Advanced Data Engineering: Assignment 6

NGUYEN T. Hoang - SID: 15M54097
(ホアン)

Fall 2015, W831 Tue. Period 5-6
Due date: 2015/12/01

Problem

- Write a pseudo code for hash join with hash collision handling.

Answer

The hash join executes in two phases: Build and probe. During the build phase, it reads all rows from the first relation (BuildTable), hashes the rows on the equijoin keys, and create an in-memory hash table. During the probe phase, the algorithm goes through the second relation (ProbeTable) to calculate hash and compare with the hash table. There are two problems regarding hash collision in the hash join scheme:

1. Two different key values from different tables that hash to the same value.
2. Two different key values from same table that hash to the same value.

To solve the first problem, my algorithm checks for the true key values equality whenever we have a hash match in the probe phase. For the second problem, I use linked-list method to solve hash collision. The hash table will store a linked list that contains the tuples instead of the data tuple. Because my solution uses a dynamic linked list, therefore it has memory efficient. However, in the worst case scenario, my hash algorithm will have performance as normal search join operation, but this case will not likely to happen if we have a properly good hashing funchion $hash(value)$ and adequate memory space. The pseudo-code for two phases are presented in the next page.

Listing 1 Hash Join Algorithm - Build Phase

```
for each row R1 in BuildTable:
begin
    // Calculate hash value
    hash_key = hash( R1[keys_indices] )
    // Check for collision
    if hash_table[hash_key] is empty :
    begin
        linked_list.add(R1)
        hash_table[hash_key] = linked_list
    end
    else :
    begin
        // Search for the value in the linked list
        if hash_table[hash_key] does not contains R1 :
        begin
            hash_table[hash_key].add(R1)
        end
    end
end
```

Listing 2 Hash Join Algorithm - Probe Phase

```
for each row R2 in the ProbeTable:
begin
    // Calculate hash value
    hash_key = {\color{RoyalBlue}hash}( R2[keys_indices] )
    // Search for the same value
    for R1 in list hash_table[hash_key] :
    begin
        if R2 joins R1 :
        begin
            return (R1, R2)
        end
    end
end
```