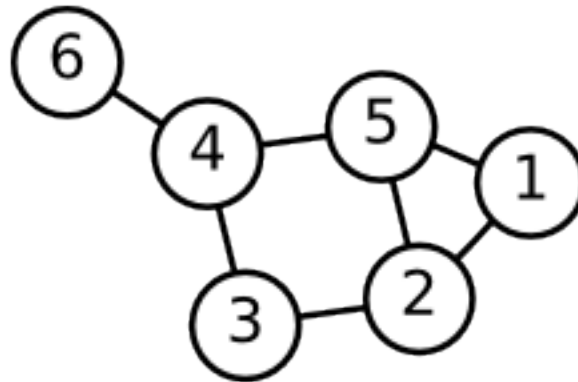


OQGRAPH Computation Engine for MySQL, MariaDB & Drizzle



Arjen Lentz

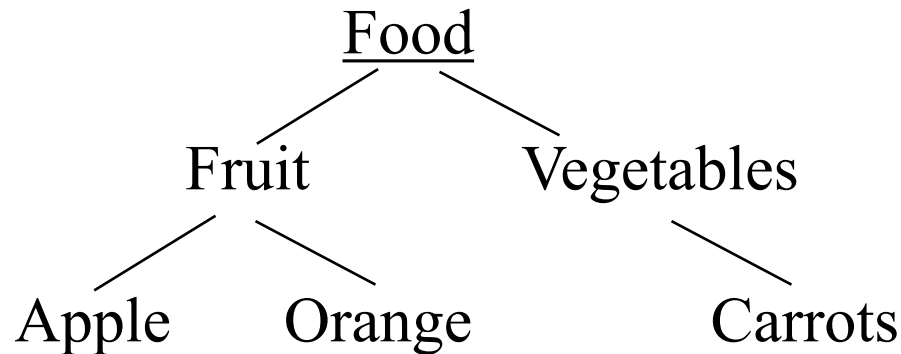
graph@openquery.com

<http://openquery.com/graph>



Hierarchies (Trees)

- In a *hierarchy* or *tree*, each node has
 - one parent node (unless it's the “root” node)
 - zero or more child nodes

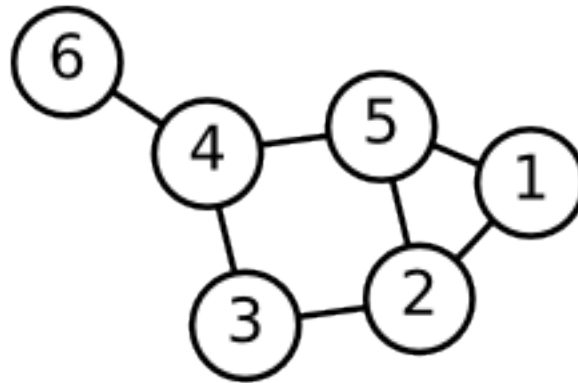


- Examples
 - menu structures
 - organisation tree (Dilbert reports to Pointy-Haired Boss)



Graphs (Networks)

- In a *graph* or *network*
 - each node has an arbitrary number of links to other nodes
 - links (edges) may be directional (digraph) or two-way
 - parts of the network may not be connected
 - there is no root/origin/base, necessarily



- Examples
 - friend-of-a-friend networks
 - matrix organisations (people reporting to two managers)

Types of Questions

- Trees
 - Who reports to Pointy-Haired Boss?
 - How many people report directly or indirectly to PHB
 - What's the path from root to here?
- Graphs
 - Does Claire Danes connect to Kevin Bacon, and how?
 - What is the shortest path from A to B?



Hierarchies & Graphs in SQL/RDBMS

- They don't fit particularly well
- Various tree models, each with limitations
 - adjacency model
 - either fixed max depth, or recursive queries
 - Oracle has `CONNECT BY PRIOR`
 - SQL-99 has `WITH RECURSIVE ... UNION ...`
 - nested set
 - complex
 - recursive queries to find path-to-root
 - materialised path
 - unrelational and ugly, but very functional
- Graphs
 - programatically, walk step-by-step in adjacency model.



What is OQGRAPH & WhoDunnit

- technically a storage engine
 - Concept by Arjen Lentz
 - for MySQL 5.0, 5.1 and above
 - for MariaDB 5.1
 - for Drizzle
- Mk.II implementation by
 - Antony Curtis
 - Arjen Lentz
- Licensing
 - GPLv2+
- Support, Engineering, Enhancements
 - Open Query



A Computation Engine...

- is not a general purpose storage engine
 - unlike MyISAM, InnoDB, PBXT, Maria
- looks like a table from the user perspective
- has a very different architecture internally
- does not, technically, operate in terms of
 - a specified # of rows you insert and retrieve
 - actually storing all columns it presents to you
 - having the indexes its table structure indicates
- could be regarded as a “magic view”



Installation

- For MySQL 5.0, it needs to be compiled in
 - binary tarballs at ourdelta.org
 - will be included in -sail (bleeding edge) builds from 5.0.87-d10
 - **SHOW GLOBAL VARIABLES LIKE 'have_oqgraph' ;**
- For MySQL/MariaDB 5.1, it's a plugin
 - likely a mariadb-oqgraph package from MariaDB 5.1.39
 - via ourdelta.org repositories
 - when installed, will automatically be loaded/enabled
 - or build your own against exactly the same source tree (even debug flags are critical)
 - **INSTALL PLUGIN oqgraph SONAME 'oqgraph_engine' ;**
 - **SHOW PLUGINS ;**
 - **SHOW STORAGE ENGINES ;**



OQGRAPH Table Structure

```
CREATE TABLE db.tblname (  
    latch    SMALLINT    UNSIGNED NULL,  
    origid   BIGINT      UNSIGNED NULL,  
    destid   BIGINT      UNSIGNED NULL,  
    weight   DOUBLE      NULL,  
    seq      BIGINT      UNSIGNED NULL,  
    linkid   BIGINT      UNSIGNED NULL,  
    KEY (latch, origid, destid) USING HASH,  
    KEY (latch, destid, origid) USING HASH  
    ) ENGINE=OQGRAPH;
```



Inserting some data

- We actually just insert edges (links)
 - *origid* -> *destid*
 - optional *weight* (**DEFAULT 1**)
 - none of the other columns actually exists!
 - it's a digraph so if you want two-way (1,2) also insert (2,1)
- `INSERT INTO foo (origid,destid) VALUES
 (1,2) , (2,3) , (2,4) ,
 (4,5) , (3,6) , (5,6) ;`

Selecting edges

- `SELECT * FROM foo;`

latch	origid	destid	weight	seq	linkid
NULL	1	2	1	0	NULL
NULL	2	3	1	1	NULL
NULL	2	4	1	2	NULL
NULL	4	5	1	3	NULL
NULL	3	6	1	4	NULL
NULL	5	6	1	5	NULL

Now Let's Do Magic! (shortest path)

- `SELECT * FROM foo
WHERE latch=1 AND origid=1 AND destid=6;`

latch	origid	destid	weight	seq	linkid
1	1	6	NULL	0	1
1	1	6	1	1	2
1	1	6	1	2	3
1	1	6	1	3	6

- `SELECT GROUP_CONCAT(linkid ORDER BY seq) AS path
FROM foo WHERE latch=1 AND origid=1 AND destid=6 \G`

path: 1,2,3,6



Other searches

- Which paths lead to node 4?

```
SELECT GROUP_CONCAT(linkid) AS list
  FROM foo WHERE latch=1 AND destid=4 \G
```

```
list: 1,2,4
```

- Where can I get to from node 4?

```
SELECT GROUP_CONCAT(linkid) AS list
  FROM foo WHERE latch=1 AND origid=4 \G
```

```
list: 6,5,4
```



Other operations

- See docs for **latch 0** and **latch NULL**
- **latch 1** Dijkstra's shortest path: $O((V + E) \cdot \log V)$
 - O is the order function.
 - $O(1)$ is constant time, irrespective of amount of data.
 - $O(N)$ is linear time, e.g. time increases with amount of data
 - V is number of vertices (nodes)
 - E is number of edges (links).
- **latch 2** Breadth-first search: $O(V + E)$
- Other algorithms coming (many more!)
- **SQL UPDATE** and **DELETE** also possible
 - within reason: don't change the path you're walking...

Making it prettier - joins

- `INSERT INTO people VALUES
 (1,'pearce'), (2,'hunnicut'), (3,'potter'),
 (4,'hoolihan'), (5,'winchester'), (6,'mulcahy');`
- `SELECT GROUP_CONCAT(name ORDER BY seq) AS path
 FROM foo JOIN people ON (foo.linkid = people.id)
 WHERE latch=1 AND origid=1 AND destid=6 \G`

`path: pearce,hunnicut,potter,mulcahy`

Tree of Life

- From tolweb.org, see info in example directory
- `SOURCE tol.sql`
- `-- create tol_tree oqgraph table`
- `INSERT INTO tol_tree (origid,destid)`
`SELECT parent,id FROM tol WHERE parent IS NOT NULL;`
- `INSERT INTO tol_tree (destid,origid)`
`SELECT parent,id FROM tol WHERE parent IS NOT NULL;`
- `SELECT COUNT(*) AS edges FROM tol_tree \G`

`edges: 178102`

Finding Homo Sapiens

- `SELECT GROUP_CONCAT(name ORDER BY seq
SEPARATOR ' -> ') AS path
FROM tol_tree JOIN tol ON (linkid=id)
WHERE latch=1 AND origid=1 AND destid=16421 \G`

```
path: Life on Earth -> Eukaryotes -> Unikonts  
-> Opisthokonts -> Animals -> Bilateria  
-> Deuterostomia -> Chordata -> Craniata  
-> Vertebrata -> Gnathostomata -> Teleostomi  
-> Osteichthyes -> Sarcopterygii  
-> Terrestrial Vertebrates -> Tetrapoda  
-> Reptiliomorpha -> Amniota -> Synapsida  
-> Eupelycosauria -> Sphenacodontia  
-> Sphenacodontoides -> Therapsida -> Theriodontia  
-> Cynodontia -> Mammalia -> Eutheria -> Primates  
-> Catarrhini -> Hominidae -> Homo -> Homo sapiens
```

How we relate to bananas

- `SELECT GROUP_CONCAT(name ORDER BY seq
SEPARATOR ' -> ') AS path
FROM tol_tree JOIN tol ON (linkid=id)
WHERE latch=1 AND origid=16421 AND destid=21506 \G`

path: Homo sapiens -> Homo -> Hominidae -> Catarrhini
-> Primates -> Eutheria -> Mammalia -> Cynodontia
-> Theriodontia -> Therapsida -> Sphenacodontoidea
-> Sphenacodontia -> Eupelycosauria -> Synapsida
-> Amniota -> Reptiliomorpha -> Tetrapoda
-> Terrestrial Vertebrates -> Sarcopterygii
-> Osteichthyes -> Teleostomi -> Gnathostomata
-> Vertebrata -> Craniata -> Chordata
-> Deuterostomia -> Bilateria -> Animals
-> Opisthokonts -> Unikonts -> Eukaryotes
-> Archaeplastida (Plantae) -> Green plants
-> Streptophyta -> Embryophytes -> Spermatopsida
-> Angiosperms -> Monocotyledons -> Zingiberanae
-> Musaceae

Are we related to retro viruses?

- `SELECT GROUP_CONCAT(name ORDER BY seq SEPARATOR ' -> ') AS path FROM tol_tree JOIN tol ON (linkid=id) WHERE latch=1 AND origid=16421 AND destid=57380 \G`

path: Homo sapiens -> Homo -> Hominidae -> Catarrhini
-> Primates -> Eutheria -> Mammalia -> Cynodontia
-> Theriodontia -> Therapsida -> Sphenacodontoidea
-> Sphenacodontia -> Eupelycosauria -> Synapsida
-> Amniota -> Reptiliomorpha -> Tetrapoda
-> Terrestrial Vertebrates -> Sarcopterygii
-> Osteichthyes -> Teleostomi -> Gnathostomata
-> Vertebrata -> Craniata -> Chordata
-> Deuterostomia -> Bilateria -> Animals
-> Opisthokonts -> Unikonts -> Eukaryotes
-> Life on Earth -> Viruses
-> DNA-RNA Reverse Transcribing Viruses
-> Retroviridae

Let's make a Maze in SQL

- ```
./maze 5 5
-- maze width=5 height=5
-- W exit (0,2) ofs=10
-- E exit (4,2) ofs=14
INSERT INTO maze_graph (origid,destid)
VALUES (0,1); -- (0,0) -> (1,0)
[and a lot more rows like this]
```

```
+--+--+--+--+
| |
+--+--+ +--+ +
| | |
+--+--+--+ +--+
 | |
+ +--+ + + +
| | | | |
+--+ + + + +
| | | |
+--+--+--+--+
```

# Dijkstra Mouse through Maze

- ```
SELECT GROUP_CONCAT(  
  CONCAT(  
    '(' ,  
    TRUNCATE(linkid % 5,0) ,  
    ' , ' ,  
    TRUNCATE(linkid / 5,0) ,  
    ') '  
  )  
  ORDER BY seq SEPARATOR ' -> ' ) AS path  
FROM maze_graph  
WHERE latch=1 AND origid=10 AND destid=14 \G
```

```
path: (0,2) -> (1,2) -> (2,2)  
      -> (2,3) -> (2,4) -> (3,4)  
      -> (4,4) -> (4,3) -> (4,2)
```

Base implementation (gratis)

- uses about 60 bytes per edge (rough estimate)
- behaves like MEMORY engine:
 - table-level locking for normal tables
 - no locking for temporary tables
 - no persistence
 - no transactions
- insert performance $O(\text{LOG}(N))$
- this means...
 - it's usable for menus & more, up to say a (few) million edges
 - you can use the `--init-file` option to copy/load on startup
- optional support from Open Query



If you have higher requirements

- persistence (we'd suggest SSD)
- more efficient in memory (16 bytes per edge)
- remaining fast with very large datasets

- Development contribution of \$/€ 5000, per customer
 - gain influence over roadmap: what and when
 - Suitable unrestricted license regardless of which MySQL/MariaDB version you use. Could be GPLv2+, or BSD.

- You may also want ongoing support
- other related custom engineering



Links

- Info, Docs, Support, Licensing, Engineering
 - <http://openquery.com/graph>
- Source collaboration
 - <https://launchpad.net/oqgraph>
- Binaries & Packages
 - <http://ourdelta.org>



Thank you!
Arjen Lentz
graph@openquery.com

