CSE 510 - Database Management System Implementation

Spring 2017

Phase III

Due Date: Midnight, April 26th

1 Goal

The version of the MiniBase I have distributed to you implements various modules of a relational database management system. Our goal this semester is to use these modules of MiniBase as building blocks for implementing an *graph DBMS*.

2 Project Description

The following is a list of tasks that you need to perform for this phase of the project. Note that getting these working may involve other changes to other modules not described below. You will be provided with test data that you can use to verify your code before submitting it.

Note: At the end of each query, the program should output the selected query plan and the number of disk pages that were read and written (seperately) at each step of the operation.

2.0 Task 0: Making the Database Files Persistent

- Implement a strategy to make the database files persistent (if you have not done so in the earlier phase).

2.1 Task 1: New Index Nested Loop Join Operator in the `iterator` Package

- Implement new index nested loop join operators for the following operations (if you have not done so in the earlier phase):

$$Node \bowtie^{source}_{edge\_condition} Edge$$

$$Node \bowtie^{dest}_{edge\_condition} Edge$$

$$Edge \bowtie^{source}_{node\_condition} Node$$

$$Edge \bowtie^{dest}_{node\_condition} Node$$

  The index nested loop operators should use the appropriate index file to access the inner relations (i.e., right relation) to implement the join operation.

2.2 Task 2: New Sort-Merge Join Operator in the `iterator` Package

- Implement new sort-merge join operator for the following operations (if you have not done so in the earlier phase):

$$Edge \bowtie_{dest\_node=source\_node} Edge$$

The sort-merge join operator should join the edges based on shared source/dstination nodes.

2.3 Task 3: Path Expression Operator, Type 1

- Implement a new path expression operator, which receives as input a node ID, NID, and a path expression of the form

$$PE1 \leftarrow NID/(NN/)^*NN$$

where

$$NN \leftarrow (Node\_Label|Node\_Descriptor)$$

and returns an iterator over the IDs of the nodes in the tail of each path that satisfies the path expression.

2.4 Task 4: Path Expression Operator, Type 2

- Implement a new path expression operator, which receives as input a node ID, NID, and a path expression of the form

$$PE2 \leftarrow NID \ /_{EN} \ (/_{EN})^*$$

where

$$EN \leftarrow (Edge\_Label|Max\_Edge\_Weight)$$

and returns an iterator over the IDs of the nodes in the tail of each path that satisfies the path expression.

2.5 Task 5: Path Expression Operator, Type 3

- Implement a new path expression operator, which receives as input a node ID, NID, and a path expression of the form

$$PE3 \leftarrow NID \ //_{Bound}$$

where

$$Bound \leftarrow (Max\_Num\_Edges|Max\_Total\_Edge\_Weight)$$

and returns an iterator over the IDs of the nodes in the tail of each path that satisfies the path expression.

2.6 Task 6: Path Expression Query, Type 1

- Implement a path expression query, which receives a path expression of the form

$$PQ1a \leftarrow NN/(NN/)^*NN$$

where

$$NN \leftarrow (Node\_Label|Node\_Descriptor)$$

and returns the labels of the nodes in the head and tail of each path that satisfies the path expression.

$PQ1b$ returns the same data, but sorts the results in the labels of the source and tail labels.

$PQ1c$ returns the same data, but only distinct head/tail node label pairs.

2.7 Task 7: Path Expression Query, Type 2

- Implement a new path expression query, which receives a path expression of the form

$$PQ2a \leftarrow NN \ /_{EN} \ (/_{EN})^*$$

where

$$NN \leftarrow (Node\_Label|Node\_Descriptor)$$

$$EN \leftarrow (Edge\_Label|Max\_Edge\_Weight)$$

and returns the labels of the nodes in the head and tail of each path that satisfies the path expression.

$PQ2b$ returns the same data, but sorts the results in the labels of the source and tail labels.

$PQ2c$ returns the same data, but only distinct head/tail node label pairs.

2.8 Task 8: Path Expression Query, Type 3

- Implement a new path expression operator, which receives a path expression of the form

$$PQ3a \leftarrow NN//_{Bound}$$

where

$$NN \leftarrow (Node\_Label|Node\_Descriptor)$$

$$Bound \leftarrow (Max\_Num\_Edges|Max\_Total\_Edge\_Weight)$$

and returns the labels of the nodes in the head and tail of each path that satisfies the path expression.

$PQ3b$ returns the same data, but sorts the results in the labels of the source and tail labels.

$PQ3c$ returns the same data, but only distinct head/tail node labels pairs.

2.9 Task 9: Triangle Query

- Implement a new path expression operator, which receives a triangle expression of the form

$$TQa \leftarrow EN; EN; EN$$

where

$$EN \leftarrow (Edge\_Label|Max\_Edge\_Weight)$$

and, for each corresponding triangle, returns the labels of the corresponding three nodes.

TQb returns the same data, but sorts the results in the labels of the corresponding nodes.

TQc returns the same data, but only distinct node label triples.

3 Deliverables

You have to return the following before the deadline:

- Your source code properly **commented**, `tar`ed and `zip`ed.
- The output of your program with the provided test data.
- A report following the given report document structure. The report should detail how the various tasks are implemented.

  The report should also describe *who did what*. This will be taken very seriously! So, be honest. Be prepared to explain on demand (not only your part) but the entire set of modifications. See the report specifications.
- A confidential document (individually submitted by each group member) which rates group members' contributions out of 10 (10 best; 0 worst). Please provide a brief explanation for each group member.

4 Appendix

- All groups are required to write a report.
- Groups with 7 students are required to finish all the tasks.
- Groups with 6 students are not required to do the following tasks:
  - Task 5
  - Task 8
- Groups with 5 students are not required to do the following tasks:
  - Task 4
  - Task 5
  - Task 7

  - Task 8

- Groups with 4 students are not required to do the following tasks:
  - Task 2
  - Task 5
  - Task 8
  - Task 9