater(() -> {
           final long tf = 1000*t + t_ref;
           System.out.printf("tf=%d\n", tf);
           this.lu_series.get("lc_rates").addValues(tf, new long[] {
              y01,
              y02 });
           this.lu_series.get("lc_distances").addValues(tf, new long[] {
              y03,
              y04,
              y05,
//            y06,
              y07,
              y08 });
           this.lu_series.get("lc_durations").addValues(tf, new long[] {
              y09,
              y10,
              y11,
              y12,
              y13,
              y14 });
           this.lu_series.get("lc_counts").addValues(tf, new long[] {
              y15 });
//            y16,
//            y17,
//            y18,
//            y19,
//            y20 });
           this.lu_series.get("lc_times").addValues(tf, new long[] {
              y21 });
        });
        Platform.runLater(() -> {
          this.txt1.setText("Count Requests Queued "+y15);
          this.txt2.setText("Service Rate "+(y01/100.0)+"%");
          this.txt3.setText("S-Violations "+y20);
          this.txt4.setText("R-Violations "+y19);
        });
      } catch (SQLException se) {
        System.err.printf("SQL failure: %s\n", se.getMessage());
      } catch (Exception ee) {
        ee.printStackTrace();
      }
      if (DEBUG) {
        System.err.printf("t=%s, execution took %d ms\n", t, (System.currentTimeMillis() - this.A0));
      }
    }
  }
```
Uses getClock 56b, getClockReferenceMs 57c, queryRequestsCountAppeared 80c, and queryServersCountAppeared 95b.

## Charts

185 ⟨DesktopController *member variables* 173c⟩+≡ (172a) ◁183
```
  private SimpleXYChartDescriptor lc_counts_descriptor = SimpleXYChartDescriptor.decimal(0, true, 3600);
  private SimpleXYChartDescriptor lc_distances_descriptor = SimpleXYChartDescriptor.decimal(0, true, 3600);
  private SimpleXYChartDescriptor lc_durations_descriptor = SimpleXYChartDescriptor.decimal(0, true, 3600);
  private SimpleXYChartDescriptor lc_rates_descriptor = SimpleXYChartDescriptor.decimal(0, true, 3600);
  private SimpleXYChartDescriptor lc_times_descriptor = SimpleXYChartDescriptor.decimal(0, true, 3600);
  private SimpleXYChartSupport lc_counts_support = null;
  private SimpleXYChartSupport lc_distances_support = null;
  private SimpleXYChartSupport lc_durations_support = null;
  private SimpleXYChartSupport lc_rates_support = null;
  private SimpleXYChartSupport lc_times_support = null;
  private SwingNode lc_counts = new SwingNode();
  private SwingNode lc_distances = new SwingNode();
  private SwingNode lc_durations = new SwingNode();
  private SwingNode lc_rates = new SwingNode();
  private SwingNode lc_times = new SwingNode();
  private String[] metric_rates = new String[] {
  /*y01*/      "Running Service Rate (%, 100x)",
  /*y02*/      "Running Distance Savings (%, 100x)"
      };
  private String[] metric_distances = new String[] {
  /*y03*/      "S-Travel",
  /*y04*/      "S-Service",
  /*y05*/      "S-Cruising",
  ///*y06*/      "R-Unassigned",
  /*y07*/      "R-Transit",
  /*y08*/      "R-Detour"
      };
  private String[] metric_durations = new String[] {
  /*y09*/      "S-Travel",
  /*y10*/      "S-Service",
  /*y11*/      "S-Cruising",
  /*y12*/      "R-Transit",
  /*y13*/      "R-Travel",
  /*y14*/      "R-Pickup"
      };
  private String[] metric_counts = new String[] {
  /*y15*/      "R-Queue",
  ///*y16*/      "R-Active",
  ///*y17*/      "R-Completed",
  ///*y18*/      "S-Active",
  ///*y19*/      "R-Violations",
  ///*y20*/      "S-Violations"
      };
  private String[] metric_times = new String[] {
  /*y21*/      "R-Handling",
      };
  private ConcurrentHashMap<String, SimpleXYChartSupport> lu_series
    = new ConcurrentHashMap<String, SimpleXYChartSupport>();
```

### 4.3.3 Chunks

#### Set cursor wait

186a ⟨*Set cursor wait* 186a⟩≡ (192–98 200b 203)
```
  Platform.runLater(() -> {
    this.stage.getScene().setCursor(Cursor.WAIT);
  });
```

#### Set cursor default

186b ⟨*Set cursor default* 186b⟩≡ (192–98 200b 203)
```
  Platform.runLater(() -> {
    this.stage.getScene().setCursor(Cursor.DEFAULT);
  });
```

#### Initialize chart series

186c ⟨*Initialize chart series* 186c⟩≡ (186f)
```
  for (String metric : this.metric_rates) {
    this.lc_rates_descriptor.addLineItems(metric);
  }
  for (String metric : this.metric_distances) {
    this.lc_distances_descriptor.addLineItems(metric);
  }
  for (String metric : this.metric_durations) {
    this.lc_durations_descriptor.addLineItems(metric);
  }
  for (String metric : this.metric_counts) {
    this.lc_counts_descriptor.addLineItems(metric);
  }
  for (String metric : this.metric_times) {
    this.lc_times_descriptor.addLineItems(metric);
  }
```

#### Set default t0, t1

186d ⟨*Set default t0, t1* 186d⟩≡ (202)
```
  if ("".equals(this.tf_t0.getText())) {
    this.tf_t0.setText("0");
  }
  if ("".equals(this.tf_t1.getText())) {
    this.tf_t1.setText("1800");
  }
```

#### Draw identifier

186e ⟨*Draw identifier* 186e⟩≡ (175 176)
```
  char[] digits = String.valueOf(uid).toCharArray();
  this.gc.drawImage(this.bufimg.get(10), x, y);
  for (int j = 0; j < digits.length; j++) {
    this.gc.drawImage(
      this.bufimg.get(Character.getNumericValue(digits[j])),  // image
        position(x + 8*(j+1)), y);  // position
  }
```

#### Add charts to chart containers

186f ⟨*Add charts to chart containers* 186f⟩≡ (202)
```
  ⟨Initialize chart series 186c⟩
  this.lc_counts_descriptor.setYAxisDescription("<html>Count (#)</html>");
  this.lc_distances_descriptor.setYAxisDescription("<html>Avg. Distance (m)</html>");
  this.lc_durations_descriptor.setYAxisDescription("<html>Avg. Duration (sec)</html>");
  this.lc_rates_descriptor.setYAxisDescription("<html>Rate (%)</html>");
  this.lc_times_descriptor.setYAxisDescription("<html>Elapsed Time (ms)</html>");
  this.lc_counts_descriptor.setXAxisDescription("<html>World Time</html>");
  this.lc_distances_descriptor.setXAxisDescription("<html>World Time</html>");
  this.lc_durations_descriptor.setXAxisDescription("<html>World Time</html>");
  this.lc_rates_descriptor.setXAxisDescription("<html>World Time</html>");
  this.lc_times_descriptor.setXAxisDescription("<html>World Time</html>");
  this.lc_counts_support = ChartFactory.createSimpleXYChart(this.lc_counts_descriptor);
  this.lc_distances_support = ChartFactory.createSimpleXYChart(this.lc_distances_descriptor);
  this.lc_durations_support = ChartFactory.createSimpleXYChart(this.lc_durations_descriptor);
  this.lc_rates_support = ChartFactory.createSimpleXYChart(this.lc_rates_descriptor);
  this.lc_times_support = ChartFactory.createSimpleXYChart(this.lc_times_descriptor);
  this.lu_series.put("lc_counts", this.lc_counts_support);
  this.lu_series.put("lc_distances", this.lc_distances_support);
  this.lu_series.put("lc_durations", this.lc_durations_support);
  this.lu_series.put("lc_rates", this.lc_rates_support);
  this.lu_series.put("lc_times", this.lc_times_support);
  this.lc_counts.setContent(this.lc_counts_support.getChart());
  this.lc_distances.setContent(this.lc_distances_support.getChart());
  this.lc_durations.setContent(this.lc_durations_support.getChart());
  this.lc_rates.setContent(this.lc_rates_support.getChart());
  this.lc_times.setContent(this.lc_times_support.getChart());
  this.container_lc_rates.setTopAnchor(this.lc_rates, 0.0);
  this.container_lc_rates.setLeftAnchor(this.lc_rates, 0.0);
  this.container_lc_rates.setRightAnchor(this.lc_rates, 0.0);
  this.container_lc_rates.setBottomAnchor(this.lc_rates, 0.0);
  this.container_lc_rates.getChildren().add(this.lc_rates);
  this.container_lc_distances.setTopAnchor(this.lc_distances, 0.0);
  this.container_lc_distances.setLeftAnchor(this.lc_distances, 0.0);
  this.container_lc_distances.setRightAnchor(this.lc_distances, 0.0);
  this.container_lc_distances.setBottomAnchor(this.lc_distances, 0.0);
  this.container_lc_distances.getChildren().add(this.lc_distances);
  this.container_lc_durations.setTopAnchor(this.lc_durations, 0.0);
  this.container_lc_durations.setLeftAnchor(this.lc_durations, 0.0);
  this.container_lc_durations.setRightAnchor(this.lc_durations, 0.0);
  this.container_lc_durations.setBottomAnchor(this.lc_durations, 0.0);
  this.container_lc_durations.getChildren().add(this.lc_durations);
  this.container_lc_counts.setTopAnchor(this.lc_counts, 0.0);
  this.container_lc_counts.setLeftAnchor(this.lc_counts, 0.0);
  this.container_lc_counts.setRightAnchor(this.lc_counts, 0.0);
  this.container_lc_counts.setBottomAnchor(this.lc_counts, 0.0);
  this.container_lc_counts.getChildren().add(this.lc_counts);
  this.container_lc_times.setTopAnchor(this.lc_times, 0.0);
  this.container_lc_times.setLeftAnchor(this.lc_times, 0.0);
  this.container_lc_times.setRightAnchor(this.lc_times, 0.0);
  this.container_lc_times.setBottomAnchor(this.lc_times, 0.0);
  this.container_lc_times.getChildren().add(this.lc_times);
```

### Retrieve serviceRate

187a ⟨*Retrieve serviceRate* 187a⟩≡ (183)

```
  output = this.controller.queryMetricServiceRateRunning();
  val = (output.length > 0 ? output[0] : 0);
```

Uses `queryMetricServiceRateRunning` 98d.

### Retrieve distanceSavings

187b ⟨*Retrieve distanceSavings* 187b⟩≡ (183)

```
  output = this.controller.queryMetricServerDistanceRunning();
  final int val1 = (output.length > 0 ? output[0] : 0);
  output = this.controller.queryMetricRequestDistanceBaseUnassignedRunning();
  final int val2 = (output.length > 0 ? output[0] : 0);
  output = this.controller.queryMetricUserDistanceBaseRunning();
  final int val3 = (output.length > 0 ? output[0] : 0);
  val = (val3 == 0 ? 0 : (100.0*100*(1 - ((double) (val1 + val2)/val3))));
```
Uses queryMetricServerDistanceRunning 101a and queryMetricUserDistanceBaseRunning 99e.

### Retrieve serverTravelDistance

188a ⟨*Retrieve serverTravelDistance* 188a⟩≡ (183)
```
  output = this.controller.queryMetricServerDistanceTotal();
  val = (output.length > 0 ? Math.round(output[0]/(double) this.ns) : 0);
```
Uses queryMetricServerDistanceTotal 100c.

### Retrieve serverServiceDistance

188b ⟨*Retrieve serverServiceDistance* 188b⟩≡ (183)
```
  output = this.controller.queryMetricServerDistanceServiceTotal();
  val = (output.length > 0 ? Math.round(output[0]/(double) this.ns) : 0);
```
Uses queryMetricServerDistanceServiceTotal 103b.

### Retrieve serverCruisingDistance

188c ⟨*Retrieve serverCruisingDistance* 188c⟩≡ (183)
```
  output = this.controller.queryMetricServerDistanceCruisingTotal();
  val = (output.length > 0 ? Math.round(output[0]/(double) this.ns) : 0);
```
Uses queryMetricServerDistanceCruisingTotal 102c.

### Retrieve serverTravelDuration

HORRIBLE HACK WARNING. Sometimes output[0] is not empty but this.ns is 0. This happens in the beginning if all servers are taxis. Taxis have an initial duration of 1 sec to move to the dummy vertex, but they don't count has appeared, hence ns = 0. the first horrible hack is to check if this.ns ¿ 0 before doing the averaging. the second horrible hack is to subtract away the 1-second. It's horrible because we've assumed all the servers are taxis, which maybe is not the case.

188d ⟨*Retrieve serverTravelDuration* 188d⟩≡ (183)
```
  output = this.controller.queryMetricServerDurationTravelTotal();
  val = (output.length > 0 && this.ns > 0
      ? Math.round(Math.max(0, output[0] - this.ns_total)/(double) this.ns) : 0);
```
Uses queryMetricServerDurationTravelTotal 104b.

### Retrieve serverServiceDuration

188e ⟨*Retrieve serverServiceDuration* 188e⟩≡ (183)
```
  output = this.controller.queryMetricServerDurationServiceTotal();
  val = (output.length > 0 ? Math.round(output[0]/(double) this.ns) : 0);
```
Uses queryMetricServerDurationServiceTotal 105e.

### Retrieve serverCruisingDuration

HORRIBLE HACK WARNING. Same horrible hack as serverTravelDuration chunk.

188f ⟨*Retrieve serverCruisingDuration* 188f⟩≡ (183)
```
  output = this.controller.queryMetricServerDurationCruisingTotal();
  val = (output.length > 0 && this.ns > 0
      ? Math.round(Math.max(0, output[0] - this.ns_total)/(double) this.ns) : 0);
```
Uses queryMetricServerDurationCruisingTotal 105a.

### Retrieve requestDistanceUnassigned

188g ⟨*Retrieve requestDistanceUnassigned* 188g⟩≡ (183)
```
  output = this.controller.queryMetricRequestDistanceBaseUnassignedTotal();
  val = (output.length > 0 ? output[0] : 0);
```
Uses queryMetricRequestDistanceBaseUnassignedTotal 107c.

### Retrieve requestTransitDistance

189a ⟨*Retrieve requestTransitDistance* 189a⟩≡ (183)
```
  output = this.controller.queryMetricRequestDistanceTransitTotal();
  val = (output.length > 0 ? Math.round(output[0]/(double) this.nr) : 0);
```
Uses `queryMetricRequestDistanceTransitTotal` 109d.

### Retrieve requestDetourDistance

189b ⟨*Retrieve requestDetourDistance* 189b⟩≡ (183)
```
  output = this.controller.queryMetricRequestDistanceDetourTotal();
  val = (output.length > 0 ? Math.round(output[0]/(double) this.nr) : 0);
```
Uses `queryMetricRequestDistanceDetourTotal` 108d.

### Retrieve requestTransitDuration

189c ⟨*Retrieve requestTransitDuration* 189c⟩≡ (183)
```
  output = this.controller.queryMetricRequestDurationTransitTotal();
  val = (output.length > 0 ? Math.round(output[0]/(double) this.nr) : 0);
```
Uses `queryMetricRequestDurationTransitTotal` 111c.

### Retrieve requestDetourDuration

189d ⟨*Retrieve requestDetourDuration* 189d⟩≡

### Retrieve requestTravelDuration

189e ⟨*Retrieve requestTravelDuration* 189e⟩≡ (183)
```
  output = this.controller.queryMetricRequestDurationTravelTotal();
  val = (output.length > 0 ? Math.round(output[0]/(double) this.nr) : 0);
```
Uses `queryMetricRequestDurationTravelTotal` 112c.

### Retrieve requestPickupDuration

189f ⟨*Retrieve requestPickupDuration* 189f⟩≡ (183)
```
  output = this.controller.queryMetricRequestDurationPickupTotal();
  val = (output.length > 0 ? Math.round(output[0]/(double) this.nr) : 0);
```
Uses `queryMetricRequestDurationPickupTotal` 110d.

### Retrieve countRequestsQueue

189g ⟨*Retrieve countRequestsQueue* 189g⟩≡ (183)
```
  val = this.controller.retrieveQueueSize();
```
Uses `retrieveQueueSize` 58d.

### Retrieve countRequestsActive

189h ⟨*Retrieve countRequestsActive* 189h⟩≡ (183)
```
  output = this.controller.queryRequestsCountActive(t);
  val = (output.length > 0 ? output[0] : 0);
```
Uses `queryRequestsCountActive` 80a.

### Retrieve countRequestsCompleted

189i ⟨*Retrieve countRequestsCompleted* 189i⟩≡ (183)
```
  output = this.controller.queryRequestsCountCompleted(t);
  val = (output.length > 0 ? output[0] : 0);
```
Uses `queryRequestsCountCompleted` 81a.

### Retrieve countRequestsFailed

190a ⟨*Retrieve countRequestsFailed* 190a⟩≡

### Retrieve countServersActive

190b ⟨*Retrieve countServersActive* 190b⟩≡ (183)

```
output = this.controller.queryServersCountActive(t);
val = (output.length > 0 ? output[0] : 0);
```

Uses `queryServersCountActive` 94d.

### Retrieve countRequestsViolations

190c ⟨*Retrieve countRequestsViolations* 190c⟩≡ (183)

```
output = this.controller.queryMetricRequestTWViolationsTotal();
val = (output.length > 0 ? output[0] : 0);
```

Uses `queryMetricRequestTWViolationsTotal` 113.

### Retrieve countServersViolations

190d ⟨*Retrieve countServersViolations* 190d⟩≡ (183)

```
output = this.controller.queryMetricServerTWViolationsTotal();
val = (output.length > 0 ? output[0] : 0);
```

Uses `queryMetricServerTWViolationsTotal` 106b.

### Retrieve timeRequestHandling

Retrieves the duration of the last call to handle request (suffers from aliasing).

190e ⟨*Retrieve timeRequestHandling* 190e⟩≡ (183)

```
val = this.controller.retrieveHandleRequestDur();
```

Uses `retrieveHandleRequestDur` 58e.

### Retrieve timeServerHandling

190f ⟨*Retrieve timeServerHandling* 190f⟩≡

### Load client

190g ⟨*Load client* 190g⟩≡ (202)

```
this.clientclass = this.tf_client.getText();
if ("".equals(this.clientclass)) {
  System.err.println("Class empty!");
  return;
}
try {
  URLClassLoader loader = new URLClassLoader(new URL[] {new URL("file://"+this.clientjar)},
      this.getClass().getClassLoader());
  Class<?> tempclass = Class.forName(this.clientclass, true, loader);
  Constructor<?> tempcstor = tempclass.getDeclaredConstructor();
  this.client = (Client) tempcstor.newInstance();
  this.controller.setRefClient(this.client);
  this.controller.forwardRefCommunicator(this.controller.getRefCommunicator());
  this.client.forwardRefCacheVertices(this.controller.retrieveRefCacheVertices());
  this.client.forwardRefCacheEdges(this.controller.retrieveRefCacheEdges());
  this.client.forwardRefCacheUsers(this.controller.retrieveRefCacheUsers());
  this.client.init();
} catch (MalformedURLException
    | ClassNotFoundException
    | NoSuchMethodException
    | InstantiationException
    | IllegalAccessException
```

```
      | InvocationTargetException me) {
    System.err.println(me.toString());
    me.printStackTrace();
    return;
  }
```

Uses forwardRefCacheEdges 62d, forwardRefCacheUsers 62e, forwardRefCacheVertices 62f, forwardRefCommunicator 62c, getRefCommunicator 58b, retrieveRefCacheEdges 59b, retrieveRefCacheUsers 59c, retrieveRefCacheVertices 59a, and setRefClient 61c.

### Load traffic

191a ⟨*Load traffic* 191a⟩≡ (202)

```
  this.trafficclass = this.tf_traffic.getText();
  try {
    if (this.trafficclass.length() > 0) {
      URLClassLoader loader2 = new URLClassLoader(new URL[] {new URL("file://"+this.trafficjar)},
          this.getClass().getClassLoader());
      Class<?> tempclass2 = Class.forName(this.trafficclass, true, loader2);
      Constructor<?> tempcstor2 = tempclass2.getDeclaredConstructor();
      this.traffic = (Traffic) tempcstor2.newInstance();
      this.traffic.forwardRefCacheEdges(controller.retrieveRefCacheEdges());
      this.traffic.forwardRefCacheVertices(controller.retrieveRefCacheVertices());
      this.traffic.init();
      this.controller.forwardRefTraffic(this.traffic);
      this.ren_road.setTraffic(this.traffic);
    }
  } catch (MalformedURLException
      | ClassNotFoundException
      | NoSuchMethodException
      | InstantiationException
      | IllegalAccessException
      | InvocationTargetException me) {
    System.err.println(me.toString());
    me.printStackTrace();
    return;
  }
```

Uses forwardRefCacheEdges 62d, forwardRefCacheVertices 62f, forwardRefTraffic 62b, retrieveRefCacheEdges 59b, and retrieveRefCacheVertices 59a.

### Load Gtree

191b ⟨*Load gtree* 191b⟩≡ (202)

```
  try {
    this.client.gtreeLoad(this.gtree_client);
  } catch (FileNotFoundException fe) {
    System.err.println(e.toString());
    return;
  }
```

Uses gtreeLoad 128b.

## 4.3.4 Methods

### Special Methods

191c ⟨DesktopController *methods* 191c⟩≡ (172a) 192a ▷

```
  public ⟨actionAbout(1) 192d⟩
  public ⟨actionClient(1) 197⟩
  public ⟨actionClientGtree(1) 198⟩
  public ⟨actionGitHub(1) 192c⟩
  public ⟨actionGtree(1) 196⟩
  public ⟨actionLoad(1) 193⟩
  public ⟨actionNew(1) 192e⟩
  public ⟨actionProb(1) 195⟩
  public ⟨actionQuery(1) 199⟩
  public ⟨actionQueryContinuous(1) 200a⟩
  public ⟨actionQuit(1) 192b⟩
  public ⟨actionRoad(1) 194⟩
  public ⟨actionStartRealtime(1) 202b⟩
  public ⟨actionStartSequential(1) 202a⟩
  public ⟨actionStop(1) 203⟩
  public ⟨actionTraffic(1) 200b⟩
  public ⟨actionZoomCanvas(1) 205b⟩
  public ⟨setStage(1) 206c⟩
  public ⟨setWindowHeight(1) 206b⟩
  public ⟨setWindowWidth(1) 206a⟩
  public ⟨toggleMetric(1) 205a⟩
```

## Private Methods

192a ⟨DesktopController *methods* 191c⟩+≡ (172a) ◁191c

```
  private ⟨initializeCanvas(0) 205c⟩
```

### actionQuit(1)

192b ⟨*actionQuit(1)* 192b⟩≡ (191c)

```
  void actionQuit(final ActionEvent e) {
    System.exit(0);
  }
```

Defines:
  actionQuit, never used.

### actionGitHub(1)

192c ⟨*actionGitHub(1)* 192c⟩≡ (191c)

```
  void actionGitHub(final ActionEvent e) {
    // ...
  }
```

Defines:
  actionGitHub, never used.

### actionAbout(1)

192d ⟨*actionAbout(1)* 192d⟩≡ (191c)

```
  void actionAbout(final ActionEvent e) {
    Alert alert = new Alert(AlertType.INFORMATION, "https:github.com/jargors");
    alert.setTitle("About");
    alert.setHeaderText("Jargo Desktop");
    alert.setGraphic(this.logo);
    alert.showAndWait();
  }
```

Defines:
  actionAbout, never used.

### actionNew(1)

192e ⟨*actionNew(1)* 192e⟩≡ (191c)

```
  void actionNew(final ActionEvent e) {
    ⟨Set cursor wait 186a⟩
    this.btn_new      .setDisable(true);
    this.btn_load     .setDisable(true);
    this.btn_stop     .setDisable(true);
    this.circ_status.setFill(C_WARN);
    this.lbl_status.setText("Create new Jargo instance...");
    CompletableFuture.runAsync(() -> {
      this.controller = new Controller();
      try {
        this.controller.instanceNew();
      } catch (SQLException se) {
        Alert alert = new Alert(AlertType.ERROR, se.getMessage());
        alert.showAndWait();
        System.exit(1);
      }
      this.controller.instanceInitialize();
      this.db = "no-name";
      Platform.runLater(() -> {
        this.access_path = 1;
        this.btn_road     .setDisable(false);
        this.btn_stop     .setDisable(false);
        this.container_canvas.setContent(null);
        this.container_lc_rates.getChildren().clear();
        this.container_lc_distances.getChildren().clear();
        this.container_lc_durations.getChildren().clear();
        this.circ_status.setFill(C_SUCCESS);
        this.lbl_status.setText("Created new Jargo instance.");
      });
      ⟨Set cursor default 186b⟩
    });
  }
```

Defines:
  `actionNew`, never used.
Uses `instanceInitialize` 39a and `instanceNew` 37b.

## actionLoad(1)

193 ⟨*actionLoad(1)* 193⟩≡ (191c)

```
  void actionLoad(final ActionEvent e) {
    this.btn_new      .setDisable(true);
    this.btn_load     .setDisable(true);
    this.btn_stop     .setDisable(true);
    DirectoryChooser dc = new DirectoryChooser();
    File db = dc.showDialog(this.stage);
    if (db != null) {
      ⟨Set cursor wait 186a⟩
      this.db = db.toString();
      this.circ_status.setFill(C_WARN);
      this.lbl_status.setText("Load '"+this.db+"'...");
      CompletableFuture.runAsync(() -> {
        try {
          this.controller = new Controller();
          this.controller.instanceLoad(this.db);
          this.controller.cacheRoadNetworkFromDB();
          this.controller.cacheUsersFromDB();
          int nv = this.controller.queryVerticesCount()[0];
          int ne = this.controller.queryEdgesCount()[0];
          this.ns = this.controller.queryServersCount()[0];
          this.nr = this.controller.queryRequestsCount()[0];
          Platform.runLater(() -> {
            this.access_path = 2;
            this.btn_prob      .setText("*in-instance problem*");
            this.prob = "*in-instance problem*";
            this.btn_road      .setText("*in-instance road network*");
            this.road = "*in-instance road network*";
            this.btn_gtree     .setDisable(false);
            this.btn_stop      .setDisable(false);
            this.circ_status   .setFill(C_SUCCESS);
            this.lbl_status    .setText("Loaded Jargo instance (#vertices="+nv+"; #edges="+ne+") (#servers="+ns+
            this.container_canvas.setContent(null);
            this.container_lc_rates.getChildren().clear();
            this.container_lc_distances.getChildren().clear();
            this.container_lc_durations.getChildren().clear();
            this.initializeCanvas();
            this.ren_road = new RendererOfRoads(this.can_road.getGraphicsContext2D(), this.controller, this.muf
            this.ren_road.start();
          });
          ⟨Set cursor default 186b⟩
        } catch (SQLException se) {
          if (DEBUG) {
            Tools.PrintSQLException(se);
          }
          Platform.runLater(() -> {
            this.circ_status  .setFill(C_ERROR);
            this.lbl_status   .setText("Failed to load snapshot!");
            Alert alert = new Alert(AlertType.ERROR, "Couldn't load snapshot! (Not a valid Jargo instance?)");
            alert.showAndWait();
            this.btn_new      .setDisable(false);
            this.btn_load     .setDisable(false);
            this.btn_stop     .setDisable(false);
            this.circ_status  .setFill(C_SUCCESS);
            this.lbl_status   .setText("Ready.");
          });
          ⟨Set cursor default 186b⟩
        }
      });
    } else {
      // FD canceled
      this.btn_new      .setDisable(false);
      this.btn_load     .setDisable(false);
      this.btn_stop     .setDisable(false);
      this.circ_status  .setFill(C_SUCCESS);
      this.lbl_status   .setText("Ready.");
    }
  }
```

Defines:
  actionLoad, never used.
Uses cacheRoadNetworkFromDB 42d, cacheUsersFromDB 44a, initializeCanvas 205c, instanceLoad 39c, PrintSQLException 163e, queryEdgesCount 72d, queryRequestsCount 79c, queryServersCount 94b, and queryVerticesCount 71a.

## actionRoad(1)

194 ⟨*actionRoad(1)* 194⟩≡ (191c)

```
  void actionRoad(final ActionEvent e) {
    this.btn_road     .setDisable(true);
    this.btn_stop     .setDisable(true);
    this.circ_status  .setFill(C_WARN);
    this.lbl_status   .setText("Select *.rnet...");
    FileChooser fc = new FileChooser();
    fc.getExtensionFilters().addAll(new ExtensionFilter("Road network *.rnet", "*.rnet"));
    File road = fc.showOpenDialog(this.stage);
    if (road != null) {
      ⟨Set cursor wait 186a⟩
      this.road = road.toString();
      this.lbl_status.setText("Load '"+this.road+"'...");
      CompletableFuture.runAsync(() -> {
        try {
          this.controller.loadRoadNetworkFromFile(this.road);
          int nv = this.controller.queryVerticesCount()[0];
          int ne = this.controller.queryEdgesCount()[0];
          Platform.runLater(() -> {
            this.btn_road     .setText(road.getName());
            this.btn_stop     .setDisable(false);
            this.btn_gtree    .setDisable(false);
            this.circ_status  .setFill(C_SUCCESS);
            this.lbl_status   .setText("Loaded "+road.getName()+" (#vertices="+nv+"; #edges="+ne+")");
            this.initializeCanvas();
            this.stage.getScene().setCursor(Cursor.DEFAULT);
            this.ren_road = new RendererOfRoads(this.can_road.getGraphicsContext2D(), this.controller, this.muf
            this.ren_road.start();
          });
          ⟨Set cursor default 186b⟩
        } catch (Exception ee) {
          if (DEBUG) {
            System.err.println("Failed: "+ee.toString());
          }
          this.circ_status  .setFill(C_ERROR);
          this.lbl_status   .setText("Failed to load road network!");
          Alert alert = new Alert(AlertType.ERROR, "Couldn't load road network! (Not a valid Jargo *.rnet?)");
          alert.showAndWait();
          this.btn_road     .setDisable(false);
          this.btn_stop     .setDisable(false);
          this.circ_status  .setFill(C_SUCCESS);
          this.lbl_status   .setText("Ready.");
          ⟨Set cursor default 186b⟩
        }
      });
    } else {
      // FD cancelled
      this.btn_road     .setDisable(false);
      this.btn_stop     .setDisable(false);
      this.circ_status  .setFill(C_SUCCESS);
      this.lbl_status   .setText("Ready.");
    }
  }
```

Uses initializeCanvas 205c, loadRoadNetworkFromFile 143d, queryEdgesCount 72d, and queryVerticesCount 71a.

## actionProb(1)

195 ⟨*actionProb(1)* 195⟩≡ (191c)

```
  void actionProb(final ActionEvent e) {
    this.btn_prob     .setDisable(true);
    this.btn_stop     .setDisable(true);
    this.circ_status  .setFill(C_WARN);
    this.lbl_status   .setText("Select *.instance...");
    FileChooser fc = new FileChooser();
    fc.getExtensionFilters().addAll(new ExtensionFilter("Problem Instance *.instance", "*.instance"));
    File pb = fc.showOpenDialog(this.stage);
    if (pb != null) {
      ⟨Set cursor wait 186a⟩
      this.prob = pb.toString();
      this.lbl_status.setText("Load '"+this.prob+"'...");
      CompletableFuture.runAsync(() -> {
        try {
          this.controller.loadProblem(this.prob);
          this.ns = this.controller.queryServersCount()[0];
          this.nr = this.controller.queryRequestsCount()[0];
          this.t0 = this.controller.query("select min (ue) from r_user where uq > 0", 1)[0];
          this.t1 = this.controller.query("select max (ue) from r_user where uq > 0", 1)[0];
          Platform.runLater(() -> {
            this.btn_prob     .setText(pb.getName());
            this.btn_client   .setDisable(false);
            this.btn_client_gtree.setDisable(false);
            this.tf_client    .setDisable(false);
            this.tf_t0        .setText(Integer.toString(t0));
            this.tf_t1        .setText(Integer.toString(t1));
            this.btn_stop     .setDisable(false);
            this.circ_status  .setFill(C_SUCCESS);
            this.lbl_status   .setText("Loaded "+pb.getName()+"(#servers="+ns+"; #requests="+nr+")");
          });
          ⟨Set cursor default 186b⟩
        } catch (Exception ee) {
          if (DEBUG) {
            System.err.println(ee.toString());
          }
          Platform.runLater(() -> {
            this.circ_status  .setFill(C_ERROR);
            this.lbl_status   .setText("Failed to load problem!");
            Alert alert = new Alert(AlertType.ERROR, "Couldn't load problem! (Not a valid Jargo instance?)");
            alert.showAndWait();
            this.btn_prob     .setDisable(false);
            this.btn_stop     .setDisable(false);
            this.circ_status  .setFill(C_SUCCESS);
            this.lbl_status   .setText("Ready.");
          });
          ⟨Set cursor default 186b⟩
        }
      });
    } else {
      // FD cancelled
      this.btn_prob     .setDisable(false);
      this.btn_stop     .setDisable(false);
      this.circ_status  .setFill(C_SUCCESS);
      this.lbl_status   .setText("Ready.");
    }
  }
```
Uses `loadProblem` 144d, `query` 68b, `queryRequestsCount` 79c, and `queryServersCount` 94b.

`actionGtree(1)`

196 ⟨*actionGtree(1)* 196⟩≡ (191c)
```
  void actionGtree(final ActionEvent e) {
    this.btn_gtree    .setDisable(true);
    this.btn_stop     .setDisable(true);
    this.circ_status  .setFill(C_WARN);
    this.lbl_status   .setText("Select *.gtree...");
    FileChooser fc = new FileChooser();
    fc.getExtensionFilters().addAll(new ExtensionFilter("G-tree *.gtree", "*.gtree"));
    File gt = fc.showOpenDialog(this.stage);
    if (gt != null) {
      ⟨Set cursor wait 186a⟩
      this.gtree = gt.toString();
      this.circ_status.setFill(C_WARN);
      this.lbl_status.setText("Load '"+this.gtree+"'...");
      CompletableFuture.runAsync(() -> {
        try {
          this.controller.gtreeLoad(this.gtree);
          Platform.runLater(() -> {
            if (this.access_path == 1) {
              this.btn_prob    .setDisable(false);
            } else if (this.access_path == 2) {
              this.btn_client .setDisable(false);
              this.btn_client_gtree.setDisable(false);
              this.tf_client   .setDisable(false);
            }
            this.btn_gtree    .setText(gt.getName());
            this.btn_stop     .setDisable(false);
            this.circ_status  .setFill(C_SUCCESS);
            this.lbl_status   .setText("Loaded "+gt.getName());
          });
          ⟨Set cursor default 186b⟩
        } catch (Exception ee) {
          if (DEBUG) {
            System.err.println("Failed: "+ee.toString());
          }
          Platform.runLater(() -> {
            this.circ_status  .setFill(C_ERROR);
            this.lbl_status   .setText("Failed to load G-tree!");
            Alert alert = new Alert(AlertType.ERROR, "Couldn't load G-tree!");
            alert.showAndWait();
            this.btn_gtree    .setDisable(false);
            this.btn_stop     .setDisable(false);
            this.circ_status  .setFill(C_SUCCESS);
            this.lbl_status   .setText("Ready.");
          });
          ⟨Set cursor default 186b⟩
        }
      });
    } else {
      // FD cancelled
      this.btn_gtree    .setDisable(false);
      this.btn_stop     .setDisable(false);
      this.circ_status  .setFill(C_SUCCESS);
      this.lbl_status   .setText("Ready.");
    }
  }
```

Defines:
 actionGtree, never used.
Uses gtreeLoad 128b.

## actionClient(1)

197 ⟨*actionClient(1)* 197⟩≡ (191c)

```
  void actionClient(final ActionEvent e) {
    this.btn_client   .setDisable(true);
    this.tf_client    .setDisable(true);
    this.btn_stop     .setDisable(true);
    this.circ_status  .setFill(C_WARN);
    this.lbl_status   .setText("Select *.jar...");
    FileChooser fc = new FileChooser();
    fc.getExtensionFilters().addAll(new ExtensionFilter("Client *.jar", "*.jar"));
    File cj = fc.showOpenDialog(this.stage);
    if (cj != null) {
      ⟨Set cursor wait 186a⟩
      this.clientjar = cj.toString();

      try {
  /*https://stackoverflow.com/questions/15720822/how-to-get-names-of-classes-inside-a-jar-file*/
  List<String> classNames = new ArrayList<String>();
  ZipInputStream zip = new ZipInputStream(new FileInputStream(this.clientjar));
  for (ZipEntry entry = zip.getNextEntry(); entry != null; entry = zip.getNextEntry()) {
    if (!entry.isDirectory() && entry.getName().endsWith(".class")) {
      String className = entry.getName().replace('/', '.');
      classNames.add(className.substring(0, className.length() - ".class".length()));
    }
  }
  /******/
        if (classNames.size() == 0) {
          Platform.runLater(() -> {
            this.circ_status  .setFill(C_ERROR);
            this.lbl_status   .setText("Bad jar!");
            Alert alert = new Alert(AlertType.ERROR, "Couldn't load client!");
            alert.showAndWait();
            this.btn_client   .setDisable(false);
            this.tf_client    .setDisable(false);
            this.btn_stop     .setDisable(false);
            this.circ_status  .setFill(C_SUCCESS);
            this.lbl_status   .setText("Ready.");
          });
          ⟨Set cursor default 186b⟩
          return;
        }
        Platform.runLater(() -> {
          this.clientclass = classNames.get(0);
          this.btn_client   .setText(cj.getName());
          this.tf_client    .setText(this.clientclass);
          this.tf_client    .setDisable(false);
          this.btn_traffic  .setDisable(false);
          this.tf_traffic   .setDisable(false);
          this.tf_t0        .setDisable(false);
          this.tf_t1        .setDisable(false);
          this.btn_startseq .setDisable(false);
          this.btn_startreal.setDisable(false);
          this.btn_stop     .setDisable(false);
          this.circ_status  .setFill(C_SUCCESS);
          this.lbl_status   .setText("Loaded "+cj.getName());
        });
        ⟨Set cursor default 186b⟩
      } catch (IOException ie) {
        Platform.runLater(() -> {
          this.circ_status  .setFill(C_ERROR);
          this.lbl_status   .setText("Bad jar!");
          Alert alert = new Alert(AlertType.ERROR, "Couldn't load client!");
          alert.showAndWait();
          this.btn_client   .setDisable(false);
          this.btn_client_gtree.setDisable(false);
          this.tf_client    .setDisable(false);
          this.btn_stop     .setDisable(false);
          this.circ_status  .setFill(C_SUCCESS);
          this.lbl_status   .setText("Ready.");
        });
        ⟨Set cursor default 186b⟩
      }
    } else {
      // FD cancelled
      this.btn_client   .setDisable(false);
      this.tf_client    .setDisable(false);
      this.btn_stop     .setDisable(false);
      this.circ_status  .setFill(C_SUCCESS);
      this.lbl_status   .setText("Ready.");
    }
  }
```

actionClientGtree(1)

198 ⟨*actionClientGtree(1)* 198⟩≡ (191c)

```
void actionClientGtree(final ActionEvent e) {
  this.btn_client_gtree.setDisable(true);
  this.btn_stop     .setDisable(true);
  this.circ_status  .setFill(C_WARN);
  this.lbl_status   .setText("Select *.gtree...");
  FileChooser fc = new FileChooser();
  fc.getExtensionFilters().addAll(new ExtensionFilter("G-tree *.gtree", "*.gtree"));
  File gt = fc.showOpenDialog(this.stage);
  if (gt != null) {
    ⟨Set cursor wait 186a⟩
    this.gtree_client = gt.toString();
    this.btn_client_gtree.setText(gt.getName());
    this.circ_status.setFill(C_WARN);
    this.lbl_status.setText("Load '"+this.gtree_client+"'...");
    Platform.runLater(() -> {
      this.btn_stop     .setDisable(false);
      this.circ_status  .setFill(C_SUCCESS);
      this.lbl_status   .setText("Loaded "+gt.getName());
    });
    ⟨Set cursor default 186b⟩
  } else {
    // FD cancelled
    this.btn_client_gtree.setDisable(false);
    this.btn_stop     .setDisable(false);
    this.circ_status  .setFill(C_SUCCESS);
    this.lbl_status   .setText("Ready.");
  }
}
```

Defines:
  actionClientGtree, never used.

## actionQuery(1)

199 ⟨*actionQuery(1)* 199⟩≡ (191c)

```
  void actionQuery(final ActionEvent e) {
    String qstr = this.txt_query.getText();
    if (qstr.length() == 0) {
      return;
    }
    int[] ncols = new int[] { 0 };
    try {
      ArrayList<String> header = new ArrayList<String>();
      int[] output = this.controller.queryQuick(qstr, ncols, header);
      int ncol = ncols[0];
      StringBuilder sb = new StringBuilder(output.length);
      for (String colname : header) {
        int resl = colname.length();
        char[] pad = new char[6 - resl];
        Arrays.fill(pad, ' ');
        sb.append(pad).append(colname);
      }
      sb.append('\n');
      for (String colname : header) {
        sb.append("------");
      }
      sb.append('\n');
      for (int row = 0; row < output.length/ncol; row++) {
        for (int col = 0; col < ncol; col++) {
          int res = output[(row*ncol + col)];
          int resl = String.valueOf(res).length();
          char[] pad = new char[6 - resl];
          Arrays.fill(pad, ' ');
          sb.append(pad).append(res);
        }
        sb.append('\n');
      }
      sb.append('\n').append("Execution time: ");
      sb.append(this.controller.getQueryDur()).append(" ms\n");
      Platform.runLater(() -> {
        this.txt_result.setText(sb.toString());
      });
    } catch (SQLException se) {
      Platform.runLater(() -> {
        this.txt_result.setText("Error: "+se.toString());
      });
    }
  }
```

Defines:
  actionQuery, used in chunk 200a.
Uses queryQuick 68c.

## actionQueryContinuous(1)

200a ⟨*actionQueryContinuous(1)* 200a⟩≡ (191c)

```
  void actionQueryContinuous(final ActionEvent e) {
    if (this.chk_continuous.isSelected()) {
      this.exe_query = Executors.newScheduledThreadPool(1);
      this.exe_query.scheduleAtFixedRate(
          () -> { actionQuery(e); }, 0, 5, TimeUnit.SECONDS);
    } else {
      this.exe_query.shutdown();
    }
  }
```

Defines:
  actionQueryContinuous(1), never used.
Uses actionQuery 199.

## actionTraffic(1)

200b ⟨*actionTraffic(1)* 200b⟩≡ (191c)

```
  void actionTraffic(final ActionEvent e) {
    this.btn_traffic   .setDisable(true);
    this.tf_traffic    .setDisable(true);
    this.tf_client     .setDisable(true);
    this.btn_stop      .setDisable(true);
    this.tf_t0         .setDisable(true);
    this.tf_t1         .setDisable(true);
    this.btn_startseq .setDisable(true);
    this.btn_startreal.setDisable(true);
    this.circ_status   .setFill(C_WARN);
    this.lbl_status    .setText("Select *.jar...");
    FileChooser fc = new FileChooser();
    fc.getExtensionFilters().addAll(new ExtensionFilter("Traffic *.jar", "*.jar"));
    File cj = fc.showOpenDialog(this.stage);
    if (cj != null) {
      ⟨Set cursor wait 186a⟩
      this.trafficjar = cj.toString();

      try {
  /*https://stackoverflow.com/questions/15720822/how-to-get-names-of-classes-inside-a-jar-file*/
  List<String> classNames = new ArrayList<String>();
  ZipInputStream zip = new ZipInputStream(new FileInputStream(this.trafficjar));
  for (ZipEntry entry = zip.getNextEntry(); entry != null; entry = zip.getNextEntry()) {
    if (!entry.isDirectory() && entry.getName().endsWith(".class")) {
      String className = entry.getName().replace('/', '.');
      classNames.add(className.substring(0, className.length() - ".class".length()));
    }
  }
  /******/
        if (classNames.size() == 0) {
          Platform.runLater(() -> {
            this.circ_status   .setFill(C_ERROR);
            this.lbl_status    .setText("Bad jar!");
            Alert alert = new Alert(AlertType.ERROR, "Couldn't load traffic!");
            alert.showAndWait();
            this.btn_traffic  .setDisable(false);
            this.tf_traffic   .setDisable(false);
            this.tf_client    .setDisable(false);
            this.btn_stop     .setDisable(false);
            this.tf_t0        .setDisable(false);
            this.tf_t1        .setDisable(false);
            this.btn_startseq .setDisable(false);
            this.btn_startreal.setDisable(false);
            this.circ_status  .setFill(C_SUCCESS);
            this.lbl_status   .setText("Ready.");
          });
          ⟨Set cursor default 186b⟩
          return;
        }
        Platform.runLater(() -> {
          this.trafficclass = classNames.get(0);
          this.tf_traffic   .setText(this.trafficclass);
          this.btn_traffic  .setText(cj.getName());
          this.tf_traffic   .setDisable(false);
          this.tf_client    .setDisable(false);
          this.btn_stop     .setDisable(false);
          this.tf_t0        .setDisable(false);
          this.tf_t1        .setDisable(false);
          this.btn_startseq .setDisable(false);
          this.btn_startreal.setDisable(false);
          this.circ_status  .setFill(C_SUCCESS);
          this.lbl_status   .setText("Loaded "+cj.getName());
        });
        ⟨Set cursor default 186b⟩
      } catch (IOException ie) {
        Platform.runLater(() -> {
          this.circ_status  .setFill(C_ERROR);
          this.lbl_status   .setText("Bad jar!");
          Alert alert = new Alert(AlertType.ERROR, "Couldn't load traffic!");
          alert.showAndWait();
          this.btn_traffic  .setDisable(false);
          this.tf_traffic   .setDisable(false);
          this.tf_client    .setDisable(false);
          this.btn_stop     .setDisable(false);
          this.tf_t0        .setDisable(false);
          this.tf_t1        .setDisable(false);
          this.btn_startseq .setDisable(false);
          this.btn_startreal.setDisable(false);
          this.circ_status  .setFill(C_SUCCESS);
          this.lbl_status   .setText("Ready.");
        });
        ⟨Set cursor default 186b⟩
      }
    } else {
      // FD cancelled
      this.btn_traffic  .setDisable(false);
      this.tf_traffic   .setDisable(false);
      this.tf_client    .setDisable(false);
      this.btn_stop     .setDisable(false);
      this.tf_t0        .setDisable(false);
      this.tf_t1        .setDisable(false);
      this.btn_startseq .setDisable(false);
      this.btn_startreal.setDisable(false);
      this.circ_status  .setFill(C_SUCCESS);
      this.lbl_status   .setText("Ready.");
    }
  }
```

## actionStartSequential(1)

```
202a  ⟨actionStartSequential(1) 202a⟩≡                                        (191c)
        void actionStartSequential(final ActionEvent e) {
          ⟨Load client 190g⟩
          ⟨Set default t0, t1 186d⟩
          this.btn_startseq .setDisable(true);
          this.btn_startreal.setDisable(true);
          this.tf_client     .setDisable(true);
          this.tf_t0        .setDisable(true);
          this.tf_t1        .setDisable(true);
          this.circ_status  .setFill(C_WARN);
          this.lbl_status   .setText("Loading '"+this.clientclass+"'...");
          ⟨Load traffic 191a⟩
          ⟨Load gtree 191b⟩
          this.t0 = Integer.parseInt(this.tf_t0.getText());
          this.t1 = Integer.parseInt(this.tf_t1.getText());
          this.controller.setClockStart(this.t0);
          this.controller.setClockEnd(this.t1);
          ⟨Add charts to chart containers 186f⟩
          this.ren_servers = new RendererOfServers(this.can_servers.getGraphicsContext2D(), this.lbl_fps, false, this
          this.ren_servers.start();
          this.ren_requests = new RendererOfRequests(this.can_requests.getGraphicsContext2D(), this.muf);
          this.ren_requests.start();
          this.circ_status  .setFill(C_SUCCESS);
          this.lbl_status   .setText("Simulation started.");
          this.exe = Executors.newScheduledThreadPool(3);
          this.cbSimulation = this.exe.schedule(() -> {
            try {
              this.controller.startSequential((status) -> {
                try {
                  this.controller.instanceExport("seq-"+String.valueOf(System.currentTimeMillis()));
                } catch (SQLException se) {
                  System.err.println("Could not export results");
                  se.printStackTrace();
                }
                Platform.runLater(() -> {
                  this.lbl_status.setText("Simulation "+(status ? "ended." : "failed."));
                });
              });
            } catch (Exception ee) {
              System.err.println("Unexepected error in startSequential");
              ee.printStackTrace();
              System.exit(1);
            }
          }, 0, TimeUnit.SECONDS);
          this.cbFetcherOfMetrics = this.exe.scheduleAtFixedRate(
              new FetcherOfMetrics(this.controller, this.lbl_status, this.pane_info, this.lu_series, this.ns, this.nr
          this.cbFetcherOfRequests = this.exe.scheduleAtFixedRate(
              new FetcherOfRequests(
                this.controller, this.muf, this.ren_requests), 0, 1, TimeUnit.SECONDS);
          this.cbFetcherOfLocations = this.exe.scheduleAtFixedRate(
              new FetcherOfLocations(
                this.controller, this.muf, this.ren_servers), 0, 1, TimeUnit.SECONDS);
        }
```

Defines:
  `actionStartSequential`, never used.
Uses `instanceExport` 40b, `setClockEnd` 60b, `setClockStart` 60a, and `startSequential` 146a.

## actionStartRealtime(1)

```
202b  ⟨actionStartRealtime(1) 202b⟩≡                                          (191c)
        void actionStartRealtime(final ActionEvent e) {
          ⟨Load client 190g⟩
```

```
    ⟨Set default t0, t1 186d⟩
    this.btn_startseq .setDisable(true);
    this.btn_startreal.setDisable(true);
    this.tf_client     .setDisable(true);
    this.tf_t0        .setDisable(true);
    this.tf_t1        .setDisable(true);
    this.circ_status  .setFill(C_WARN);
    this.lbl_status   .setText("Loading '"+this.clientclass+"'...");
    ⟨Load traffic 191a⟩
    ⟨Load gtree 191b⟩
    this.t0 = Integer.parseInt(this.tf_t0.getText());
    this.t1 = Integer.parseInt(this.tf_t1.getText());
    this.controller.setClockStart(this.t0);
    this.controller.setClockEnd(this.t1);
    ⟨Add charts to chart containers 186f⟩
    this.ren_servers = new RendererOfServers(this.can_servers.getGraphicsContext2D(), this.lbl_fps, true, this.
    this.ren_servers.start();
    this.ren_requests = new RendererOfRequests(this.can_requests.getGraphicsContext2D(), this.muf);
    this.ren_requests.start();
    this.circ_status  .setFill(C_SUCCESS);
    this.lbl_status   .setText("Simulation started.");
    this.exe = Executors.newScheduledThreadPool(3);
    this.cbSimulation = this.exe.schedule(() -> {
      try {
        this.controller.startRealtime((status) -> {
          try {
            this.controller.instanceExport("real-"+String.valueOf(System.currentTimeMillis()));
          } catch (SQLException se) {
            System.err.println("Could not export results");
            se.printStackTrace();
          }
          Platform.runLater(() -> {
            this.lbl_status.setText("Simulation "+(status ? "ended." : "failed."));
          });
        });
      } catch (Exception ee) {
        System.err.println("Unexepected error in startRealtime");
        ee.printStackTrace();
        System.exit(1);
      }
    }, 0, TimeUnit.SECONDS);
    this.cbFetcherOfMetrics = this.exe.scheduleAtFixedRate(
        new FetcherOfMetrics(this.controller, this.lbl_status, this.pane_info, this.lu_series, this.ns, this.nr
    this.cbFetcherOfRequests = this.exe.scheduleAtFixedRate(
        new FetcherOfRequests(
          this.controller, this.muf, this.ren_requests), 0, 1, TimeUnit.SECONDS);
    this.cbFetcherOfLocations = this.exe.scheduleAtFixedRate(
        new FetcherOfLocations(
          this.controller, this.muf, this.ren_servers), 0, 1, TimeUnit.SECONDS);
  }
```

Defines:
  `actionStartRealtime`, never used.
Uses `instanceExport` 40b, `setClockEnd` 60b, `setClockStart` 60a, and `startRealtime` 145c.

## actionStop(1)

203 ⟨*actionStop(1)* 203⟩≡ (191c)

```
  void actionStop(final ActionEvent e) {
    ⟨Set cursor wait 186a⟩
    if (this.controller != null) {
      this.btn_stop     .setDisable(true);
      if (this.ren_road != null) {
        this.ren_road.stop();
      }
      if (this.ren_servers != null) {
        this.ren_servers.stop();
      }
      if (this.exe != null) {
        this.exe.shutdown();
      }
      this.circ_status.setFill(C_WARN);
      this.lbl_status.setText("Close '"+this.db+"'...");
      CompletableFuture.runAsync(() -> {
        try {
          this.controller.stop((status) -> {
            Platform.runLater(() -> {
              this.lbl_status.setText("Simulation stopped.");
            });
          });
          this.controller.instanceClose();
          this.controller.gtreeClose();
          Platform.runLater(() -> {
            this.btn_new      .setDisable(false);
            this.btn_load     .setDisable(false);
            this.btn_prob     .setDisable(true);
            this.btn_prob     .setText("(empty problem instance)");
            this.prob = null;
            this.btn_road     .setDisable(true);
            this.btn_road     .setText("(empty road network)");
            this.road = null;
            this.btn_gtree    .setDisable(true);
            this.btn_gtree    .setText("(empty G-tree)");
            this.gtree = null;
            this.btn_client   .setDisable(true);
            this.btn_client   .setText("(empty client)");
            this.btn_client_gtree.setDisable(true);
            this.btn_client_gtree.setText("(empty G-tree)");
            this.btn_traffic  .setDisable(true);
            this.btn_traffic  .setText("(empty traffic)");
            this.client = null;
            this.clientjar = null;
            this.clientclass = null;
            this.traffic = null;
            this.trafficjar = null;
            this.trafficclass = null;
            this.tf_client    .setDisable(true);
            this.tf_client    .setText("");
            this.tf_traffic   .setDisable(true);
            this.tf_traffic   .setText("");
            this.tf_t0        .setDisable(true);
            this.tf_t0        .setText("");
            this.tf_t1        .setDisable(true);
            this.tf_t1        .setText("");
            this.btn_startseq .setDisable(true);
            this.btn_startreal.setDisable(true);
            this.db = null;
            this.circ_status.setFill(C_SUCCESS);
            this.lbl_status.setText("Closed instance.");
          });
          ⟨Set cursor default 186b⟩
        } catch (SQLException se) {
          System.err.println("Failure");
          System.exit(1);
        }
      });
    } else {
      ⟨Set cursor default 186b⟩
    }
  }
```

Defines:
  actionStop, never used.
Uses gtreeClose 128d, instanceClose 41a, and stop 146c.

## toggleMetric(1)

205a ⟨toggleMetric(1) 205a⟩≡ (191c)

```
  void toggleMetric(final ActionEvent e) {
  }
```

Defines:
  toggleMetric, never used.

## actionZoomCanvas(1)

205b ⟨actionZoomCanvas(1) 205b⟩≡ (191c)

```
  void actionZoomCanvas(ScrollEvent e) {
    if (e.getDeltaY() > 0) {
      this.zoom += 1;
    }
    if (e.getDeltaY() < 0) {
      this.zoom -= 1;
    }
    this.zoom = Math.max(this.zoom, 1);
    this.zoom = Math.min(this.zoom, 5);
    this.muf.setUnit(this.unit*this.zoom);
    this.can_road.setWidth(this.window_width*this.zoom);
    this.can_road.setHeight(this.window_height*this.zoom);
    this.can_servers.setWidth(this.window_width*this.zoom);
    this.can_servers.setHeight(this.window_height*this.zoom);
    this.can_requests.setWidth(this.window_width*this.zoom);
    this.can_requests.setHeight(this.window_height*this.zoom);
    if (this.ren_road != null) {
      this.ren_road.forceRender();
    }
    if (this.ren_servers != null) {
      this.ren_servers.setZoom(this.zoom);
    }
    if (this.ren_requests != null) {
      this.ren_requests.setZoom(this.zoom);
    }
    e.consume();
  }
```

Defines:
  actionZoomCanvas, used in chunk 205c.

## initializeCanvas(0)

205c ⟨initializeCanvas(0) 205c⟩≡ (192a)

```
  void initializeCanvas() {
    try {
      this.can_road     = new Canvas(this.window_width, this.window_height);
      this.can_servers  = new Canvas(this.window_width, this.window_height);
      this.can_requests = new Canvas(this.window_width, this.window_height);
      this.lbl_fps      = new Label("FPS");
      // Determine pixels-per-coordinate
      this.mbr   = this.controller.queryMBR();
      this.xunit = this.can_road.getWidth() /(double) (this.mbr[1] - this.mbr[0]);
      this.yunit = this.can_road.getHeight()/(double) (this.mbr[3] - this.mbr[2]);
      this.unit  = Math.min(this.xunit, this.yunit);
      // Set map units
      this.muf = new FetcherOfMapUnits();
      this.muf.setUnit(this.unit);
      this.muf.setLngMin(this.mbr[0]);
      this.muf.setLatMin(this.mbr[2]);
      this.muf.setLngMax(this.mbr[1]);
      this.muf.setLatMax(this.mbr[3]);
      // Add canvas to pane
      this.container_canvas_container = new Pane(
          this.can_road,
          this.can_servers,
          this.can_requests
  //        this.lbl_fps
      );
      this.container_canvas.setContent(this.container_canvas_container);
      // Register mouse event handlers
      // (can_requests is on top so it will trap all mouse events)
      this.can_requests.setOnScroll((e) -> { actionZoomCanvas(e); });
    } catch (SQLException se) {
      System.err.println("Failed with SQLException");
      Tools.PrintSQLException(se);
      return;
    }
  }
```

Defines:
`initializeCanvas`, used in chunks 193 and 194.
Uses `actionZoomCanvas` 205b, `PrintSQLException` 163e, and `queryMBR` 69b.

### `setWindowWidth(1)`

206a ⟨*setWindowWidth(1)* 206a⟩≡ (191c)

```
  void setWindowWidth(double w) {
    this.window_width = w;
  }
```

Defines:
`setWindowWidth`, used in chunk 170d.

### `setWindowHeight(1)`

206b ⟨*setWindowHeight(1)* 206b⟩≡ (191c)

```
  void setWindowHeight(double h) {
    this.window_height = h;
  }
```

Defines:
`setWindowHeight`, used in chunk 170d.

### `setStage(1)`

206c ⟨*setStage(1)* 206c⟩≡ (191c)

```
  void setStage(Stage s) {
    this.stage = s;
  }
```

Defines:
`setStage`, used in chunk 170d.

# Chapter 5

# Troubleshooting

## 5.1 Limitations

Check this list of limitations if you run into any problems.

1. **The check constraint 'C91' was violated while performing an INSERT or UPDATE on table '"APP"."CPD"'.**

   This violation occurs because a request pick-up time in the submitted schedule is earlier than the request early time. This violation might unexpectedly appear in the following scenario:

   **Example 1.** Server 1 has schedule $(0, 29247, 1, 0), (1, 0, 1, 0)$, in other words the server is idling at vertex 29247. The time is $t = 15$. At this time, Request 2 appears, with origin 29247 and destination 11353. A client algorithm produces the following new route for Server 1:

   $$(0, 29247), ..., (218, 11353), (219, 0)$$

   and the following new schedule:

   $$(0, 29247, 1, 0), (0, 29247, 0, 2), (218, 11353, 0, 2), (219, 0, 1, 0).$$

   The labeled waypoint $(0, 29247, 0, 2)$ indicating pick-up of Request 2 occurs at time $0 < 15$, producing the error.

   If the pick-up was changed from time 0 to a later time, then the error would be avoided. But be careful here, as Jargo does not allow self-referencing edges. Thus the next example would produce a C10 violation:

   **Example 2.** Let the new schedule be

   $$(0, 29247, 1, 0), (20, 29247, 0, 2), ...$$

   and the new route be

   $$(0, 29247), (20, 29247), ...$$

   The pick-up time of 20 is safely later than the early time of 15. But as self-referencing edges are not allowed, edge $(29247, 29247)$ in the route produces a C10 violation.

   To avoid the limitation, insert a vertex in the route:

   $$(0, 29247), (10, 29248), (20, 29247)$$

   Now there is no self-referencing edge.

2. **The transaction was aborted because of a deferred constraint violation: Foreign key 'F20' defined on "APP"."W" referencing constraint 'P11' defined on "APP"."W", key ".**

   This error can occur when trying to update a server's route, and the new route does not contain any waypoint in the server's existing route. The error arises because Jargo puts the first waypoint in the new route into the `t1`, `v1` columns of the route table, W, and these two columns have foreign key constraint F20 on columns `t2`, `v2` in the same table. Constraint F20 helps to enforce that `v1`, `v2` form an edge.

   The reason for putting the first waypoint into `t1`, `v1` is because the new and existing routes must align somewhere, otherwise the server will seem to "teleport" to the new route. By using `t1`, `v1` and F20, alignment is guaranteed to be possible.

3. **com.github.jargors.sim.RouteIllegalOverwriteException: Overwrite occurred!**

   This error can occur when trying to update a server's route, and the waypoints in the new route before the world time at time of update do not match the waypoints in the existing route up until this time. The reason for this error is to prevent altering the historical traveled routes. Here is an example.

| | | |
|---|---|---|
| | Time: | 35 |
| **Example 3.** | Old route: | $(0, 4815), (16, 4814), (23, 13872), (30, 13870), (37, 30028), ...$ |
| | New route: | $(30, 13870), (42, 13872), ...$ |

At $t = 35$, the server is traveling toward vertex 30028. But the new route tells it to go to vertex 13872. If the server could teleport, then the new route would be feasible. Otherwise, the new route is infeasible because the server must first visit 30028, turn back to 13870, only then visit 13872. A correct new route could be $(30, 13870), (37, 30028), (44, 13870), (56, 13872), ....$

4. **The transaction was aborted because of a deferred constraint violation: Foreign key 'F46' defined on "APP"."PD" referencing constraint 'P12' defined on "APP"."PD", key ".**

   This violation occurs because a pick-up or drop-off listed in Table CQ cannot be found in Table PD. This violiation might unexpectedly appear in the following scenario:

   **Example 4.** Server 1 has schedule

   $$(30, 20763, 0, 5142), (1141, 19903, 0, 5142), (1142, 0, 1, 0).$$

   A client algorithm prepends a new pick-up and drop-off to the front of the schedule, create a new schedule

   $$(30, 20763, 0, 2151), (1018, 14209, 0, 2151), (2006, 19903, 0, 5142), (3117, 19903, 0, 5142), (3118, 0, 1, 0).$$

   The client submits the new schedule, along with a new route

   $$(30, 20763), (34, 20764), ..., (3118, 0).$$

   When Jargo goes to update the schedule, it looks for the time in the first waypoint of the new route, in this case 30, to determine where to start overwriting the old schedule. Then, it deletes all pick-up and drop-off events from the old schedule where the event time is *greater than* this time from Table CQ. As the existing event $(30, 20763, 0, 5142)$ does not occur after the time 30, it does not get deleted from CQ. Later when Jargo goes to update Table PD with the new pick-up time for request 5142, this undeleted event in CQ produces F46 violation as it cannot find the event time in PD.

   If *greater-than-or-equals* was used instead, then the common case of computing a new route from a vehicle's origin would break because the vehicle's own "pick-up" event would be deleted!

   To avoid this limitation, do not change any existing schedule events on the first waypoint of the new route to be submitted.

5. **The statement was aborted because it would have caused a duplicate key value in a unique or primary key constraint or unique index identified by 'C105' defined on 'CQ'.**

   This violation occurs because a schedule event is inserted twice into CQ. This violation can unexpectedly occur if the first waypoint in a submitted route has existing schedule events, and those events are also found in the submitted schedule. To avoid this issue, do not include existing events on this waypoint in the submitted schedule.

6. **The check constraint 'C102J' was violated while performing an INSERT or UPDATE on table '"APP"."CQ"'.** occurs when submitting a schedule with multiple events on a waypoint and capacity is not violated.

   If the events on the waypoint are ordered in a way that capacity violation occurs, then this error will occur, even if other events on the waypoint "balance out" the violation. For example, consider a server with 3-capacity that has two prior pick-up events and no drop-off events. As it approaches waypoint $w_i$, it has a load of $q = -3 + 2 = -1$. Now on this waypoint, there are two more pick-ups and one drop-off. The total load after the waypoint is $q = -1 + 2 - 1 = 0$, causing no violation. However, if the pick-ups both occur before the drop-off, Jargo will detect that a violation did occur and throw the error. Considering only the two pick-ups, the load becomes $q = -1 + 2 = +1$, which is a violation. The way to avoid this violation is to order the drop-offs in front of the pick-ups.

## 5.2 Bugs

1. **java.sql.SQLException: The external routine is not allowed to execute SQL statements.**

   If `-Dderby.stream.error.extendedDiagSeverityLevel=0` is set, the Derby error log may additionally contain a statement such as `Error compiling prepared statement: SELECT 1 FROM "APP"."CPD" ... validateCheckConstraint=e2cdc...`. The cause of the error is unknown. The error sometimes appears when using sequential mode.

   Minimizing check constraint violations might avoid the issue. No bug report has been filed with Derby yet because no minimum-reproducible example is available.

# Appendix A

# Appendix: Brief Primer on Relations

Relations can be defined in terms of sequences and tuples. A sequence is an ordered list of elements. In this document, the integer sequence from $i$ to $j$ is written as $i..j$. The sequence

$$(a_i)_{i \in 1..n} = a_1, a_2, ..., a_{n-1}, a_n$$

is written as $a_1..a_n$ or simply $a$ (without any subscript). The number of elements in $a$ is called the length of $a$ and is expressed as $|a|$. A copy $b$ of sequence $a$ but with some elements removed is called a subsequence. Sequence $b$ is called a substring of $a$ only if some $k$ exists such that

$$b = a_{1+k}..a_{|b|+k},$$

in other words the elements in $b$ form a contiguous subsequence of $a$. Sequence $a$ is called a tuple if each element of $a$ is labeled. A labeled element is called a component. A function mapping an element based on its position in the tuple to a label is called a labeling scheme. Each component has a domain from which the component takes its value, for example the set of real numbers. A tuple of length $m$ is called an $m$-tuple. A 2-tuple is called a pair. A tuple definition is written here as its labels surrounded by parentheses with the domains given. Label names are written in `typewriter` script to avoid confusion with positional indices.

**Example 5.** The sequence $a = a_{\mathtt{x}}, a_{\mathtt{y}}$ is a 2-tuple with components named x and y. The labeling scheme for $a$ maps $1 \rightarrow \mathtt{x}$ and $2 \rightarrow \mathtt{y}$, with the integers 1 and 2 referring to the position of the elements. A possible definition for $a$ could be $a := (\mathtt{x}, \mathtt{y}), a_{\mathtt{x}} \in \mathbb{R}, a_{\mathtt{y}} \in \{\text{pi}, \text{Euler}\}$, and a possible value for $a$ could be

$$a = 3.14, \text{pi}.$$

A set of unique $m$-tuples with the same labeling scheme is called an $m$-ary relation, or simply relation. Two operators can be applied onto relations[1]. The selection operator $\sigma_P(R)$ is a function that returns a subset $R' \subseteq R$ such that predicate $P(R'_i)$ is true for each tuple $R'_i \in R'$. The projection operator $\pi_L(R)$ is a function that returns a copy $R'$ of $R$ such that each tuple $R'_i \in R'$ is distinct, and only components with a label in set $L$ are included.

Observe that an $m$-tuple is an $m$-ary relation with one element. The projection operator thus naturally applies to tuples. For instance, see that for $R = R_{\mathtt{x}}, R_{\mathtt{y}} := (\mathtt{x}, \mathtt{y})$, the x component is extracted with $R_{\mathtt{x}} = \pi_{\mathtt{x}}(R)$ and the y component is extracted with $R_{\mathtt{y}} = \pi_{\mathtt{y}}(R)$.

[1]Jargo's ridesharing model does not use joins.

# Appendix B

# Appendix: List of Chunks

⟨*..round to nearest meter* 159c⟩ 159b, 159c
⟨`Client` *constructor* 151a⟩ 150a, 151a
⟨`Client` *member variables* 150c⟩ 150a, 150c
⟨`Client` *methods* 151b⟩ 150a, 151b, 151c, 151d
⟨`Command` *methods* 167a⟩ 165a, 167a
⟨`Communicator` *constructor* 148d⟩ 148a, 148d
⟨`Communicator` *member variables* 148c⟩ 148a, 148c
⟨`Communicator` *methods* 149a⟩ 148a, 149a, 149b, 149c, 149d
⟨`Controller` *constructor* 140a⟩ 134a, 140a
⟨`Controller` *member variables* 135e⟩ 134a, 135e
⟨`Controller` *methods* 140c⟩ 134a, 140c, 141a, 141b, 141c, 142a, 142b
⟨`Desktop` *methods* 170c⟩ 170a, 170c
⟨`DesktopController` *member variables* 173c⟩ 172a, 173c, 174a, 174b, 175, 176, 178, 180, 181, 182, 183, 185
⟨`DesktopController` *methods* 191c⟩ 172a, 191c, 192a
⟨`Storage` *constructor* 131b⟩ 129a, 131b
⟨`Storage` *member variables* 130b⟩ 129a, 130b, 130c, 130d, 130e, 131a
⟨`Storage` *methods* 131c⟩ 129a, 131c, 132, 133a, 133b
⟨`Tools` *constructor* 158b⟩ 157a, 158b
⟨`Tools` *member variables* 158a⟩ 157a, 158a
⟨`Tools` *methods* 158c⟩ 157a, 158c, 158d, 158e, 158f
⟨`Traffic` *constructor* 155d⟩ 155a, 155d
⟨`Traffic` *member variables* 155c⟩ 155a, 155c
⟨`Traffic` *methods* 155e⟩ 155a, 155e, 155f
⟨*actionAbout(1)* 192d⟩ 191c, 192d
⟨*actionClient(1)* 197⟩ 191c, 197
⟨*actionClientGtree(1)* 198⟩ 191c, 198
⟨*actionGitHub(1)* 192c⟩ 191c, 192c
⟨*actionGtree(1)* 196⟩ 191c, 196
⟨*actionLoad(1)* 193⟩ 191c, 193
⟨*actionNew(1)* 192e⟩ 191c, 192e
⟨*actionProb(1)* 195⟩ 191c, 195
⟨*actionQuery(1)* 199⟩ 191c, 199
⟨*actionQueryContinuous(1)* 200a⟩ 191c, 200a
⟨*actionQuit(1)* 192b⟩ 191c, 192b
⟨*actionRoad(1)* 194⟩ 191c, 194
⟨*actionStartRealtime(1)* 202b⟩ 191c, 202b
⟨*actionStartSequential(1)* 202a⟩ 191c, 202a
⟨*actionStop(1)* 203⟩ 191c, 203
⟨*actionTraffic(1)* 200b⟩ 191c, 200b
⟨*actionZoomCanvas(1)* 205b⟩ 191c, 205b
⟨*Add charts to chart containers* 186f⟩ 186f, 202a, 202b
⟨*Admin: cacheRoadNetworkFromDB(0)* 42d⟩ 42d, 141c

# Appendix C

# Appendix: List of Identifiers