# Eng 3006 Software Engineering & Project

*Shortest Path Algorithm for Material Transportation*

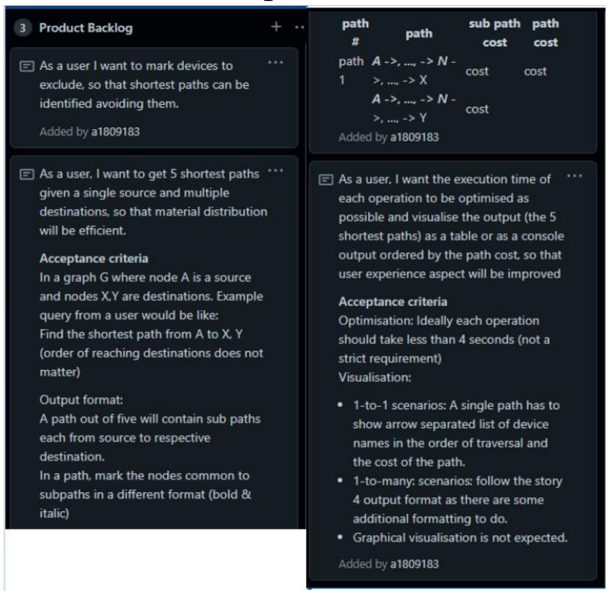
## Sprint Retrospective 4 of Group Path 12

**SEP Group Path 12 Members:** Adam Rebes (a1793912), William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

October 2023

# Snapshots

## Snapshot 3.1



3 Product Backlog

As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

Added by a1809183

As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.

**Acceptance criteria**

In a graph G where node A is a source and nodes X,Y are destinations. Example query from a user would be like:
Find the shortest path from A to X, Y (order of reaching destinations does not matter)

Output format:
A path out of five will contain sub paths each from source to respective destination.
In a path, mark the nodes common to subpaths in a different format (bold & italic)

| path # | path | sub path cost | path cost |
|---|---|---|---|
| path 1 | ***A*** ->, ..., -> ***N*** ->, ..., -> X | cost | cost |
| | ***A*** ->, ..., -> ***N*** ->, ..., -> Y | cost | |

Added by a1809183

As a user, I want the execution time of each operation to be optimised as possible and visualise the output (the 5 shortest paths) as a table or as a console output ordered by the path cost, so that user experience aspect will be improved

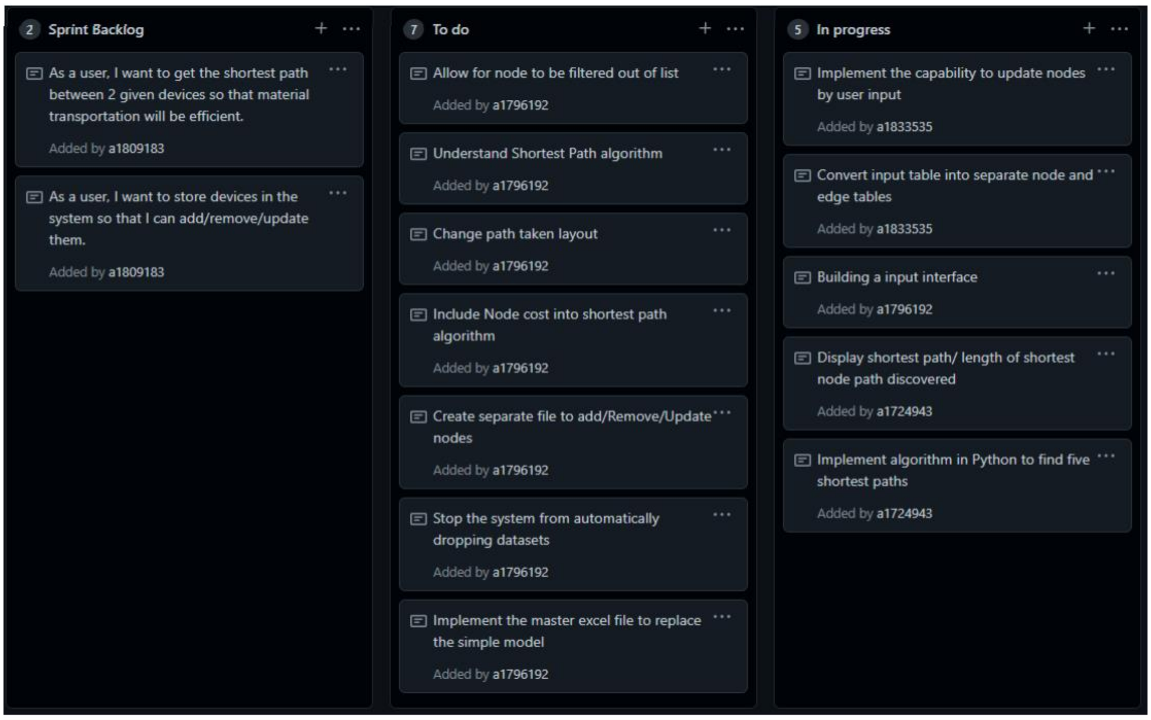
**Acceptance criteria**

Optimisation: Ideally each operation should take less than 4 seconds (not a strict requirement)
Visualisation:

- 1-to-1 scenarios: A single path has to show arrow separated list of device names in the order of traversal and the cost of the path.
- 1-to-many: scenarios: follow the story 4 output format as there are some additional formatting to do.
- Graphical visualisation is not expected.

Added by a1809183

*Figure 1: Product Backlog*



2 Sprint Backlog

As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
Added by a1809183

As a user, I want to store devices in the system so that I can add/remove/update them.
Added by a1809183

7 To do

Allow for node to be filtered out of list
Added by a1796192

Understand Shortest Path algorithm
Added by a1796192

Change path taken layout
Added by a1796192

Include Node cost into shortest path algorithm
Added by a1796192

Create separate file to add/Remove/Update nodes
Added by a1796192

Stop the system from automatically dropping datasets
Added by a1796192

Implement the master excel file to replace the simple model
Added by a1796192

5 In progress

Implement the capability to update nodes by user input
Added by a1833535

Convert input table into separate node and edge tables
Added by a1833535

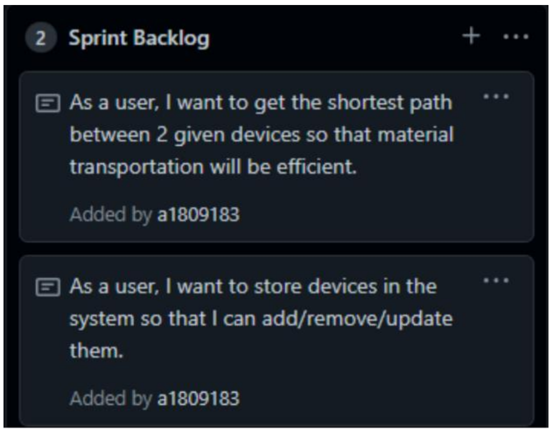
Building a input interface
Added by a1796192

Display shortest path/ length of shortest node path discovered
Added by a1724943

Implement algorithm in Python to find five shortest paths
Added by a1724943

*Figure 2: Task Board*

## Sprint Backlog and User Stories



2 Sprint Backlog

As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.

Added by a1809183

As a user, I want to store devices in the system so that I can add/remove/update them.

Added by a1809183

*Figure 3: Sprint Backlog*

The user story that the group will dedicate work resources to is “As a user, I want to get 5 shortest paths given a single source and multiple destinations so that material distribution will be efficient” as well as fixing the deliverables for the previous user story. The group will dedicate some members to fix the calculation of the cost of the shortest paths to include the cost of the nodes.

The current subtasks include issues that were raised with older deliverables with the purpose of the subtasks to rectify the issues. The subtasks focus on the correct calculation of costs in the shortest path algorithm, creating procedures to add and delete edge/nodes and decoupling code into modular queries. As shown in Figure 2, our current sprint backlog task revolves around updating, and optimizing existing implementations to fix bugs mentioned by the advisor. The bugs as mentioned include the dataset automatically deletes itself after every execution, node cost is not included in the path algorithm, including the master excel file into the SQL server instead of the existing basic implementation. Future goals are to understand the shortest path algorithm in SQL in detail to be able to start the deliverable mark devices for exclusion.

## Definition of Done

• If code has been written, it must follow the coding standards defined in the initial report.

• If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.

• If code has been written, it must pass localization testing defined by the programmer, automated testing written by the team, and automated regression testing for any future alterations to ensure correct behaviour.

• If code has been written, it must ensure functionality continues to work on the build server, and local device.

• The created sprint task must be linked towards the product backlog which in turn must be linked towards the user story.

• Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to the user.

• For any implementation of code or system architecture, sufficient documentation is required to justify its relation with the user story, and configuration changes must be well-documented.

• The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering standards, and be signed off by the product owner.

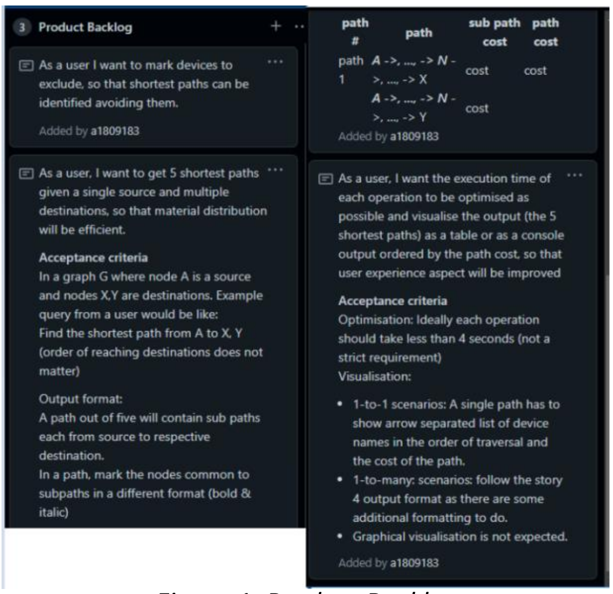
• Verification and validation of key scenarios are considered to ensure basic functionality before in-depth processes are considered for implementation.

• The final product must address the provided user stories to a reasonable standard, being able to at the very least produce a shortest path between two specified devices.

## Summary of changes

Since the last snapshot in week 8, we have split members to work on deliverables “Add/Remove Path” and “Shortest Path Algorithm”. From a given master list which is made by the user, the SQL code can automatically construct node and edge tables and implement an ID number to such a path and node point. Filtering methods have been used to prevent duplicate edge and node ID to be presented to avoid complications when referencing back to the dataset. The user can now determine which source and destination point the shortest path algorithm uses by naming the points in the respective variable slots. A model of the shortest path algorithm, in both default SQL functionality and custom functionality has been implemented. This current method can find the total cost of the edge path and represent which path was taken. Edge points can be changed in cost where the shortest path algorithm can detect the change and recalculate path.

## Snapshot 3.2



3 **Product Backlog**

As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

Added by a1809183

As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.

**Acceptance criteria**
In a graph G where node A is a source and nodes X,Y are destinations. Example query from a user would be like:
Find the shortest path from A to X, Y (order of reaching destinations does not matter)

Output format:
A path out of five will contain sub paths each from source to respective destination.
In a path, mark the nodes common to subpaths in a different format (bold & italic)

| **path #** | **path** | **sub path cost** | **path cost** |
|---|---|---|---|
| path 1 | *A* ->, ..., -> *N* ->, ..., -> X | cost | cost |
| | *A* ->, ..., -> *N* ->, ..., -> Y | cost | |

Added by a1809183

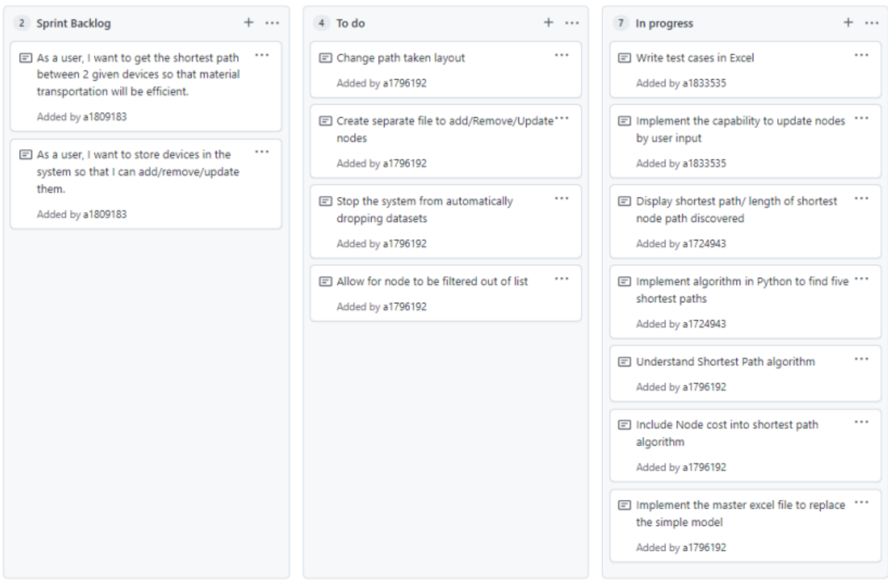
As a user, I want the execution time of each operation to be optimised as possible and visualise the output (the 5 shortest paths) as a table or as a console output ordered by the path cost, so that user experience aspect will be improved

**Acceptance criteria**
Optimisation: Ideally each operation should take less than 4 seconds (not a strict requirement)
Visualisation:

- 1-to-1 scenarios: A single path has to show arrow separated list of device names in the order of traversal and the cost of the path.
- 1-to-many: scenarios: follow the story 4 output format as there are some additional formatting to do.
- Graphical visualisation is not expected.

Added by a1809183

*Figure 1: Product Backlog*



2 Sprint Backlog

- As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
  Added by a1809183
- As a user, I want to store devices in the system so that I can add/remove/update them.
  Added by a1809183

4 To do

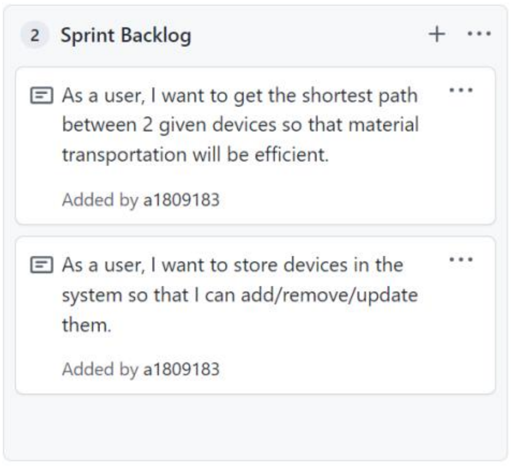
- Change path taken layout
  Added by a1796192
- Create separate file to add/Remove/Update nodes
  Added by a1796192
- Stop the system from automatically dropping datasets
  Added by a1796192
- Allow for node to be filtered out of list
  Added by a1796192

7 In progress

- Write test cases in Excel
  Added by a1833535
- Implement the capability to update nodes by user input
  Added by a1833535
- Display shortest path/ length of shortest node path discovered
  Added by a1724943
- Implement algorithm in Python to find five shortest paths
  Added by a1724943
- Understand Shortest Path algorithm
  Added by a1796192
- Include Node cost into shortest path algorithm
  Added by a1796192
- Implement the master excel file to replace the simple model
  Added by a1796192

*Figure 2: Task Board*

## Sprint Backlog and User Stories



2 Sprint Backlog

- As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
  Added by a1809183
- As a user, I want to store devices in the system so that I can add/remove/update them.
  Added by a1809183

*Figure 3: Sprint Backlog*

The user story that the group will dedicate work resources is “As a user, I want to get 5 shortest paths given a single source and multiple destinations so that material distribution will be efficient” as well as fixing the deliverables for the previous user story. The group continues to dedicate some members to fix the calculation of the cost of the shortest paths to include the cost of the nodes. Additionally, some group members are assigned to develop testing strategies and documentation.

The current subtasks include issues that were raised with older deliverables with the purpose of the subtasks to rectify the issues. The subtasks focus on the correct calculation of costs in the shortest path algorithm, creating procedures to add and delete edge/nodes and decoupling code into modular queries. As shown in Figure 2, our current sprint backlog task revolves around updating, and optimizing existing implementations to fix bugs mentioned by the advisor. The bugs as mentioned include the dataset automatically deletes itself after every execution, node cost is not included in the path algorithm, including the master excel file into the SQL server instead of the existing basic implementation. The shortest path algorithm in SQL is beginning to be better understood, and the capability to import data from Excel into relevant programs and formats is being implemented.

Adam Rebes – a1793912

## Definition of Done

• If code has been written, it must follow the coding standards defined in the initial report.

• If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.

• If code has been written, it must pass localization testing defined by the programmer, automated testing written by the team, and automated regression testing for any future alterations to ensure correct behaviour.

• If code has been written, it must ensure functionality continues to work on build server, and local device.

• The created sprint task must be linked towards the sprint backlog which in turn must be linked towards the user story.

• Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to the user.

• For any implementation of code or system architecture, sufficient documentation is required to justify its relation with the user story, and configuration changes must be well-documented.

• The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering standards, and be signed off by the product owner.

• Verification and validation of key scenarios are considered to ensure basic functionality before in-depth processes are considered for implementation.

• The final product must address the provided user stories to a reasonable standard, being able to at the very least produce a shortest path between two specified devices.

## Summary of changes

So far in this sprint, the team has successfully implemented functions to import the factory data from the Excel document in both Python and SQL. The team has refined its Python program, focusing primarily on using lists and a depth first search algorithm. Additionally, the team has furthered its progress with implementing functions for developing node and edge tables, and finding five shortest paths in SQL. Though this has not been completed, it has developed significantly throughout the sprint so far, and some group members have written several test cases in hopes of having some working code before the end of this sprint.

***"I attended the sprint review/planning meeting on Thursday the 19th of October with the tutor Dileepa Pitawela"***

# Sprint

### What went well?

A large amount of progress was achieved in this sprint. Four procedures were made in MSSQL to add and remove edges and nodes, including costs, which allows for changes to the factory layout. We also made changes to the script that creates the node and edge tables to work of a master table which is created by importing a csv of the data containing the factory layout keeping the default import settings. We also made other changes such as adding a usage column to the node and edge tables where 0 indicates the node or edge is free and 1 indicates it is in use with the usage currently being updatable by using the update node and update edge procedures. We also managed to get our shortest path algorithm to return the 5 shortest paths accounting for both node and edge costs. It currently takes input by changing the source and destination variables within the script and can also avoid a device by making a change to an avoid variable. We also created various test cases with corresponding diagrams of what case should look like. Finally, the python version of the shortest path algorithm is now able to return the five shortest paths as well.

### What could be improved?

Currently the gap isn't fully bridged between the script for creating the node and edge tables from the csv data master table and the shortest path algorithm. Both teams used different tables with differing attributes with the shortest paths algorithm using one large table instead of the form the node and edge tables which would allow it to take advantage of node and edge table functionality. For now, we addressed this by creating a temp table by joining the node table twice onto the edge table so that we'd have all the data in a similar form to that in the shortest path algorithm and replace the corresponding attribute names.

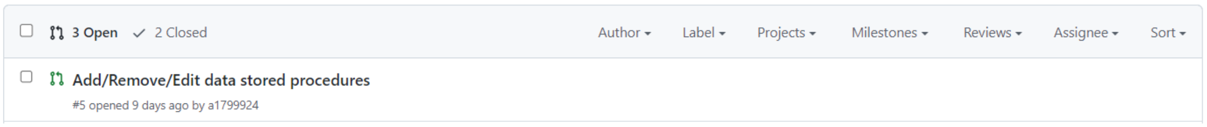
Also, our shortest path algorithm doesn't currently take advantage of the usage attribute to avoid certain node and edges when calculating a shortest path. Instead, right now it can only avoid a single device at a time by putting its name into the avoid variable in the script. This is how source and destination are set as well, which should ideally instead take from the attribute in the data stating whether a node is a source and attribute, however our current code only supports 1 source and destination currently. Also, on the python side, the code doesn't work for datasets too large due to the sheer amount of recursion required, so the algorithm may need to be changed. The python implementation also currently doesn't connect to a database, instead relying on the data file and user input for updates but with no way to persist between iterations.

### What will the group commit to improve in the next sprint?

We need to make sure the shortest path algorithm utilises the usage attribute of the table to avoid any nodes and edges in use when calculating shortest paths. We also need to be able to return paths for multiple sources and destinations and take advantage of whether the node or edge is set as source or destination in the data. Also, we need to fully bridge the gap between the table creation scripts and the shortest path algorithm so that there's no issues and inconsistencies between the two and the algorithm can be more efficient. In addition, we need checks in place to make sure that the script doesn't try to create tables that already exist or use tables or databases that might not exist, and that no data is dropped between runs, to fully utilise the persistent database required for the final product. We'll need to decide if the python implementation is something which still may be useful and if so it would need to be able to handle the large factory layout and also connect to a database, so data persists between different run throughs of the program.

Adam Rebes – a1793912

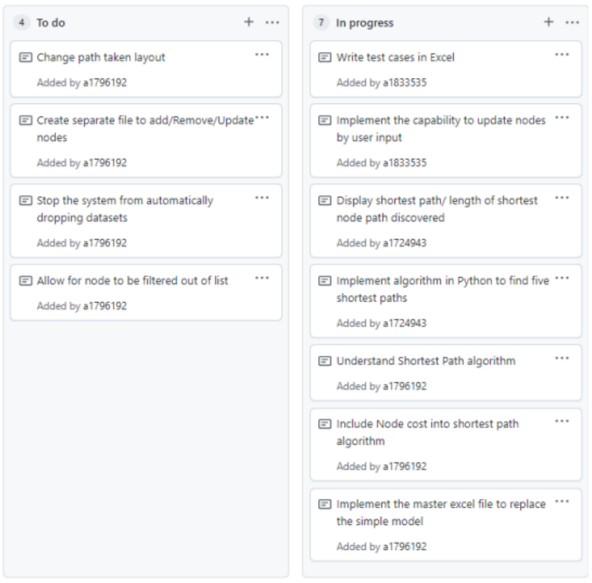
## Sprint Progress



3 Open ✓ 2 Closed Author Label Projects Milestones Reviews Assignee Sort

Add/Remove/Edit data stored procedures

#5 opened 9 days ago by a1799924



4 To do

Change path taken layout
Added by a1796192

Create separate file to add/Remove/Update nodes
Added by a1796192

Stop the system from automatically dropping datasets
Added by a1796192

Allow for node to be filtered out of list
Added by a1796192

7 In progress

Write test cases in Excel
Added by a1833535

Implement the capability to update nodes by user input
Added by a1833535

Display shortest path/ length of shortest node path discovered
Added by a1724943

Implement algorithm in Python to find five shortest paths
Added by a1724943

Understand Shortest Path algorithm
Added by a1796192

Include Node cost into shortest path algorithm
Added by a1796192

Implement the master excel file to replace the simple model
Added by a1796192

During the first part of the sprint, I worked with another team member on creating the four stored procedures in MSSQL. The four procedures were Update Node, Update Edge, Delete Node and Delete Edge. The update node procedure allows the user the put in a string with the name of a node and a cost, whether it's a; source, dest or neither, and whether it's in use or not. If the node doesn't exist, it will create it. For creating edges, this is the same except that the edge is created by putting the names of the two nodes the edge is between. A pull request was made on the github with an earlier version of the procedures.

In the second half of the sprint, we worked on importing data from a csv into MSSQL using the default import settings to create a master table and then creating the node and edge tables from that master table. From there we created a temporary table to work with the current version of the shortest path algorithm.