Adding Water Path Capabilities to QWAT Databases

Bogdan Vaduva, Honoriu Valean
Automation Department, Technical University of Cluj-Napoca, Cluj-Napoca, Romania

***Abstract*—The main purpose of this article is to show how to extend an existing open source database, namely QWAT (Acronym from Quantum GIS Water Plugin), by using pgRouting (PostgreSQL routing extension) in order to achieve the ability to find the water flow in a water network. The water path in a water network is a key information needed by any water supplying company for different activities such as customer identification, meter the water flow or isolating areas of the water network. In our environment an open source database was used and that database didn't have any means to identify the water path, so our research is intended into that direction. Once a water path is found, our next goal was to show that identifying customers for a water supplying company is just a click away (by using no directional graphs). Another key information needed by the water supplying companies is to know which valves should be closed in order to shut off the water for an area of the water network. As result, the second purpose of the article is to show how to identify the necessary valves, to be closed or open, in order to shut off or on the water (within the pipe network).**

***Keywords*—Relational database; graphs; water network; water path; open source; QWAT**

I. Introduction

Water supplying companies all over the world needs to know who their customers are and this task is a very important one, for reasons like: knowing to whom they will invoice, knowing how much water was used and if a leak occurs to identify which valves needs to be closed in order to shut off the water. In a previous paper we presented a way of predicting leakage in QWAT[1] databases, but identifying and predicting leakage is not enough, because, any existing or future leakage has a negative impact on the water supplying company's image [1].

QWAT databases are relational databases [2] that models water network pipes, by keeping information like: pipe attributes and geographical position, valve attributes and geographical position, meters, hydrants, network elements, subscribers, leaks. Pipes are kept in a table (within a schema called qwat_od) and have a set of attributes from whom we will focus on two, first node (fk_node_a) and second node (fk_node_b).

In the abstract of this paper, we said that we want to show a way of finding the water path within QWAT databases, but if the pipe table does have a first and a second node, why QWAT model doesn't have a water path function until the final consumer? Or does it? The answer is given by QWAT designers, into their documentation, where they presented how the QWAT model should be used. In that documentation they recommend that pipes should be broken whenever the pipes change their material, function, type or diameter. Other recommendations are:

- Pipes should only be coupled to the right of each intersection with another pipe (Fig. 1).
- Pipes should only be coupled to the right of each intersection with a branch (Fig. 1).
- Pipes should not be coupled in any case to the right of a private branch (Fig. 1).

Those recommendations were put into a picture as in Fig. 1.

If a QWAT database is filled the way described above, the water path should be available, but not actually to the final user/consumer, because, usually, water supplying companies have their own software for customer management and invoicing. We have to outline that, at the time this article was written, there wasn't any function within QWAT database, that allowed to find the water path or extracts the customers for some parts/sections of the water network and we came up with some proposals formulated into a document in 2017. [3].

When it comes to valves they also did some recommendations (Fig. 2):

- Do not cut the pipe on the right of each valve.
- Place the valves on the vertices of the pipe line.

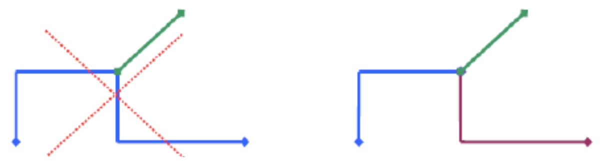
Fig. 1. Example of How to Break Pipes at Intersections in QWAT Databases.

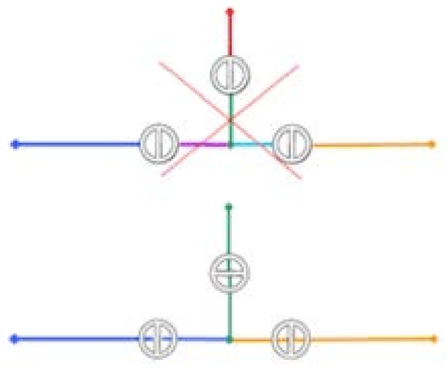
Fig. 2. Example of How to Place Valves in QWAT Databases.

In our case we begin using the QWAT model and we start entering data into that model, which uses a PostgreSQL [4]

[1] QWAT – Acronym for QGIS Water Plugin – http://www.qwat.org

database server. After a few months, we found that even if we somehow link customers, to a water distribution pipe, we can't actually differentiate between customers and all of that, because of the initial recommendations and the way our water network was drew. In our case, the colleagues from GIS (Geographic Information Systems) department, responsible with drawing the water network, were not splitting the pipes every time a pipe intersects a private branch.

We have to clarify what an intersection means from a GIS standing point: two or more pipes that actually connect to each other and not pipes that traverse other pipes (on top or under).

On other hand, getting outside on the field and looking at an actual private branch connection, we found that private branches are directly coupled to the distribution pipe (Fig. 3).

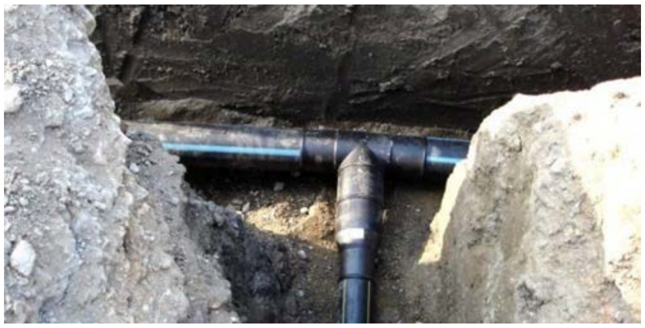

Fig. 3. Example of How a Private Branch is Actually Connected to the Distribution Pipe.

The current paper will continue with the following chapters: Database Background, Database Proposed Changes, Applied Research, Results and Conclusions.

II. Database Background

We started to study the QWAT model and see how and if, can be changed in order to accommodate our needs.

The QWAT model has a table called "pipe" which holds information for each water pipe. That information refers to attributes like: pipe id, pipe function, install method, material, bedding, precision, installation year, node a, node b, parent id, geometry and a few other that are not relevant to our paper. Let's consider a real street, which has a distribution pipe and each private branch is connected to that distribution pipe the way presented in Fig. 3. From a QWAT database standing point, the data is as in Fig. 4 and Table I.

Fig. 4 presents a distribution pipe, on which we placed, one node of the private branches, on its vertices (distribution pipe's vertices). That means that distribution pipe is not fragmented as in Fig. 1 right section and it means that we can't have a path (water flow) from node 38578 (placed on a private branch) to node 38529 (placed on distribution pipe), for example. In order to be able to do that a parallel water network should be created and kept within QWAT database. By having that parallel water network, we will actually create a graph that will contain nodes and edges for describing the water flow.

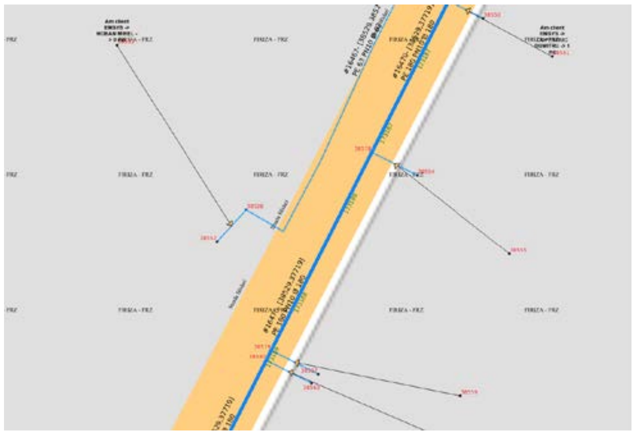

Fig. 4. Example of How a Private Branch is Actually Connected to the main Pipe.

TABLE I. Example of Data Kept in the Pipe Table

Pipe ID	Pipe Table				
	Function	***Material***	***Node A***	***Node B***	**…**
16470	Distribution pipe	Steel	38529	37719	…
18900	Private branch	Polyethylene	38580	38560	…
17898	Private branch	Polyethylene	38579	38557	…
14567	Private branch	Polyethylene	38578	38554	…
….					

In the current database model, we can achieve this water path goal only by fragmenting the distribution pipe every time it intersects a private branch. Doing so in a real-life scenario is unrealistic because of the amount of work and can't be achieved.

III. Database Proposed Changes

Analyzing the above data, we came with some initial conclusions and to do list, such as:

1) We need to (somehow) connect private branches to the distribution pipe without redrawing the water network pipes.

2) We need to know/change the valve state in order to be able to have a water path to the final consumer.

3) We need to match a private branch to a consumer.

4) We need to be able to show the water path from the source(s) to any consumer. Knowing that, we will be able to determine which valves should be closed in order to shut off or on the water to a specific customer.

We stated above that we need to address at least four database improvements in order to make it water path friendly. The first improvement on that list: "connect private branches to the distribution pipe without redrawing the water network pipes", it is the one that will allow us to determine the water flow within the water network and further to achieve the other goals on our list.

The first thought we had was to somehow split the distribution pipe, but in real world those distribution pipes are not split (Fig. 3) and our next idea was to build a parallel water network that will have the desired format. The newly parallel water network will be fragmented as in Fig. 5. Furthermore, that parallel network should be created without asking the user to redraw the network in the format presented in Fig. 1.

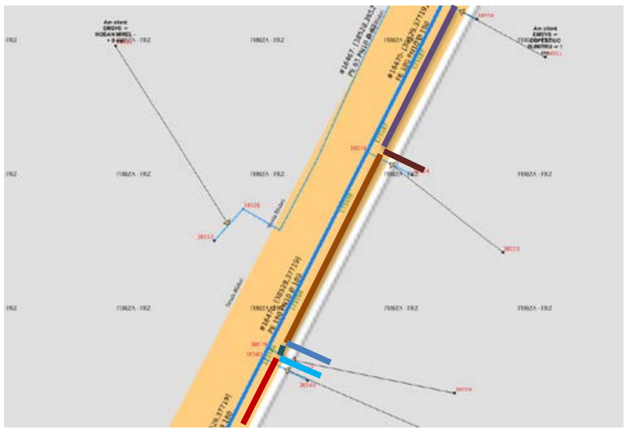

Fig. 5. Example of How we want to Build the Parallel Water Network. We used different Colors just to show that we will have Different Segments.

The parallel network should be built at the same time the user draws the water network in it's usually way, Fig. 4 and 5, where the user does not split the distribution pipe every time it intersects a private branch but places one end of a private branch on the vertices of the distribution pipe. In addition to this goal we figured out that a refresh button that creates that parallel water network for the current bounding box is acceptable and probably necessary.

Once the parallel network will be built we planned to add an existing PostgreSQL extension, called pgRouting [5] to QWAT database.

The first change to the QWAT model was to add a new table, which we called it "pipe_reference" and has the following fields:

- id, integer – an id for each pipe segment that will be generated/created,
- fk_pipe, integer – a reference to the initial pipe,
- fk_node_a, integer – a reference to the first node of the new segment,
- fk_node_b, integer – a reference to the second node of the new segment,
- geometry, geometry – the geometry of the newly created segment.

The reason behind the "pipe_reference" table is to keep the parallel water network saved, for a later use. At this point, we designed the "pipe_reference" table, but that one needed data and we planned to populate it with the help of a new function, called "fn_pipe_reference_update". Further we will present what that function does.

We imagined the function to have as input parameter the ID of the pipe (distribution pipe) that we want to split (ex. 16470 – Table I, Fig. 5). The function works as follow:

- For the input ID we delete all the records from "pipe_reference" having the fk_pipe field equal to the input parameter.
- We select all the pipes (usually private branches) that have one end placed on or near the vertices of the inputted pipe. For all the selected pipes we keep only the nodes placed on the vertices of the inputted pipe.
- Using all the nodes placed on the inputted pipe (distribution pipe), together with the first and second node of it (distribution pipe), we insert the missing rows into "pipe_reference" table (Table II).

TABLE II. EXAMPLE OF WHAT THE PIPE_REFERENCE TABLE WILL HOLD

Pipe Reference ID	**Pipe Reference**			
	Fk_pipe	***Node A***	***Node B***	***Geometry***
1	16470	38529	38577	…
2	16470	38577	38578	…
3	16470	38578	38579	…
4	16470	38579	38580	…
5	16470	38580	38581	…
6	16470	38581	38582	…
7	16470	38582	42086	…
8	16470	40286	37719	…
…				

The next envisioned step was to make a union between the content of the "pipe_reference" table with the records from the "pipe" table for whom the ids cannot be found in any of "fk_pipe" values of "pipe_reference" table. By doing so, we will have the parallel water network (automatically generated), labeled in green, that will be pgRouting friendly (Fig. 6).

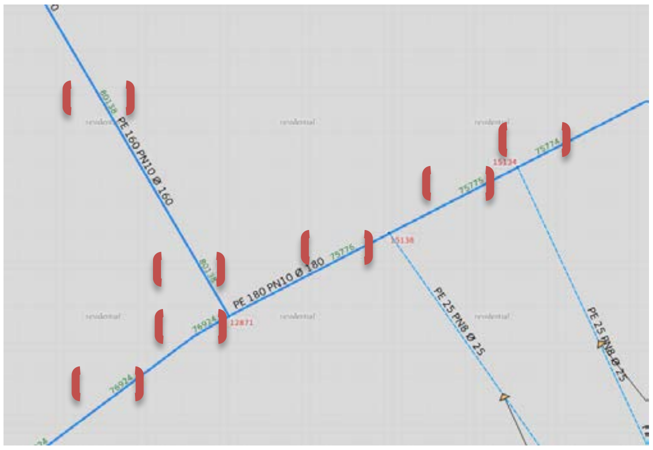

Fig. 6. Example of How the Parralel Water Network was Overlapped with the Pipe Table. Observe the Green Labels in between Orange Brackets on Top of some Pipes.

IV. Applied Research

After envisioning the parallel water network design, we moved forward by adding the new “pipe_reference” table into the QWAT database and the function named “fn_pipe_reference_update” that will construct / fill the parallel network. At the same time, we installed pgRouting in our database server. The new extension, gave us some new routing functions, from which we used “pgr_dijkstra”. This function returns the shortest path using Dijkstra algorithm. In 1956 Edsger Dijkstra created a graph search algorithm (graph with non-negative edge path costs) for determining the shortest path [6]. Because the function works on graphs with non-negative edge path costs and because there are valves on pipes (which could be open or closed) we also added a function which we called “fn_element_valve_status”. This function checks, if there are valves on a pipe and if one valve is closed on that pipe we consider the cost as being negative.

The example of running the function between node 38557 and node 38529 can be viewed in Fig. 7.

To this point, we addressed only the first and second item on our to do list, but we said that we want have a match between a private branch and a consumer and thus the path from the water source to the final user.

On our map (Fig. 7), we have outlined that each private branch serves a household. At some point the owners can change and we have to take that into consideration. On other hand the information about consumers is kept in a different database that is not by default accessible to QWAT database.

One of the last changes done to our QWAT database was to link the data about consumers into QWAT by adding a new table called “qwat_od_subscriber_location” which has two fields fk_subscriber and fk_location. The first field is a foreign key to “subscriber” table, which is native to QWAT model. The second field is a foreign key to “location” view, in a foreign database that keeps the consumer information and invoices. Regarding the foreign databases that keeps the consumer information and could be linked to QWAT, the only restrictions are related to the fact that those databases should be addressable by a PostgreSQL feature called Foreign Data Wrappers. The “location” view should always give us the information about the current consumer for a specific location.

We built that function that displays basic information about the current consumer at a specific location and the result can be seen in Fig. 7.

We succeed in solving another point in our to do list, but one item was still on that list. It is about the ability to find out if the water flows from one source of the water network to one final consumer.

Knowing which the clients are will allow the water supplying company, to send customized information, to its customers and thus improving its image. Water can flow in either directions on a pipe and as result, we have a no directional graph when it comes to water network pipes. We have implemented the extraction of this graph, in JAVA, on the server side of our web-based application. The result of this tool can be seen in Fig. 8.

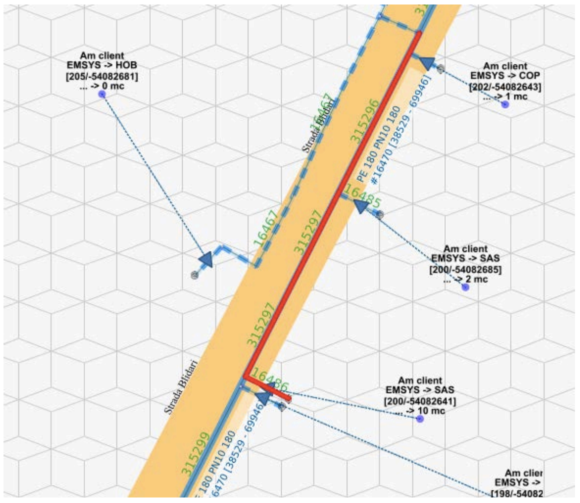

Fig. 7. Example of the Path between Node 38557 and 38529.

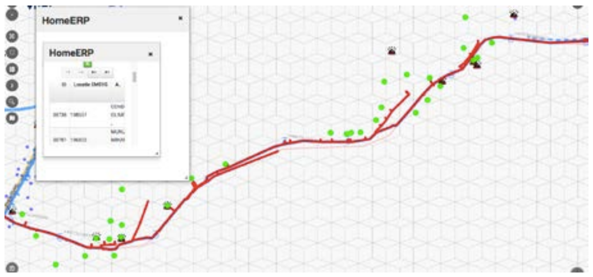

Fig. 8. Example of Determing/Identifying Clients/Customers.

To solve the last item in our to do list, we created a new function within QWAT database that we called “fn_valve_to_close”. The new function takes as input parameter, the subscriber, which translates as the end node of consumer’s private branch and determines all the water paths from the sources, defined for that water network, to that subscriber/consumer. For all the paths we outline the valves and we color the closest one with green. The result of running the function, on subscriber corresponding to the node 38557 can be seen in Fig. 9.

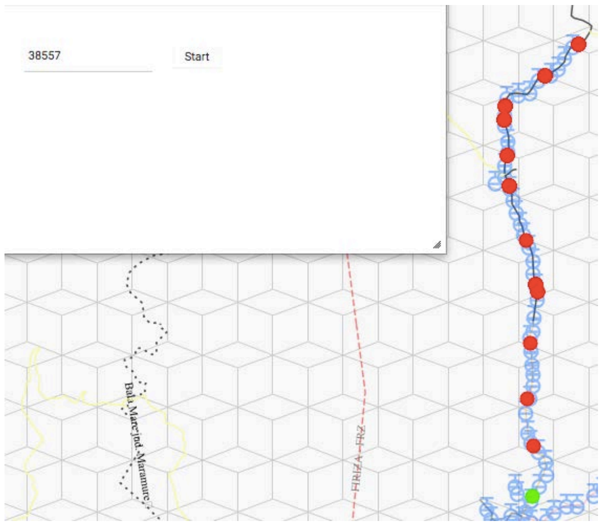

Fig. 9. Example of the Valves that Needs to be Closed for Node 38557.

We have to make a note here, related to “pipe_reference” table. The “pipe_reference” table has to be kept in synchronization with the “pipe” table, which means, that every time a user makes changes, to a pipe geometry, the “pipe_reference” table has also to be updated, to keep up with the modifications.

Regarding the hydraulic analysis that can be done on the water network [8] we have to note that are applications that already do that very well. From those applications we will mention the one called EPANET[2] (software application used throughout the world to model water distribution systems) given freely by the government of USA [9] [10].

V. Result

We showed in our article how an existing open source water network model (QWAT model) can be changed in order to accommodate new requirements, but it was only one of the steps we’ve done to improve the QWAT model.

Our way of extending the open source model, namely QWAT, it’s not the only way, and other persons embraced our ideas and extended QWAT model in a slightly different way, which brought us joy to see that our work has been appreciated [7].

Lately, another idea came to our attention, which is the possibility to use graph databases for our processes [11] [12]. We know that graph databases are suitable for recommendation engines and those can be used in our environment, namely QWAT. Before presenting our envisioned workflow, we have to outline that our relational database data is already in a friendly graph format and already uses a graph extension called pgRouting. All these will probably allow us in the future, to export our data to graph databases and to extract more information out of it. At this point and this time, we formulated the following algorithm to be used to export data to a graph database [13]:

- choose a graph database – only once.
- define case scenarios.
- specify relations or define new views to be exported.
- export the data to graph databases.
- run case scenarios to find out results.
- import results into the relational database and make use of them.

By doing the above steps in an automated fashion we will allow users to focus onto the results and thus maximizing the amount of information extracted from a relational database. We plan to test our proposed use of graph databases in the nearest future and see how much we can benefit out of it.

The work presented in the current paper shows how to find the water path in a water network, modeled in an open source database (QWAT), but there are commercial databases that also do that. One of those commercial databases is proposed by ESRI (Environmental Systems Research Institute). Regarding the performance of our model compared to ESRI’s model we can’t tell much because of the costs involved in installing the ESRI application.

We used our model for identifying the customers in a Romanian city called Sighetu Marmatiei located in the northern part of Romania. The city of Sighetu Marmatiei has about 20000 customers and from those customers only about 9000 are placed on the map and linked to private branches. The time to extract those city customers using our model is around 2-3 minutes which is a reasonable time.

VI. Conclusion

The current paper shows that an open source database model of a water network can be extended to accommodate new functions and those are similar to the functions from a commercial database. We achieved that purpose by adding tables, functions to an existing relational database, namely QWAT. The new tables and functions allowed us to identify the basic water flow of a water network but that flow did not take in consideration pressure zones or pumps [14]. On other hand our paper didn’t go into the hydraulic analysis of the water network.

An improvement that easily can be done by any water supplying company, is to add electric valve actuators, either to every consumer or to valves within the water network and to each actuator add a Wi-Fi circuit breaker. Connecting those circuit breakers to the Internet and further to QWAT model, the water supplying companies will be able to shut on or off water to its clients, remotely [15].

In conclusion our paper presented a way of solving a problem for a water supplying company by using open source databases and their features.

References

[1] Bogdan Vaduva, Honoriu Valean - Water pipes leak prediction in QWAT databases, ICSTCC 2021, Iasi, Romania, 2021.

[2] H. Darwen, An Introduction to Relational Database Theory, United Kingdom: Bookboon. com, 2010.

[3] Internet - https://github.com/qwat/qwat-data-model/issues/171 - 2017.

[4] Hans-Jurgen Schonig - Mastering PostgreSQL 11: Expert Techniques to Build Scalable, Reliable, and Fault-tolerant Database Applications, 2nd Edition.

[5] Internet – www.pgrouting.org – last access in September 2021.

[6] Schulz, Frank & Wagner, Dorothea & Weihe, Karsten. (1999). Dijkstra’s Algorithm On-Line: An Empirical Case Study from Public Railroad Transport. Algorithm Engineering.

[7] Internet - https://github.com/benoitblanc - last accessed in October 2021.

[8] Naser Moosavian, Mohammad Reza Jaefarzadeh, "Hydraulic Analysis of Water Distribution Network Using Shuffled Complex Evolution", Journal of Fluids, vol. 2014, Article ID 979706, 12 pages, 2014. https://doi.org/10.1155/2014/979706.

[9] G. VenkataRamanaDr.aCh.V.S.S.SudheerbB.Rajasekhar - Network Analysis of Water Distribution System in Rural Areas using EPANET - Procedia Engineering, Volume 119, 2015, Pages 496-505.

[10] Rai, R. K. and Lingayat, Prashant, Analysis of Water Distribution Network Using EPANET (February 22, 2019). Proceedings of Sustainable Infrastructure Development & Management (SIDM) 2019, Available at SSRN: https://ssrn.com/abstract=3375289 or http://dx.doi.org/10.2139/ssrn.3375289.

[11] R. De Virgilio, A. Maccioni, and R. Torlone. Converting relational to graph databases. In GRADES, 2013.

[2] Internet – https://www.epa.gov

[12] Konstantinos Xirogiannopoulos, Udayan Khurana, Amol Deshpande - GraphGen: Exploring Interesting Graphs in Relational Data, Proceedings of the VLDB Endowment, Volume 8, 2014-2015.

[13] Ahmad Shahzad, Frans Coenen - Automated Generation of Graphs from Relational Sources to Optimise Queries for Collaborative Filtering – 2020.

[14] Mrs. Vaidya Deepali R., Mali Sandip T. - Pressure driven approach in water distribution network analysis: A Review - The International journal of analytical and experimental modal analysis, Volume XI, Issue VII, July/2019, ISSN NO: 0886-9367.

[15] Rosiberto Gonçalves, Jesse J. M. Soares and Ricardo M. F. Lima - An IoT-Based Framework for Smart Water Supply Systems Management - Future Internet 2020, 12, 114; doi:10.3390/fi12070114.