Poster Paper

Empirical evaluation of guarded structural indexing

Erik Agterdenbos
George H. L. Fletcher
Eindhoven University of Technology
agterdenbos@gmail.com,
g.h.l.fletcher@tue.nl

Chee-Yong Chan
National University of Singapore
chancy@comp.nus.edu.sg

Stijn Vansummeren
Université Libre de Bruxelles
svsummer@ulb.ac.be

ABSTRACT

Traditional indices in relational databases are designed for queries that are *selective by value*. However, queries can also retrieve records on their *relational structure*. In our research, we found that traditional indices are ineffective for structurally selective queries. To accelerate such queries, so-called 'structural indices' have been applied in graph databases. These indices group together structurally similar nodes to obtain a compact representation of the graph structure.

We studied how structural indices can be applied in relational databases and evaluated their performance. Guarded bisimulation groups together relational tuples with similar structure, which we use to obtain a guarded structural index. Our solution requires significantly less space than traditional indices. At the same time, it can offer several orders of magnitude faster query evaluation performance.

1. PROBLEM DEFINITION

Queries that are selective by value can be accelerated by using B-trees or Hash indices on attributes in the selection condition. The problem is that such indices cannot efficiently answer structurally selective queries. Consider a relation $customer(\underline{id}, name, address, phone)$ and a relation $order(\underline{id}, status, total\_price, customer\_id)$, where $\underline{id}$ denotes the primary key (PK) in both relations. Further, we have a foreign key (FK) from $order(customer\_id)$ to $customer(id)$.

For example, we might want to retrieve all names of customers that do or do not have an order. To answer these queries, a semijoin or antijoin has to be processed. Semijoins and antijoins require table scans, index scans or index only scans on both relations. These operations are relatively expensive. Moreover, these are *tuple selecting queries*. The result is a subset of a *single* relation: the *customer* relation. No information from the *order* relation occurs in the output. However, the *order* relation or its index must be scanned to determine which tuples must be returned, which impacts the performance when the *order* relation is much larger than the the *customer* relation.

Value-based indices are not directly useful in accelerating structurally-sensitive selections such as these.

2. BACKGROUND

The main concept of structural indices is to build a summary that is smaller than the original database, while preserving relevant structural properties. Bisimilarity and similarity are two formal notions of structural similarity in graphs which have been used for summarizing semi-structured and graph databases for structural indexing [3, 7]. Andréka et al. [2] and Otto [5] introduced guarded bisimulation and guarded simulation which extends these notions to relational databases. Picalausa et al. characterized query invariance under guarded (bi)simulation (i.e., for which query languages guarded (bi)simulation is the correct structural notion for indexing w.r.t. queries in the language), providing a formal basis for the engineering of guarded structural indices [6]. This abstract summarizes the main results of our empirical study of the practical feasibility of this novel approach to indexing; details can be found in [1].

3. APPROACH

Our approach to build a structural index can be summarized as follows: first we represent the relational database instance as a graph. Second, we apply an external memory bisimulation partitioning algorithm to group similar nodes. Third, we map the partitioning of the original tuples. We summarize each step in the following paragraphs.

Relational databases allow joins on any set of attribute pairs with equal types. Because FK constraints are popular candidates for join conditions, we only consider PK-FK joins. The graph is constructed as follows: we create a node for each PK value in the database. Then we create forward edges from PK to FK values and backward edges from FK to PK values. We use relation names as node labels and FK constraint names for edge labels.

We apply a localized version of bisimulation equivalence on the graph, namely, k-bisimulation. This equivalence relation induces a partitioning of nodes with respect to topological features of their k-neighborhoods. The k-neighborhood of a node n is the subgraph consisting of all nodes at most k edges away from n. The partitioning result consists of a mapping from nodes (tuples) to partition blocks, which represent k-bisimulation equivalence classes, and a reduced graph that summarizes the original structure. The mapping is stored by tagging each tuple in each relation (in an additional attribute) with the distinct identifier of the partition block to which the tuple belongs. The reduced

Table 1: Queries for TPC-H dataset

Query	description
FACQ 1	Select partsupp supplycosts having lineitem
FACQ 2	Select part names having a lineitem
FACQ 4	Select region having nation, customer, orders and lineitem
GF 1A	Select partsupp comment without lineitem
GF 1B	Select customers not having an order
GF 4	Select all suppliers that sell parts that are offered but not sold by other suppliers

graph is stored in an additional relation *edge_label*(*from_id*, *to_id*). After construction of *edge_label*, the PK-FK graph is no longer needed and is discarded.

k-bisimulation structural indices allow the acceleration of semijoins and antijoins with join trees of height $h \leq k$. The reduced graph is used to determine which equivalence classes are selected. Then, the projected relation is scanned and tuples that are tagged with those equivalence classes are returned. Queries with join trees of height $h > k$ can be partially accelerated via query decomposition.

4. EXPERIMENTAL STUDY

Set up. We used the DBLP[1] data set and the TPC-H[2] data set with scale factor 1 to evaluate guarded structural indexing. PostgreSQL 9.3 and the external memory bisimulation partitioning solution of Luo et al. [4] were used. Table 1 lists the queries used in our evaluation.

Results. Figure 1 shows the number of partition blocks that are generated under k-bisimulation. A higher value of k leads to more equivalence classes, uses more disk space, and can accelerate higher join (sub)trees. Figure 2 shows the reduced graph under 2-bisimulation. We observe significant compression while preserving non-trivial structure.

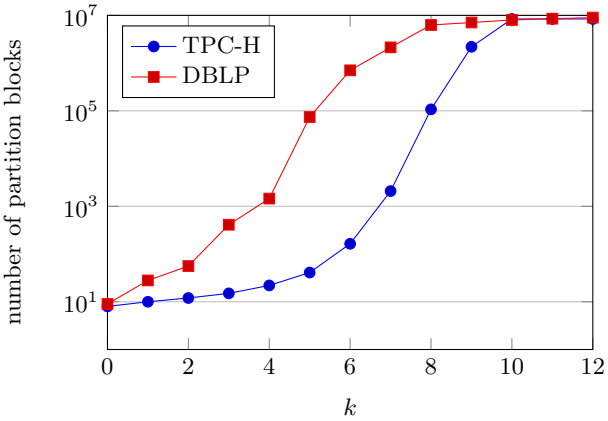

Figure 1: Partition size under k-bisimulation

Figure 3 shows the speed-up of structural indexing over value-based B-trees on foreign keys. For $4 \leq k \leq 7$, this is between 4 and 190 times faster. We also observed that our index uses only 1 megabyte of disk space for these k values compared to the more than 450 megabytes required for value-based indexes.

[1]http://dblp.uni-trier.de/xml/
[2]http://www.tpc.org/tpch/

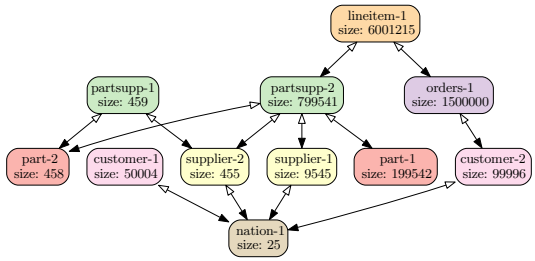

Figure 2: TPC-H partitioning under 2-bisimulation

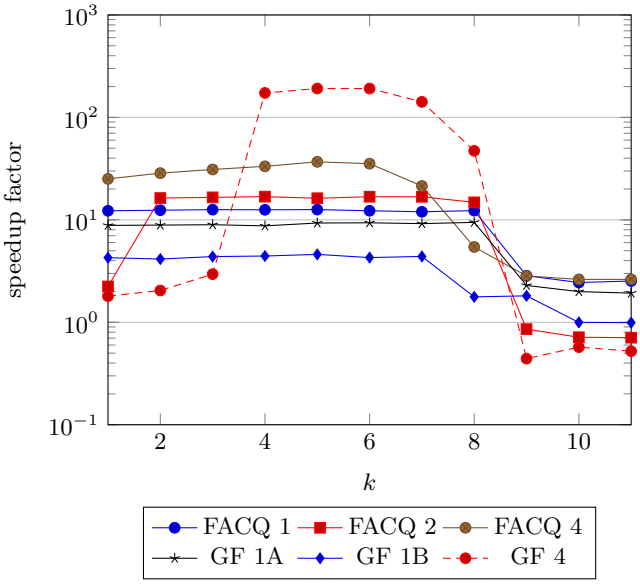

Figure 3: Query running time speedup factor

Conclusions. Our results show that guarded structural indices can be orders of magnitude smaller and faster than traditional indexes. This indicates the significant promise of further study of this new approach to indexing.

5. REFERENCES

[1] E. Agterdenbos. Structural indexing for accelerated join-processing in relational databases. Master's thesis, Eindhoven Universtity of Technology, 2015.

[2] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *J. Phil. Logic*, 27(3):217–274, 1998.

[3] Y. Luo et al. Storing and indexing massive RDF datasets. In R. De Virgilio et al, editor, *Semantic Search over the Web*, pages 31–60. Springer, 2012.

[4] Y. Luo et al. External memory k-bisimulation reduction of big graphs. In *CIKM*, pages 919–928, 2013.

[5] M. Otto. Highly acyclic groups, hypergraph covers, and the guarded fragment. *J. ACM*, 59(1), 2012.

[6] F. Picalausa et al. Principles of guarded structural indexing. In *ICDT*, pages 245–256, 2014.

[7] D. Sangiorgi and J. Rutten. *Advanced topics in bisimulation and coinduction*. C. U. Press, 2011.