Database Cracking: Concept Evaluation

Zulsar Batmunkh
MIT

Dana Mukusheva
MIT

Abstract—abstract goes here.

1 Introduction

2 Background and related work

Blablabla one Nobody [3]. Blablabla two Nobody [1]. Blablabla three Nobody [2].

2.1 Database Cracking

2.2 Related Work

3 System Overview

In order to analyze the performance effects of the database cracking, we have implemented our own small database system MiniDB. In this section we will describe components and implementation details of MiniDB.

3.1 MiniDB overview

MiniDB is a simple single column database implemented in Java, where each table consists of a single uniquely named column. Database keeps a hash map with (column name, column object) pairs. The database tuples are 32-bit positive integers. Column tuples are stored as a list. MiniDB has three column types: Simple Column, Sorted Column, and Cracker Column. Simple Column does not maintain a specific tuple order and inserts a tuple to the end of the tuple list. In Simple Columns tuple lookup requires linear scan of the tuples list. Sorted Column preserves the order in the tuples list. Tuple lookup takes logarithmic time and is implemented as a binary search. Cracker Column stores tuples in a partially sorted manner, that is, once its tuples are reorganized and partitioned into two sublists (one with all tuples whose values are less than or equal to the partition value and one with all tuples whose values are greater than the partition value). CrackerColumns cannot exist independently in our database, they can only be coupled with Simple Columns that support cracking and initialized after the first query. Cracker Columns contain same values as their corresponding Simple Columns, but in a different and constantly changing order. For each Cracker Column, there is a Cracker Index instance. The Cracker Index instances are necessary to keep most-up-to date information about all partitions of the Cracker Column tuples. Particularly, Cracker Index stores (v, p) pairs, which indicate that all tuples located at the positions less than and including p have values less that and including v.

3.2 Cracker Index implementation

4 Experiments

4.1 Experiment Setup

Describe the hardware (server, memory, OS, number of cores, etc) Describe how we timed things

4.2 Performance Evaluation

4.3 Comparison to MonetDB

not sure if this comes here

- *Zulsar Batmunkh is MIT undergraduate. email: zulsar@mit.edu*
- *Dana Mukusheva is MIT undergraduate. email: mukushev@mit.edu*

5 Discussion

Talk about what needs to be done to improve the quality of our experiments, some assumptions we made and what we have neglected

6 Conclusion

References

[1] S. Idreos, M. L. Kersten, and S. Manegold. Database cracking. In *CIDR*, pages 68–78, 2007.
[2] M. Kersten and S. Manegold. Cracking the database store. In *CIDR*, 2005.
[3] F. M. Schuhknecht, A. Jindal, and J. Dittrich. The Uncracked Pieces in Database Cracking Management. *Proc. VLDB Endow.*, 7(2):97–108, Oct. 2013.