Squeezing the Most Out of Relational Database Systems

Jonathan Goldstein
Microsoft Research

Raghu Ramakrishnan
University of Wisconsin - Madison

With the increasing speed of CPUs relative to disks, using compression as a means of improving disk information throughput is becoming very attractive. Traditional compression algorithms such as Lempel-Ziv, which is the basis of the standard gzip compression package, are inadequate for compressing relations in a relational database system. This inadequacy is derived from two problems. The first is that conventional compression algorithms like Lempel-Ziv require uncompressing a large portion of the file even if only a small part of that file is required. This combined with the unpredictability of compression ratios introduces significant complexity in the buffer manager.

In response, we developed (in previous work) a compression technique called frame of reference (i.e. FOR) compression, that can be used to compress fixed length fields in relations. While previous work presented the basic technique and demonstrated its application to indexes, the effect on relational databases in general was not studied. This is the topic of the current work, which makes the following contributions:

- We present compelling experimental evidence of the suitability of FOR compression for many database applications.
- We show that basic streaming operator speedups are commensurate with compression ratios, providing a convenient way to estimate speedups. The exception to this was nested loops join, where improved buffer utilization led to better than expected speedups.
- We show that data warehouses are ideal candidates for FOR compression. Their workloads, which are read mostly with infrequent batch updates, are perfectly suited to FOR compression. In addition, the central fact table of a star schema (the schema design usually employed in data warehouses) consists of many, typically low cardinality, fixed length fields. Consequently, the expected compression ratios are very high. In addition, these central fact tables are generally the bulk of the database. As a result, scans and sorts of these tables are the dominant contributors to query costs. We study the effect of compression on TPCD data and queries and show that speedup factors of about three can be obtained.
- We apply FOR compression to a real dataset from a major catalog company and achieve compression ratios of nearly a factor of three. Scans and sorts on this data were again sped up by a factor of two to three.
- We discuss the addition of FOR compression to an existing database and conclude that it is straightforward, and affects relatively few parts of the system.
- We demonstrate that strategies which rely on efficient tuple ID based access (e.g., secondary indexes) are not adversely affected by compression. This can be traced to the selective decompressibility of individual fields of individual tuples.
- While all experiments in this paper used fixed length tuples for ease of implementation, we observe that it is not hard to do fixed length field compression in the presence of variable length fields. In fact, the offset and length of variable length fields are fixed length themselves and can be compressed!
- We discuss strategies for handling updates over compressed data that can actually result in improved update performance.

In summary, while there has been some previous and concurrent work on compressing relations, no alternative solution combines the high compression ratios, low overhead, ease of incorporation into an RDBMS, and selective decompression that FOR compression achieves. Overall, we believe that this is a very attractive approach to implementing compression in an RDBMS.