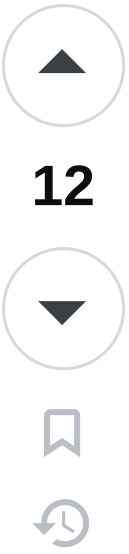
# Best Way to Store/Access a Directed Graph

Asked 16 years, 2 months ago Modified 15 years, 6 months ago

Viewed 5k times



I have around 3500 flood control facilities that I would like to represent as a network to determine flow paths (essentially a directed graph). I'm currently using SqlServer and a CTE to recursively examine all the nodes and their upstream components and this works as long as the upstream path doesn't fork alot. However, some queries take exponentially longer than others even when they are not much farther physically down the path (i.e. two or three segments "downstream") because of the added upstream complexity; in some cases I've let it go over ten minutes before killing the query. I'm using a simple two-column table, one column being the facility itself and the other being the facility that is upstream from the one listed in the first column.

I tried adding an index using the current facility to help speed things up but that made no difference. And, as for the possible connections in the graph, any nodes could have multiple upstream connections and could be connected to from multiple "downstream" nodes.

It is certainly possible that there are cycles in the data but I have not yet figured out a good way to verify this (other

than when the CTE query reported a maximum recursive count hit; those were easy to fix).

So, my question is, am I storing this information wrong? Is there a better way other than a CTE to query the upstream points?

rdbms common-table-expression directed-graph

Share

Improve this question

Follow

edited May 30, 2009 at 10:33

Michael Dorfman

**4,100** ● 1 ● 25 ● 29

asked Oct 10, 2008 at 15:33

Michael Todd

**17k** ● 4 ● 51 ● 70

What indexes do you have on the data? – tvanfosson Oct 10, 2008 at 15:41

Are you sure that there are no cycles in the graph (even ones accidentally introduced?) 3500 rows is not a very large number, especially for SQL Server. – Cervo Oct 10, 2008 at 15:46

Well actually if it is a network it could be 3500 * 3500 records, because one facility is not one row. – Tomas Pajonk Dec 7, 2008 at 10:51

## 6 Answers

Sorted by: Highest score (default)

6

The best way to store graphs is of course to use a native graph db :-)

Take a look at neo4j. It's implemented in Java and has Python and Ruby bindings as well.

I wrote up two wiki pages with simple examples of domain models represented as graphs using neo4j: assembly and roles. More examples are found on the domain modeling gallery page.

Share Improve this answer

Follow

answered Dec 7, 2008 at 10:14

nawroth

**4,341** ● 1 ● 27 ● 22

> Whether a graph database is the way to go depends on the use cases involved. Are there going to be a number of concurrent writes? Does storage need to scale horizontally? Are operations being done which would be the most time efficient on a denormalized representation of the data? Etc.
> – Eric Walker Jul 5, 2010 at 2:41

4

I know nothing about flood control facilities. But I would take the first facility. And use a temp table and a while loop to generate the path.

```
-- Pseudo Code
TempTable (LastNode, CurrentNode, N)
```

```
DECLARE @intN INT
SET @intN = 1



INSERT INTO TempTable(LastNode, CurrentNode, N)
  -- Insert first item in list with no up
stream items...call this initial condition
  SELECT LastNode, CurrentNode, @intN
  FROM your table
  WHERE node has nothing upstream



WHILE @intN <= 3500
BEGIN
     SEt @intN = @intN + 1
    INSERT INTO TempTable(LastNode,
CurrentNode, N)
      SELECT LastNode, CurrentNode, @intN
      FROM your table
      WHERE LastNode IN (SELECT CurrentNode
FROM TempTable WHERE N = @intN-1)



  IF @@ROWCOUNT = 0
       BREAK
```

If we assume that every node points to one child. Then this should take no longer than 3500 iterations. If multiple nodes have the same upstream provider then it will take less. But more importantly, this lets you do this...

SELECT LastNode, CurrentNode, N FROM TempTable ORDER BY N

And that will let you see if there are any loops or any other issues with your provider. Incidentally 3500 rows is not that much so even in the worst case of each provider pointing to a different upstream provider, this should not take that long.

Share Improve this answer

Follow

answered Oct 10, 2008 at 15:57

Cervo

**3,082** ● 1 ● 25 ● 27

I should also add that I am doing this because I suspect that your structure has a cycle in it and that is why it is taking too long. By looking at this query you should be able to see which nodes are repeated on different levels. – Cervo Oct 10, 2008 at 16:02

It's absolutely possible that I have cycles. I've implemented your pseudo-code and there are 33,147 results, but I'm not sure what I should be looking for to determine if there's a cycle issue. – Michael Todd Oct 10, 2008 at 16:23

The problem if you have cycles is that you never stop. Flood Control A->Flood Control B->Flood Control C->Flood Control A->Flood Control B->Flood Control C->..... How do you know when to stop? if you blindly follow the path with a CTE you'll never stop – Cervo Oct 10, 2008 at 17:54

SELECT LastNode, CurrentNode, @intN FROM your table WHERE LastNode IN (SELECT CurrentNode FROM TempTable WHERE N = @intN-1) AND LastNode NOT IN (SELECT CurrentNode FROM TempTable) Above will stop... – Cervo Oct 10, 2008 at 17:55

What you should look at in the original query is any CurrentNode that is repeated. SELECt CurrentNode FROM TempTable GROUP BY CurrentNode HAVING COUNT(*) > 1 – Cervo Oct 10, 2008 at 17:56

3

Traditionally graphs are either represented by a matrix or a vector. The matrix takes more space, but is easier to process(3500x3500 entries in your case); the vector takes less space(3500 entries, each have a list of who they connect to).

Does that help you?

Share Improve this answer

Follow

answered Oct 10, 2008 at 15:44

Paul Nathan

40.2k ● 30 ● 120 ● 215

2

i think your data structure is fine (for SQL Server) but a CTE may not be the most efficient solution for your queries. You might try making a stored procedure that traverses the graph using a temp table as a queue instead, this should be more efficient.

the temp table can also be used to eliminate cycles in the graph, though there shouldn't be any

Share Improve this answer

Follow

answered Oct 10, 2008 at 15:49

Steven A. Lowe

61.1k ● 19 ● 135 ● 204

1

Yes (maybe). Your data set sounds relatively small, you could load the graph to memory as an adjacency matrix or adjacency list and query the graph directly - assuming you program.

As far as on-disk format, DOT is fairly portable/popular among others. It also seems pretty common to store a list of edges in a flat file format like:

```
vertex1 vertex2 {edge_label1}+
```

Where the first line of the file contains the number of vertices in the graph, and every line after that describes edges. Whether the edges are directed or undirected is up to the implementor. If you want explicit directed edges, then describe them using directed edges like:

```
vertex1 vertex2
vertex2 vertex1
```

Share Improve this answer

Follow

answered Oct 10, 2008 at 15:51

oz10

158k ● 27 ● 98 ● 129

0

My experiences with storing something like you described in a SQL Server database:

I was storing a distance matrix, telling how long does it take to travel from point A to point B. I have done the naive representation and stored them directly into a table called distances with columns A,B,distance,time.

This is very slow on simple retreival. I found it is lot better to store my whole matrix as text. Then retreive it into memory before the computations, create an matrix struxture in memory and work with it there.

I could provide with some code, but it would be C#.

Share Improve this answer

Follow

answered Dec 7, 2008 at 10:58

Tomas Pajonk

**5,202** ● 8 ● 43 ● 51