Cracking Big Data

by Stratos Idreos

A fundamental and emerging need with big amounts of data is data exploration: when we are searching for interesting patterns we often do not have a priori knowledge of exactly what we are looking for. Database cracking enables such data exploration features by bringing, for the first time, incremental and adaptive indexing abilities to modern database systems.

Source: Shutterstock

Good performance in state of the art database systems relies largely on proper tuning and physical design. Typically, all tuning choices happen up front, assuming sufficient workload knowledge and idle time. Workload knowledge is necessary in order to determine the appropriate tuning actions, ie to decide which proper indexes should be created, while idle time is required in order to actually perform those actions. In other words, we need to know what kind of queries we are going to ask and we need to have enough time to prepare the system for those queries.

However, in dynamic environments with big data, workload knowledge and idle time are scarce resources. For example, in scientific databases, new data arrive on a daily or even hourly basis, while query patterns follow an exploratory path as the scientists try to interpret the data and understand the patterns observed; there is no time and knowledge to analyze and prepare a different physical design every hour or even every day; even a single index may take several hours to create.

Traditional indexing presents three fundamental weaknesses in such cases: (a) the workload may have changed by the time we finish tuning; (b) there may be no time to finish tuning properly; and (c) there is no indexing support during tuning.

In this project, we propose a new approach to the physical design problem, called database cracking. Cracking introduces the notion of continuous, incremental, partial and on demand adaptive indexing. Thereby, indexes are incrementally built and refined during query processing. The net effect is that there is no need for any upfront tuning steps. In turn, there is no need for any workload knowledge and idle time to set up the database system. Instead, the system autonomously builds indexes during query processing, adjusting fully to the needs of the users. For example, as a scientist starts exploring a big data set, query after query, the system follows the exploration path of the scientist, incrementally building and refining indexes only for the data areas that seem interesting for the exploration path. After a few queries, performance adaptively improves to the level of a fully tuned system.

From a technical point of view cracking relies on continuously physically reorganizing data as the users pose more and more queries. Every query is used as a hint on how data should be stored. For example a data column referenced in queries is continuously reorganized (partitioned) as part of processing those queries. The actual query selection bounds are used for partitioning. This brings structure and partitioning information, allowing future queries to access data faster. Future queries exploit the partitioning information gained from the previous queries but they also introduce even more partitioning, bringing the continuous adaptation property. From a performance point of view cracking imposes a minimal overhead compared to the default performance of using no indexes, while at the same time it continuously converges at the optimal performance of using full indexes even though it requires zero initialization cost.

Cracking was proposed in the context of modern column-stores and has been hitherto applied for boosting the select operator performance, joins, maintenance under updates, and arbitrary multi-attribute queries. In addition, more recently these ideas have been extended to exploit a partition/merge-like logic as well as workload robustness is achieved via stochastic cracking. Future and ongoing research aims to tackle concurrency control, disk based cracking, maintenance under long query sequences and holistic indexing.

This project takes place primarily at the Database Architectures group of CWI in Amsterdam, The Netherlands and is part of the MonetDB column-oriented database system research projects. Other labs involved include HP Labs, Palo Alto, USA, National University of Singapore (NUS) and Rutgers University, USA. Several researchers are involved: Stratos Idreos (CWI), Eleni Petraki (CWI), Stefan Manegold (CWI), Martin Kersten (CWI), Goetz Graefe (HP Labs), Harumi Kuno (HP Labs), Panagiotis Karras (Rutgers U.), Felix Halim (NUS), and Roland C Yap (NUS).

CWI and Stratos Idreos won several awards for this research, including the 2011 ERCIM Cor Baayen Award and the 2011 ACM SIGMOD Jim Gray Dissertation award.

Links:
http://homepages.cwi.nl/~idreos/
http://www.monetdb.org

Please contact:
Stratos Idreos
Database Architectures group
CWI, The Netherlands
Tel: +31 20 592 4169
E-mail: s.idreos@cwi.nl