```
/** This file can be fed directly to a mySql command to create the
 *  empPaths table.
 *
 *  This is well tested but before you change it to wack your live data
 *  REMEMBER TO MOUNT A SCRATCH MONKEY
 *  http://en.wikipedia.org/wiki/Scratch_monkey
 *
 *  @author Grant Tegtmeier <Grant@Tegt.com>
 *  @package closure
 *  @copyright no
 */

/** Build a paths (closure) table for an existing employees table.
 *  There is a row for every path from every superior to all
 *  subordinates. It is much more efficient for SQL to maintain these
 *  rows than the "closure strings" often used. As a matter of fact
 *  it's DirtSimple. Thanks to PJ Eby there's a fine little walk
 *  through on this at:
 *  http://dirtsimple.org/2010/11/simplest-way-to-do-tree-based-queries.html
 *
 *  So first lets create the paths table for this data.
 */

DROP TABLE IF EXISTS empPaths;
CREATE TABLE `empPaths` (
  `bossId` int(11) NOT NULL COMMENT 'Employee table Boss id',
  `empId` int(11) NOT NULL COMMENT 'Employee table id',
  `distance` int(11) NOT NULL COMMENT 'distance between them',
  KEY `empId` (`empId`),
  KEY `bossId` (`bossId`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

/** Since employee is already filled make an empty copy of the
 * structure.
 */

DROP TABLE IF EXISTS empDone;
CREATE TABLE empDone LIKE employees;

/** In a more dynamic example this INSERT trigger would be added to
 *  the employee table right after it was created. That trigger would
 *  then maintain the empPaths table as it was populated (or
 *  restored). Here we simply attach it to a clone of the employee
 *  structure.
 *
 *  It's the INSERT...SELECT cross-product that's producing all of the
 *  rows. But they're little and it's really quick.
 */

DROP TRIGGER IF EXISTS insEmp;
delimiter ;;
CREATE TRIGGER insEmp AFTER INSERT ON empDone
       FOR EACH row BEGIN
           INSERT INTO empPaths (bossId, empId, distance)
                  VALUES (NEW.id, NEW.id, 0);
           IF NEW.bossId <> NEW.id THEN /* skip for the CEO */
```

```
57.                 INSERT INTO empPaths (bossId, empId, distance)
58.                       SELECT b.bossId, e.empId, b.distance + e.distance + 1
59.                             FROM empPaths AS b, empPaths AS e
60.                             WHERE b.empId=NEW.bossId AND e.bossId=NEW.id;
61.             END IF;
62.         END;
63. ;;
64. delimiter ;
65.
66. /** Now run that trigger for every row in the employee table. */
67.
68. INSERT INTO empDone SELECT * FROM employees;
69.
70. /** Since it's a clone we'll just use the empPaths table with the
71.  *  original employees table. So let's drop this one. N.B. in the
72.  *  mySql 5.1 I'm running there are problems if this is done with a
73.  *  TEMPORARY table. Seems not to like cleaning up a TEMPORARY with a
74.  *  trigger attached. So we'll just do it by hand.
75.  */
76.
77. DROP TABLE empDone;
```