Xiaowei Liang, Lingting Ge

CSE 232 Database Systems: Advanced Topics and Implementation

March 14, 2017

Milestone 3

In this project, we are asked to 1. implement a query rewriter which transforms a query following specific syntax to one using join explicitly; 2. realize join function based on the idea of hash join. The report will describe our approach on solving these two tasks.

1. Query Rewriter

Instead of processing on the string of query, we use antlr4 to parse the query for convenience. We utilize the idea of relational database to represent different nodes. For a node like $a in doc(…)/XXX, we treat it as a relation (or a table). For a node like $b in $a/XXX, we regard it as an attribute. Notice different from normal relational models, a relation itself is also its attribute. Once we read an assignment clause, we first decide whether it's a relation or an attribute. If it is an attribute, we also need to find which relation it belongs to. We use a map var2var<Node,Node> to store the attribute-relation mapping.

If we find a join clause, we will not rewrite the query as there will be no partial join according to the guide.

When we start to read where clause, for each $v1 eq $v2, there are two possible cases. Case 1, $v2 is a constant or $v1 and $v2 are attributes of the same relation r1. In this case, we just select the tuples from r1 satisfying the conditions and no join will be applied.

Case 2, $v1 and $v2 are attributes of different relation r1 and r2. For attribute class, we introduce another member variable eqTo which is a list of attributes to store the attributes it equals to.

Once we complete reading assignment clause and where clause, we are able to apply the join. We use a naive approach to join all relations. Pick one relation r1, and find a relation r2 where r2 has an attribute a_21 equal to one of relation r1's attributes a_11. Then we start to join r1 and r2: find a set of attributes s1 of r1 and s2 of r2 where attribute in s1 equals to that in s2. Then the output is formatted as: join(att1 in def(att1), for att2 in def(att2), [s1], [s2]) for each attribute att1 of relation r1 and att2 of relation r2. def(att) returns the definition of an attribute which is stored when reading assignment clause.

To rewrite return clause, we need to rewrite every relation $r and attribute $a as $r/* and $a/*.

2. Hash join

For simplicity, we use the string representation of a node as the key value of the map. For example, we join relation r1 and relation r2 and joined attributes of r1 and r2 are s1 and s2. s1=[a_1, a_2,…,a_n], s2 = [b_1, b_2, …, b_n]. We use attribute b_1 of each tuple of r2 as the index. Then we scan over each tuple t1 in r1, search the hash map using attribute a_1 of t1 and get the node list lst. For every node t2 in lst, we compare value of attribute b_i and value a_i of t. If for i=2…n, the values are the same, we join two tuples and create a new node containing children of both nodes and add it to the result.