Deliverable 1 2ID35
Database Technology

C. Lambrechts - 0733885 - c.lambrechts@student.tue.nl
K. Triantos - ?? - k.triantos@student.tue.nl
J. Wulms - 0747580 - j.j.h.m.wulms@student.tue.nl

May 7, 2014

Abstract

This report contains the first deliverable of the project for the course 2ID35 Database Technology. We provide some background for the paper we are studying, work out the research problems that have been addressed and what results have been claimed by the paper. We then proceed by giving an overview of what we are going to do in order to verify the research that has been done by the authors of the paper, and a discussion of the results so far.

1 Introduction

Optimizing a query can be done in several ways. The operation that is the computational most expensive is the join operation. So it makes sense to start by making this operation faster. Changing the order in which joins are performed is a common approach. For this approach cost approximations are computed before actually performing the joins. This approach requires the development of a cost model, an assignment of an estimated cost to each query processing plan and searching in the (huge) search space for the cheapest cost plan. There are also approaches that do not use a cost plans. One of the approaches focusses on minimizing the number of joins instead of using a optimal join order. This approach requires a homomorphism test, which is NP-complete. An other approach focusses on the structural properties of the queries. It will try to find a project-join order that will minimize the size of the intermediate results during query evaluation. Practical it means, use only the data that you need and throw away the rest as soon as possible.

The paper we are reviewing is such a structural approach. The authors of that paper try to push down the projections, such that the attributes that are not needed are projected out as early as possible. The projection pushing strategy has been applied to solve constraint satisfaction problems in Artificial Intelligence with good experimental results. The input to a constraint-satisfaction problem consists of a set of variables, a set of possible values for the variables, and a set of constraints between the variables; the question is to determine whether there is an assignment of values to the variables that satisfies the given constraints. IN the database area, this means that you are searching for entries that satisfy the constraints on their attributes.

Optimizing the query manually, often focusses on reducing the search space before the join operation. In this fashion the number of entries used in the join is less, which can result in a huge performance gain. Most of the time the projections are pushed down by selecting

as soon as possible or pushing the projection to a sub query. It is possible to make a sub-query that projects out all irrelevant information and tries to reduce the intermediate results. Optimizing the query in an automated and structural way looks promising and practical.

2 Problem Description

3 Claimed results

In order to verify the paper we are looking into, we need a precise overview of the results that are claimed by the authors of the paper. This way we are able to make clear comparisons between the claimed results and our own.

3.1 Paper results

We first look at the results for the 3-COLOR graphs where order is fixed and the density scales:

- The curves for Boolean and non-Boolean queries have roughly the same shape.
- At first the running time increases as density increases, becaue of an increased number of joins.
- Eventually the size of intermediate results becomes small or empty, and additional joins have little effect on overall running time.
- At low density each optimization method improves upon the previous. For denser instances, optimizations using early projection lose their effectiveness.
- Bucket elimination completely dominates the greedy methods.

Proceeding with the results for 3-COLOR graphs where density is fixed and the order scales. The density is fixed at 2 values, and the authors assume that the lower value is most likely associated with 3-colorable graphs, and the higher density with non-3-colorable graphs:

- All methods show exponential increase in running, when order is increased. (This is shown by a linear slope in logscale.)
- Bucket elimination maintains a lower slope in logscale, in comparison to the other optimazations. This lower slope in logscale translates to a strictly smaller exponent, so we have an exponential improvement.

The next focus of the report was order-scaling experiments with structured queries. We first look into augmented path instances:

- Bucket elimination is again the best, but early projection is competitive for these instances, because the problem has a natural order that works well for early projection.
- For non-Boolean graphs the optimizations do not scale as well as for Boolean graphs. This is due to the fact that there are $\geq 20\%$ less vertices to exploit in the optimization. Early projection and bucket elimination still dominate the other optimizations in this case.

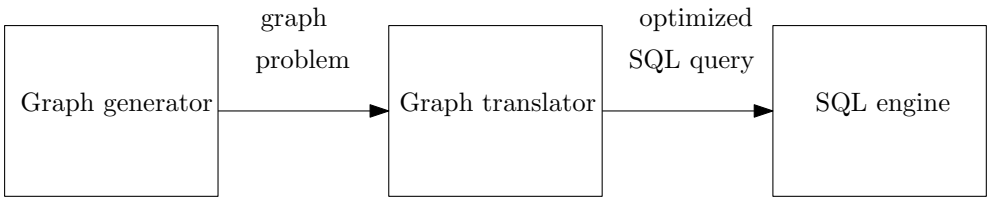

Figure 1: All steps in the process of verification

The final claims are about the results for ladder graph instances, and augmented ladder instances:

- For ladder instances, the heuristic for reordering is not only unable to find a better order, but actually finds a worse one.
- Furthermore, ladder instances give results very similar to augmented path instances.
- Augmented ladder instances shows even more differences between optimization methods.
- Non-Boolean cases for augmented ladder instances struggle to reach order 20 with the faster optimizations.

The conclusion for all these results is that bucket elimination dominates the field at every turn with an exponential improvement. In a discussion about future research areas, the authors also claim that they found results consistent with 3-COLOR queries, when using 3-SAT and 2-SAT to construct queries.

3.2 Verification

In our verification, the main claims we want to verify are the domination of bucket elimination and the exponential improvement it shows in the paper. The next step is going into the details of all the different query types (random and structured), and getting consistent results there, or finding out why we have different results. When everything works out as planned, we can further look into queries constructed from other sources than 3-COLOR, such as 3-SAT or 2-SAT.

4 Methodology

This section will elaborate on the steps we are going to take to verify the results in the paper.

5 Discussion of progress