B561 Assignment 8

1. Object Relational Programming
2. For this problem you can not use arrays.

Consider the relational schema Tree(parent int, child int) represent­ing the schema for storing a rooted tree.[[1]](#footnote-2) A pair of nodes (m, n) is in Tree if m is the parent of n in Tree. Notice that a node m can be the parent of multiple children but a node n can have at most one parent node.

It should be clear that for each pair of different nodes m and n in Tree, there is a unique shortest path of nodes (n1,... ,nk) in Tree from m to n provided we interpret the edges in Tree as undirected. A good way to think about this path from a node m to a node n is to first consider the lowest common ancestor node of m of n in Tree. Then the unique path from m to n is the path that is comprised of the path up the tree from m to this common ancestor and then, from this common ancestor, the path down the tree to the node n. (Note that in this path n1 = m and nk = n.)

Define the distance from m to n to be k — 1 if (n1,..., nk) is the unique shortest path from m to n in Tree.

Write a PostgreSQL function distance(m,n) that computes the distance in Tree for any possible pair of different nodes m and n in Tree.

For example, if m is the parent of n in Tree then distance(m, n) = 1 because the shortest path from m to n is (m, n) which has length 1. If m is the grandparent of n in Tree then distance distance(m, n) = 2 since (m,p, n) is the path from m to n where p is the parent of m and p is a child of n. And if m and n have a common grandparent k then distance(m, n) = distance(m, k) + distance(k, n) = 4, etc.

1. For this problem you can use arrays.

Consider the relation schema Graph(source int, target int) repre­senting the schema for storing a directed graph G of edges.

Now let G be a directed graph that is acyclic, a graph without cycles.[[2]](#footnote-3)

A topological sort of an acyclic graph is a list of all of its nodes (n1, n2,..., nk) such that for each edge (m, n) in G, node m occurs before node n in this list.

Write a PostgreSQL program topologicalSort() that returns a topolog­ical sort of G.

1. For this problem, you can not use arrays.

Consider the following relational schemas. (You can assume that the do­main of each of the attributes in these relations is int.)

partSubpart(pid,sid,quantity)

basicPart(pid,weight)

A tuple (p, s, q) is in partSubPart if part s occurs q times as a direct subpart of part p. For example, think of a car c that has 4 wheels w and 1 radio r. Then (c,w, 4) and (c,r, 1) would be in partSubpart. Further­more, then think of a wheel w that has 5 bolts b. Then (w, b, 5) would be

in partSubpart.

A tuple (p, w) is in basicPart if basic part p has weight w. A basic part is defined as a part that does not have subparts. In other words, the pid of a basic part does not occur in the pid column of partSubpart.

(In the above example, a bolt and a radio would be basic parts, but car and wheel would not be basic parts.)

We define the aggregated weight of a part inductively as follows:

1. If p is a basic part then its aggregated weight is its weight as given in the basicPart relation
2. If p is not a basic part, then its aggregated weight is the sum of the aggregated weights of its subparts, each multiplied by the quantity with which these subparts occur in the partSubpart relation.

Example tables: The following example is based on a desk lamp with pid 1. Suppose a desk lamp consists of 4 bulbs (with pid 2) and a frame (with pid 3), and a frame consists of a post (with pid 4) and 2 switches (with pid 5). Furthermore, we will assume that the weight of a bulb is 5, that of a post is 50, and that of a switch is 3.

Then the partSubpart and basicPart relation would be as follows:

partSubPart

|  |  |  |
| --- | --- | --- |
| pid | sid | quantity |
| 1 | 2 | 4 |
| 1 | 3 | 1 |
| 3 | 4 | 1 |
| 3 | 5 | 2 |

Then the aggregated weight of a lamp is 4 x 5 + 1 x (1 x 50 + 2 x 3)= 76.

basicPart

|  |  |
| --- | --- |
| pid | weight |
| 2 | 5 |
| 4 | 50 |
| 5 | 3 |

Write a PostgreSQL function aggregatedWeight(p integer) that re­turns the aggregated weight of a part p.

1. For this problem you need to use arrays.

Consider the relation schema document(doc int, words text[]) rep­resenting a relation of pairs (d, W) where d is a unique id denoting a document and W denotes the set of words that occur in d.

Let W denote the set of all words that occur in the documents and let t be a positive integer denoting a threshold.

Let X C W. We say that X is t-frequent if

count({d|(d, W) G document and X C W}) > t

In other words, X is t-frequent if there are at least t documents that contain all the words in X.

Write a PostgreSQL program frequentSets(t int) that returns the set of all t-frequent set.

In a good solution for this problem, you should use the following rule: if X is not t-frequent then any set Y such that X C Y is not t-frequent either. In the literature, this is called the Apriori rule of the frequent itemset mining problem. This rule can be used as a pruning rule. In other words, if you have determined that a set X in not t-frequent then you no longer have to consider any of X's supersets.

To learn more about this problem you can visit the site

<https://en.wikipedia.org/wiki/ApriorWalgorithm>.

1. For this problem you can use arrays.

For this problem, first read about the k-means clustering problem in

<http://stanford.edu/~cpiech/cs221/handouts/kmeans.html>

Look at the k-means clustering algorithm described in this document. Your task is to implement this algorithm in PostgreSQL for a dataset that consists of a set of points in a 2-dimensional space.

Assume that the dataset is stored in a ternary relation with schema

Points(p int, x float, y float)

where p is an integer uniquely identifying a point (x, y).

Write a PostgreSQL program kMeans(k integer) that returns a set of k points that denote the centroids for the points in Points. Note that k is an input parameter to the kMeans function.

You will need to reason about how to determine when the algorithm ter­minates. A possible termination condition is to set a number of iterations that denotes how many iterations are run to approximate the centroids. Another termination condition is to consider when the set of centroids no longer changes.

1. Physical Database Organization and Algo­rithms
2. Consider the following parameters:

4096 bytes

block size block-address size block access time record size record key size

1. bytes
2. ms (micro seconds) 200 bytes

12 bytes

Assume that there is a B+-tree, adhering to these parameters, that indexes 108 million records on their primary key values.

Show all the intermediate computations leading to your answers.

1. Specify (in ms) the minimum time to retrieve a record with key k in the B+-tree provided that there is a record with this key.
2. Specify (in ms) the maximum time to retrieve a record with key k in the B+-tree.
3. Consider the following B+-tree of order 2 that holds records with keys 2, 7, 9, and 11. (Observe that (a) an internal node of a B+-tree of order 2 can have either 1 or 2 keys values, and 2 or 3 sub-trees, and (b) a leaf node can have either 1 or 2 key values.)

/ \  
/\

I 2 7 |--->| 9 11 |

1. Show the contents of your B+-tree after inserting records with keys 6, 10, 14, and 4, in that order.
2. Starting from your answer in question 7a, show the contents of your B+-tree after deleting records with keys 2, 14, 4, and 10, in that order.
3. Consider an extensible hashing data structure wherein (1) the initial global depth is set at 1 and (2) all directory pointers point to the same empty block which has local depth 0. So the hashing structure looks like this:

global-depth 1 local-depth 0

0 | --| >|

I

1 | --| >|

Assume that a block can hold at most two records.

1. Show the state of the hash data structure after each of the following insert sequences:3
2. records with keys 2 and 6.
3. records with keys 1 and 7.
4. records with keys 4 and 8.
5. records with keys 0 and 9.
6. Starting from the answer you obtained for Question 8a, show the state of the hash data structure after each of the following delete sequences:
7. records with keys 1 and 2.
8. records with keys 6 and 7.
9. records with keys 0 and 9.
10. Let R(A, B) and S(B, C) be two relations and consider their natural join Rn S.

Assume that R has 1,500,000 records and that S has 5, 000 records.

Furthermore, assume that 30 records of R can fit in a block and that 10 records of S can fit in a block.

Assume that you have a main-memory buffer with 101 blocks.

1. How many block IO’s are necessary to perform R M S using the block nested-loops join algorithm? Show your analysis.
2. How many block IO’s are necessary to perform R M S using the sort-merge join algorithm? Show your analysis. [[3]](#footnote-4) [[4]](#footnote-5)
3. Repeat question 9b under the following assumptions.

Assume that there are p different B-values and that these are uni­formly distributed in R and S.

Observe that to solve this problem, depending on p, it may be neces­sary to perform a block nested-loop join per occurrence of a B-value.

1. How many block IO’s are necessary to perform R txi S using the hash-join algorithm? Show your analysis.
2. Concurrency Control
3. State which of the following schedules Si, S2, and S3 over transactions Ti, T2, and T3 are conflict-serializable, and for each of the schedules that is serializable, given a serial schedule with which that schedule is conflict- equivalent.
4. Si = *Rj(x)R2(y)Ri(z)R2(x)Ri(y).*
5. S2 = Ri(x)W2(y)Ri(z)R3(z)W2(x)Ri(y).
6. S3 = Ri(z)W2(x)R2(z)R2(y)Wi(x)W3(z)Wi(y)R3(x).
7. Give 3 transactions Ti, T2, T3 and a schedule S on these transactions whose precedence graph (i.e. serialization graph) consists of the edges (Ti, T2), (T2,Ti), (Ti,T3), (T3,T2).
8. Give 3 transactions Ti, T2, and T3 that each involve read and write opera­tions and a schedule S that is conflict-equivalent with all serial schedules over Ti, T2, and T3.
9. Consider the following transactions:

T1: read(A); read(B);

if A = 0 then B := B+1; write(B).

T2: read(B); read(A);

if B = 0 then A := A+1; write(A).

Let the consistency requirement be A = 0 V B = 0, and let A = B = 0 be the initial values.

(a) Show that each serial schedule involving transaction Ti and T2 pre­serves the consistency requirement of the database.

1. Construct a schedule on T\ and T2 that produces a non-serializable schedule.
2. Is there a non-serial schedule on T\ and T2 that produces a serializable schedule. If so, give an example.

1. We assume that a tree is a connected graph with a finite number of nodes. [↑](#footnote-ref-2)
2. A cycle is a path (vo,..., Vk) where vo = Vk. [↑](#footnote-ref-3)
3. 3You should interpret the key values as bit strings of length 4. So for example, key value [↑](#footnote-ref-4)
4. is represented as the bit string 0111 and key value 2 is represented as the bit string 0010. [↑](#footnote-ref-5)