ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Institute of Information Systems
Moira C. Norrie
David Simmen, David Weber, Christoph Zimmerli

globis

# Information Systems
## Spring 2013

### Project 1

In the lecture the idea of column-oriented DBMS was presented as proposed by Stonebreaker et. al. [2]. In this paper the authors stated that column-oriented DBMS are fundamentally different from the existing row stores. Nicolas Bruno from Microsoft presented their take on the subject in another paper [1].

In this project, your task is to follow the argumentation of Bruno's paper and thus try to emulate column store behaviour in a row-oriented DBMS. You will thereby get hands-on experience with the effects of varying the physical schema for given data, indexing and see a query optimiser in action.

You can work on this project alone or in groups. Please submit your results (measurement data and analysis) until Tuesday, April 16, at noon to your assistant. The findings will then be discussed as part of the exercise sessions on April 18/19.

#### Exercise 1.1 *Preparation*

Read Bruno's paper. Especially pay attention to how data is represented in a row store to emulate columns and which DBMS features, such as (clustered) indexes and materialised views, are employed.

Choose a row-oriented DBMS for your implementation. Examples are PostgeSQL, MySQL/MariaDB, IBM DB2, Oracle and Microsoft SQL Server. Note that not all of them support the same features – e.g. materialised views are not found everywhere.

In the paper data and adapted queries from the TPC-H benchmark[1] are employed. Have a look at its scenario and the data generator `dbgen`. Note that the generator produces output for several DBMS – which may also influence your choice of product.

Install and configure your DBMS of choice. Use the TPC-H generator to generate your test data (e.g. using a scale factor of 1) and load it into your DMBS. Find out how to measure the execution time of queries in your DBMS and how to see the query plan chosen by the optimiser for a given query.

#### Exercise 1.2 *Row-oriented Baseline*

To test your setup and establish a baseline for further experiments, run the queries $Q_1$ to $Q_7$ from Figure 1 in the paper. Most queries contain a date parameter `D`. You can either choose several values for `D` and run the each query for each value, averaging the execution times. Or you can choose a fixed value for `D` such as `'1995-01-01'` (make sure the value occurs in the data). In either case run your queries several times, measuring individual execution times, and collect the results for statistical analysis.

#### Exercise 1.3 *Simple Column Schema*

In the beginning of Section 2 in the paper previous work is mentioned that tried a simple column representation in row stores, where tables are used as columns. Implement such a schema and populate it with data from the row-oriented schema. Rewrite the queries so that they now work on the new schema. Perform measurements on the new schema and queries.

[1] `http://www.tpc.org/tpch/`

What indexes could be be added to the column schema to increase the performance of our seven queries? Find out how to create indexes in your DBMS and add the appropriate ones. Also check the execution plans to see whether the optimiser deems the indexes useful.

**Exercise 1.4** *Materialised View Schema*

Section 2.1 in the paper describes the idea of improving the physical design with materialised views (MV). Note that for our scenario actual MV are not necessary since there are no insert/update/delete operations – only read-only queries. Create the MV schema, populate it with data from the row or column schema, rewrite the queries and measure their execution times.

In a second step again add indexes where appropriate and check their usefulness by additional measurements and looking at the query plans.

**Exercise 1.5** *C-Tables*

Section 2.2 in the paper proposes the use of C-tables as a tradeoff between single-column indexes and MV in terms of query performance and flexibility. Implement another schema based on C-tables and populate it with data from a previous schema. Make sure to also perform RLE when importing the data. Rewrite the queries (like in Section 2.2.2) for the new schema and measure their execution times. Then, again, try to improve things with indexing.

The final step is to consider the optimisations to query rewriting proposed in Section 2.2.3 and apply them to the queries where possible. Of course, we are also interested in the execution times after these rewrites.

**Exercise 1.6** *Analysis and Report*

Analyse the measurements collected in the previous steps and comment your findings. Also document your design choices and issues that needed to be resolved along the way. Be prepared to say a few words about your work during the project discussion in the exercise session.

**Exercise 1.7** *Automatisation (optional)*

You probably have performed all the schema changes, data import and query rewrites manually along the way so far. Would it be possible to automate these processes in a layer on top of the DBMS? Such a layer could allow the user to specify the schema, insert data and perform queries in traditional row-oriented manner. It would then behind the scenes translate schema, data and queries to other approaches investigated in this project. Implement such a layer for your DBMS.

# References


[1] N. Bruno. Teaching an Old Elephant New Tricks. In *4th Biennial Conference on Innovative Data Systems Research (CIDR 2009)*, Asilomar, Jan. 2009. Microsoft.

[2] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik. C-Store: A Column-oriented DBMS. In *Proc. Intl. Conf. on Very Large Databases (VLDB)*, pages 553–564, June 2005.