Mrs. Aarti Chugh al. al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 2, Issue 2, June 2015, pp. 42-44

Review of Database Cracking Techniques

Mrs. Aarti Chugh

Assistant professors, Department of Computer Science, Amity University, Haryana.

Abstract- **As there is a huge demand of dynamic and efficient data storage environments, numerous physical data reorganization techniques are designed and tested every day. This paper focuses on database cracking which aims to build the truly self-organizing database system that will continuously and automatically adapt to workload changes. Cracking completely removes the need for human administration [1]. Different techniques are discussed which give an insight into this area. A brief overview of stochastic database cracking is given which is the most efficient way for managing changing database workload environments.**

Keywords— Database cracking, Self-Organizing, Physical organization, Workload, Cracking

I. Introduction

Database cracking is a new query processing paradigm and adaptation paradigm towards truly self-tuned systems. [12] Cracking requires zero human input, no a priori workload knowledge and no idle time to prepare. The ultimate goal of database cracking is to build the first truly self-organizing database system that will continuously and automatically adapt to workload changes (random). Cracking completely removes the need for human administration. Though cracking is not an auto-tuning tool, i.e., it is not an external piece of software/hardware to help with system administration. Instead cracking represents a new internal kernel design by introducing new ways of storing and accessing data. This way, the very way data is stored and subsequently accessed by queries is continuously changing to adapt to the workload and to converge to the ultimate performance.[2]

Database cracking sets a new query processing and adaptation paradigm. It follows both automatic index selection and partial indexes for future queries, it refined until sequential searching a partition is faster than binary searching into the AVL tree guiding a search to apply partition.[4]

II. Related Work

In original database cracking, cracking treats each query as a hint on how to reorganize data in a blinkered manner; it takes each query as a literal instruction on what data to index, without looking at the bigger picture. [12] It is thanks to this literalness that cracking can instantly adapt to a random workload; yet, this literal character can also be a liability. With a non-ideal workload, strictly adhering to the queries and reorganizing the array so as to collect the query result, and only that, in a contiguous area, amounts to an inefficient quick sort-like operation; small successive portions of the array are clustered, one after the other, while leaving the rest of the array unaffected.

DB cracking is being used in Monet DB system [6][8] at the "Centrum Wiskunde & Informatica (CWI) in Amsterdam database architectures research group since 1993. Monet DB is an open-source column-store DBMS with multiple innovations in its core design. Till now many releases of Monet DB has been introduced in market i.e. Monet DB 2009, Monet DB/X Query [3][5][7].

It becomes a revolutionary DB as it stores each attribute column wise instead of row wise in traditional databases and then cracks it for the benefits of future query in terms of fast response time and throughput.

Another technique named C-Store (Stonebraker et al., 2005), is a column oriented database system and its main architecture novelty is that each column/attribute is sorted and this order is propagated to the rest of the columns [1]. Multiple projections of the same relation can be maintained, up to one for each attribute to prepare for the workload. It has shown best compression capabilities.

Approaches such as soft indexes [1] try to exploit the scan of relevant data (e.g., by a select operator) and send this data to a full-index creation routine at the same time. This way, data to be indexed is read only once. Still, the problem remains that creating full indexes significantly penalizes individual queries. Hence, various approaches are designed to enhance self-organizing database system.

Several auto tuning tools are also designed to prepare database to handle fluctuating workload. ((Chaudhuri and Narasayya, 1997). These tools help DBA by continuous monitoring and analyzing the various alternatives in the system. They rely on the what-if analysis paradigm and close interaction with the system's query optimizer [6, 8].

III. Methodology

Database cracking uses THREE PIECE CRACK and TWO PIECE CRACK algorithm on "copy of a column" for very first query and rest Queries respectively; maintaining the benefits of (a) Original column remains intact (b) No overhead of maintaining complete table.

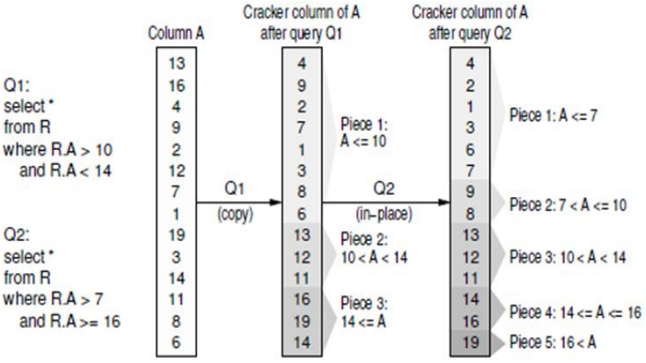

Figure 1 showing basic Cracking method for range queries [1] [3].

As figure shows that on copied column three piece crack algorithm is used resulting in three pieces ranging A<=10,10<A<14,14<=A and next range predicate. Two piece crack algorithm has been applied resulting in 5 more pieces.
This type of data cracking is known as Selection Cracking. [1] The main innovation is that the physical data store is continuously changing with each incoming query q, using q as a hint on how data should be stored. All crack actions happen as part of the query operators, requiring no external administration. Figure 1 shown above is an example of two queries cracking a column using their selection predicates as the partitioning bounds. Query Q1 cuts the column in three pieces and then Q2 enhances this partitioning more by cutting the first and the last piece even further, i.e., where its low and high bound fall. Each query has collected its qualifying tuples in a contiguous area.
This first ever made purely column oriented database gives an extra ordinary performance when compared to traditional FULL SORTING and SCAN in dynamic environment along with benefits such as

- Performance in Random workload [3]
- Self-organizing Tuple Reconstruction in Column-stores [6]
- Self tuning without DBA[3]
- Histogram for free[7]
- No need for idle time and prior knowledge regarding workload [3][9-10]
- Deals with only required column/tables/range/queries [5][9]

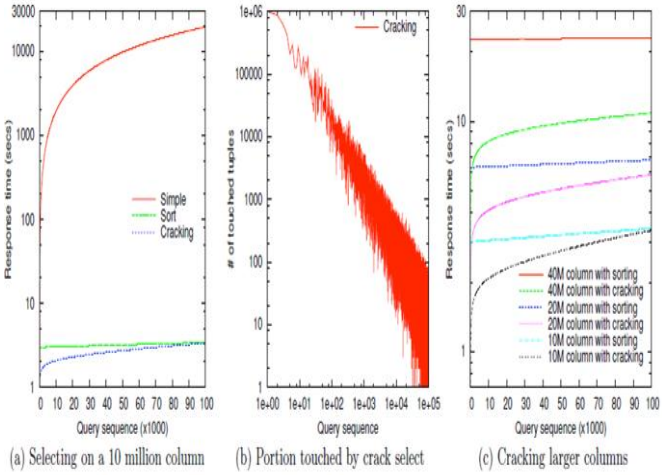

Figure 2: shows DB cracking performance against SCAN and SORT for response time per query. [3].

IV. Stochastic Cracking

Stochastic cracking ventured to drop the strict requirement in original cracking that each individual query be literally interpreted as a re-organization suggestion. It forced reorganization actions that are not strictly driven by what a query requests, but are still beneficial for the workload at large. Therefore partially driven action "by what queries want" "partially arbitrary in character".
Stochastic database cracking is a significantly more resilient approach to adaptive indexing. Stochastic cracking also uses each query as a hint on how to reorganize data, but not blindly so; it gains resilience and avoids performance bottlenecks by deliberately applying certain arbitrary choices in its decision making. Thereby, this technique brings adaptive indexing forward to a mature formulation that confers the workload-robustness previous approaches lacked. It has verified that stochastic cracking maintains the desired properties of original database cracking while at the same time it performs well with diverse realistic workloads. [1], while maintaining original properties of database cracking.
Stochastic cracking adopted four different techniques that try to strike a balance between
(a) Adding auxiliary reorganization steps with each query, and
(b) Remaining lightweight enough so as to significantly (if at all) not penalize individual queries.
The algorithms used for stochastic cracking are Data driven Center (DDC), Data driven Random (DDR), Variant of DDC and DDR (DDC1 and DDR1), Materialization data driven random1 (MDD1R) and Progressive Stochastic Cracking (PMDD1R).

V. Summary and Future Work

This paper gives an overview of database cracking. It has been shown that original cracking relies on the randomness of the workloads to converge well. However, where the workload is non-random, cracking needs to introduce randomness on its own. Stochastic Cracking clearly improves over original cracking by being robust in workload changes while maintaining all original cracking features when it comes to adaptation. Future work can provide more algorithms or optimization of existing algorithms.

VI. References

[1]. S. Idreos, "Database Cracking: Towards Auto-tuning Database Kernels," 2010 DBcrackingThesis.pdf
[2]. S. Idreos, S. Manegold, H. Kuno, and G. Graefe." Merging what's cracked, cracking what's merged:Adaptive indexing in main-memory column-stores". PVLDB, 4(9):585–597, 2011.
[3]. By Stratos Idreos" Database Cracking:Towards Auto-tuning Database Kernels", 2010
[4]. G. Graefe and H. Kuno." Adaptive indexing for relational keys." SMDB, pages 69–74, 2010
[5]. P. Boncz, A. Wilschut, and M. Kersten. Flattening an Object Algebra to Provide Performance. In Proc. Of the IEEE Int'l. Conf. on Data Engineering, 1998.
[6]. S. Idreos, M. L. Kersten, and S. Manegold. "Database cracking. CIDR", pages 68–78, 2007
[7]. Peter Boncz (CWI) Adapted from VLDB "Column-Oriented Database Systems "2009
[8]. Tutorial Column-Oriented Database Systems with Daniel Abadi (Yale) Stavros Harizopuolos (HP Labs)
[9]. By Martin Kersten, Stefan Manegold, joerd Mullender "The Database Architectures Research Group at CWI "
[10]. S. Idreos, M. L. Kersten, and S. Manegold." Self-organizing tuple reconstruction in column Stores". In SIGMOD, pages 297–308, 2009
[11]. Stratos Idreos, Martin Kersten and Stefan Manegold CWI Amsterdam, The Netherland "database cracking ppts " 2010

[12]. Felix Halim, Stratos Idreos, Panagiotis Karras, Roland H. C. Yap "Stochastic Database Cracking: Towards Robust Adaptive Indexing in Main-Memory Column-Stores" ,2012
[13]. Meenash Bhardwaj, Aarti Chugh " Optimization of Stochastic Database Cracking" in Cornell University Library.
[14]. Goetz Graefe: Sorting and indexing with partitioned B-trees. CIDR 2003.
[15]. M. Stonebraker, D. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S.
[16]. Madden, E.O'Neil, P. O'Neil, A. Rasin, N. Tran, and S. Zdonik. C-store: A column oriented
[17]. dbms. In Proc. of the Int'l. Conf. on Very Large Data Bases, 2005.
[18]. Stratos Idreos, Stefan Manegold and Goetz Graefe Ppts on "adaptive indexing in modern databases "
[19]. Goetz Graefe:" Sorting and indexing with partitioned B-trees". CIDR 2003
[20]. Goetz Graefe: Implementing sorting in database systems. ACM Comput. Surv. 38(3): (2006).
[21]. Milena Ivanova, Martin L. Kersten, Niels Nes: Self-organizing strategies for a column-store database.EDBT 2008: 157-168

