Database Systems WS 2023/24
Exercise 6: Distributed 27.11.2023, Due 04.12.2023 12:00 MEZ
Submitted by Erik Schwede, Suma Keerthi

1 Denormalization and rewriting (1 P.)

For this question, we will take another look at the uni_db schema from sheet 2:

1. How can you modify/denormalize this schema, to answer the following query efficiently? Which downsides will this have?
   Describe the normal forms of the original and changed schema in your answer. Also note which additional steps have to be taken to keep the data consistent.

   Required submission: Denormalized table schema; Explanation and downsides of denormalization; Query after denormalization; Explanation on how to keep data consistent; Normal form of original table schema; Normal form of new table schema.

   - Q1: A list of all student names and the lecture title and SWS points of all the lectures they attend.

   Solution:
   Denormalized table schema: The query for the given question is as follows:

   ```
   SELECT S.NAME, A.LID, L.SWS
   FROM STUDENTS S, ATTENDS A, LECTURES L
   WHERE S.Matrnr = A.Matrnr AND A.LID = L.LID
   ```

   Explanation and downsides of denormalization:
   We can store all the queries in one table to avoid multiple table accesses by which we can avoid joins. The data after being stored is a single table would be redundant. Query after denormalization; The attends table can be used to store the additional column as it is acting as a foreign key to both students and lectures tables. Hence, we add the columns of student name and sws to the attends table.

   ```
   SELECT A.NAME, A.LID, A.SWS
   FROM ATTENDS A
   ```

   Explanation on how to keep data consistent:
   As we are combining additional columns, we need more storage to store the data and keep it consistent. If the original table has any changes to data, the changes have to be made to this attends table columns that have been added(name and sis columns), so the data would be up-to date in both the original table as well as this attends table. Normal form of original table schema; Normal form of new table schema The normal form of the original table schema would be 3NF as it does not have transitive partial dependencies and reduces the data duplication. Adding the columns to the attends table, would repeats data making it in 1NF.

2. Rewrite the following queries to more efficient queries, returning the same result set. Consider the conditions noted for each of the queries.

   (a)
   ```
   SELECT DISTINCT *
   FROM Professors
   ```

   Output:
   ```
   SELECT * FROM PROFESSORS
   ```

   Usage of distinct can be avoided here as each professor has unique ids given and the tuples with the primary key in the table would be returned.
   Result: 7 records

   (b)
   ```
   SELECT COUNT(DISTINCT l.LID)
   FROM Prerequisites p, Lectures l
   WHERE p.Required = l.LID
   ```

   Output:
   ```
   SELECT COUNT(DISTINCT p.REQUIRED)
   FROM Prerequisites p
   ```

   Usage of join is not required here as we can also fetch the count of required column using the prerequisites table.

   Result: 4 records

   (c)
   ```
   SELECT COUNT(HeldBy)
   FROM Lectures
   GROUP BY LID
   HAVING LID = 5001
   ```

   Output:
   ```
   SELECT COUNT(HeldBy)
   FROM Lectures
   WHERE LID = 5001
   ```

   We are only searching for one record, hence we do need to apply GROUP BY and HAVING clause, instead we could directly search for the given record using the WHERE Clause.
   Result: 1 record

3. Can the view be used to execute the query? Explain in detail why the view can be used or why it cannot be used.

   Query

   ```
   SELECT MatrNr
   FROM Exam
   WHERE Grade < 3
   ```

   View

   ```
   SELECT MatrNr, PID
   FROM Exam
   WHERE Grade = 1
   ```

   Solution:
   The view only stores the result of the MatrNr and PID from the Exams table for Grade = 1, but the query which we are running is asking all the matriculation numbers from the exam tables less than 3. Hence the view cannot be used to execute the query.

   Output of query: 3 records
   Output of view: 1 record

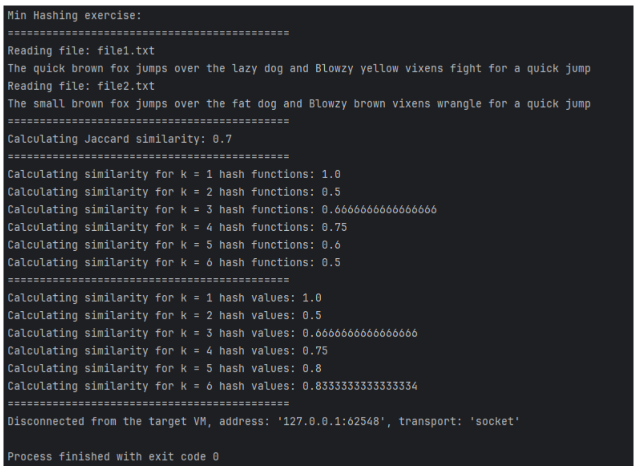
2 Min-Hashing (1 P.)

Implement Min-Hashing in a language of your choice. Your program has to take two text files as input and parse them into sets of words. The program has to calculate and display:

- Jaccard coefficient between the two texts.
- The similarity estimated by Min-Hashing with k different hash functions (for each $k \in [1, 6]$)
- The similarity estimated by Min-Hashing with k different min values of one hash function (for each $k \in [1, 6]$)

You can use the template in OLAT, which already parses the files and provides 6 hash functions. Submit the code and the output of your program when executed with the two data files provided in OLAT. If you do not use the template, also submit instructions on how to compile and execute your program. Do not submit the data files.

Solution:



```
Min Hashing exercise:
============================================
Reading file: file1.txt
The quick brown fox jumps over the lazy dog and Blowzy yellow vixens fight for a quick jump
Reading file: file2.txt
The small brown fox jumps over the fat dog and Blowzy brown vixens wrangle for a quick jump
============================================
Calculating Jaccard similarity: 0.7
============================================
Calculating similarity for k = 1 hash functions: 1.0
Calculating similarity for k = 2 hash functions: 0.5
Calculating similarity for k = 3 hash functions: 0.6666666666666666
Calculating similarity for k = 4 hash functions: 0.75
Calculating similarity for k = 5 hash functions: 0.6
Calculating similarity for k = 6 hash functions: 0.5
============================================
Calculating similarity for k = 1 hash values: 1.0
Calculating similarity for k = 2 hash values: 0.5
Calculating similarity for k = 3 hash values: 0.6666666666666666
Calculating similarity for k = 4 hash values: 0.75
Calculating similarity for k = 5 hash values: 0.8
Calculating similarity for k = 6 hash values: 0.8333333333333334
============================================
Disconnected from the target VM, address: '127.0.0.1:62548', transport: 'socket'

Process finished with exit code 0
```

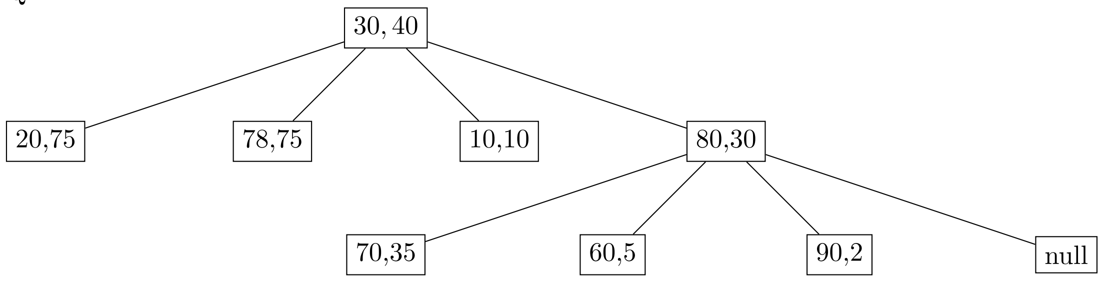
3 Quadtrees (1 P.)

1. Give 8 points, with coordinates where each dimension has values in the range $[0, 100]$, and sort them into two insertion orders. The first should have less empty leaf nodes if inserted into a quadtree than if they were inserted into a PR quadtree. The second one the other way around.

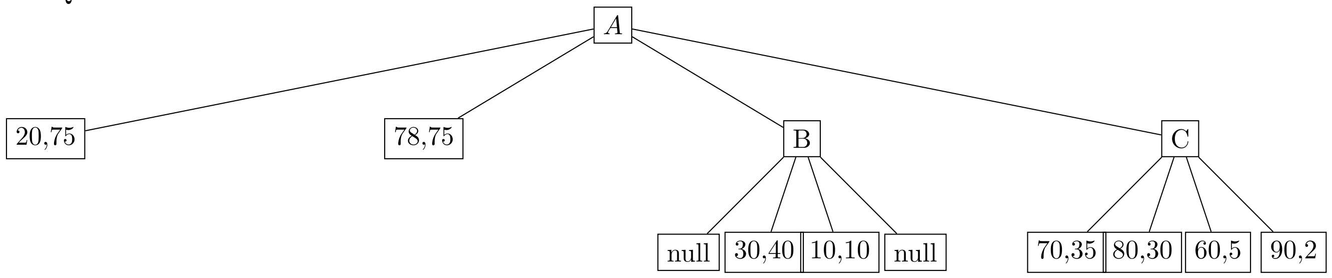
   Submit the points, the two insertion orders and draw the (PR) quadtrees. You can either draw them as tree or as the grid visualization.

   Solution:
   The quadtree has less empty leave nodes than the pr-quadtree if the 8 points get inserted in the following order: $A(30, 40); B(20, 75); C(78, 75); D(10, 10); E(80, 30); F(60, 5); G(90, 2); H(70, 35)$

   Quadtree:



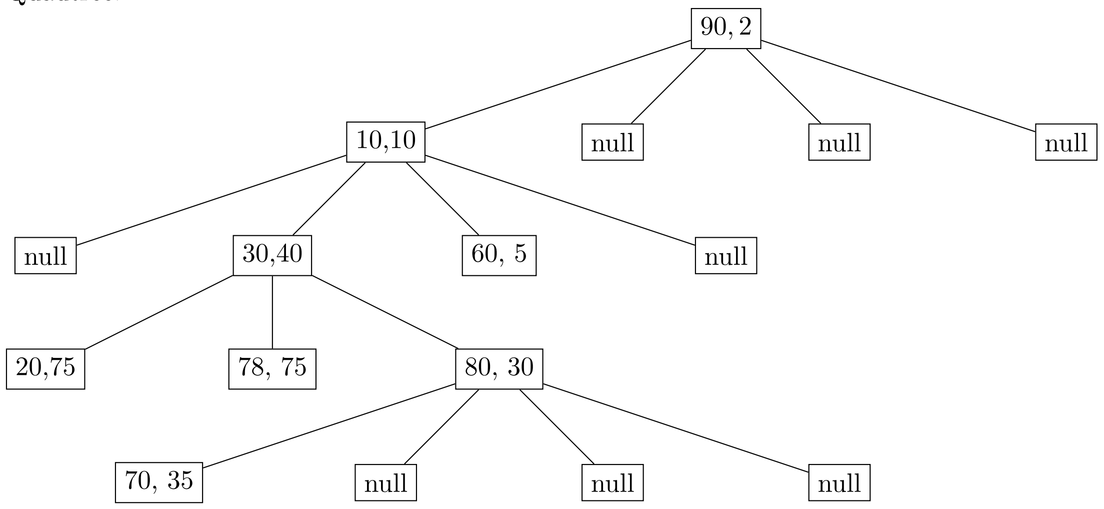
   PR-Quadtree:



   Changed order:

   $G(90, 2); D(10, 10); A(30, 40); F(60, 5); B(20, 75); C(78, 75); E(80, 30); H(70, 35)$

   Quadtree:



PR-Quadtree: The PR-Quadtree is not relying on the order of the tree items. No mater what order is used the tree will be always the same.

2. Assuming we have the whole dataset present upon insertion, describe one possible way that optimizes the creation of the quadtree, such that the height of the resulting quadtree is minimized. Note that before the bulk-loading, preprocessing of the data is allowed.

   Solution: The data point that has the is the closesd to the average of the y koordinate and the x koordinate should be choosen to inserted first. From than on this gets repreated for each sub quadrant. until all the data is put into the tree structure.

3. Assuming we are given uniformly distributed point data, what is the probability that a node at depth k contains a particular point in a PR Quad tree? Additionally, for a collection of v points, calculate the probability that none of the points lies in a given cell at depth k?

   Solution: The probability is $1/4^k$ for a particular point.
   The probability that no point is inside a specific cell at the depth k is $(1 - 1/4^k)$

4 kd tree (1 P.)

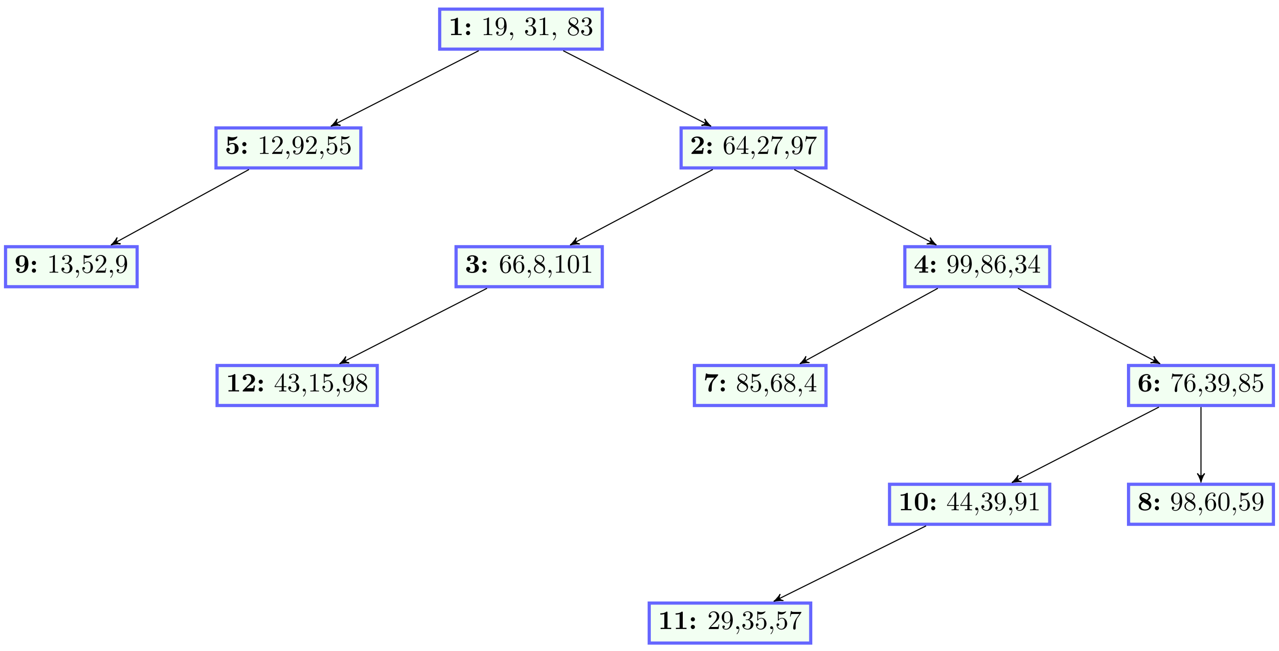
1. Insert the following values, in provided order, into an empty *k*d tree, with $k = 3$:

   1. (19, 31, 83)
   2. (64, 27, 97)
   3. (66, 8, 101)
   4. (99, 86, 34)
   5. (12, 92, 55)
   6. (76, 39, 85)
   7. (85, 68, 4)
   8. (98, 60, 59)
   9. (13, 52, 9)
   10. (44, 39, 91)
   11. (29, 35, 57)
   12. (43, 15, 98)

   Draw the result as a tree, like shown in the lecture. Note which values you used for the split.

   Solution:

   KD Tree implementation



Data Points	Split Value	Split Position
(19, 31, 83)	19	0
(64, 27, 97)	64	0
(66, 8, 101)	8	1
(99, 86, 34)	86	1
(12, 92, 55)	12	0
(76, 39, 85)	85	2
(85, 68, 4)	4	2
(98, 60, 59)	98	0
(13, 52, 9)	52	1
(44, 39, 91)	44	0
(29, 35, 57)	35	1
(43, 15, 98)	98	2

In the above table, we are considering the given data points/ co-ordinates as (X,Y,Z) and (0,1,2) to represent the split positions for the respective X,Y,Z values

2. Adaptive kd tree

   Provide **in detail** an algorithm, in pseudo code, that takes a list of coordinates in k dimensions and builds a kd tree. This tree may have a maximum of ten data points in its leaves. The dimension used for the split should be the one with the largest variance. The value used for the split, should be the average of the split dimension of all relevant coordinates.

   Solution: The Adaptive kd tree is the variant of KD Tree, where the partitioning is done not only on the basis of alternating dimensions, but also depends on data distribution. Not sure of the algorithm