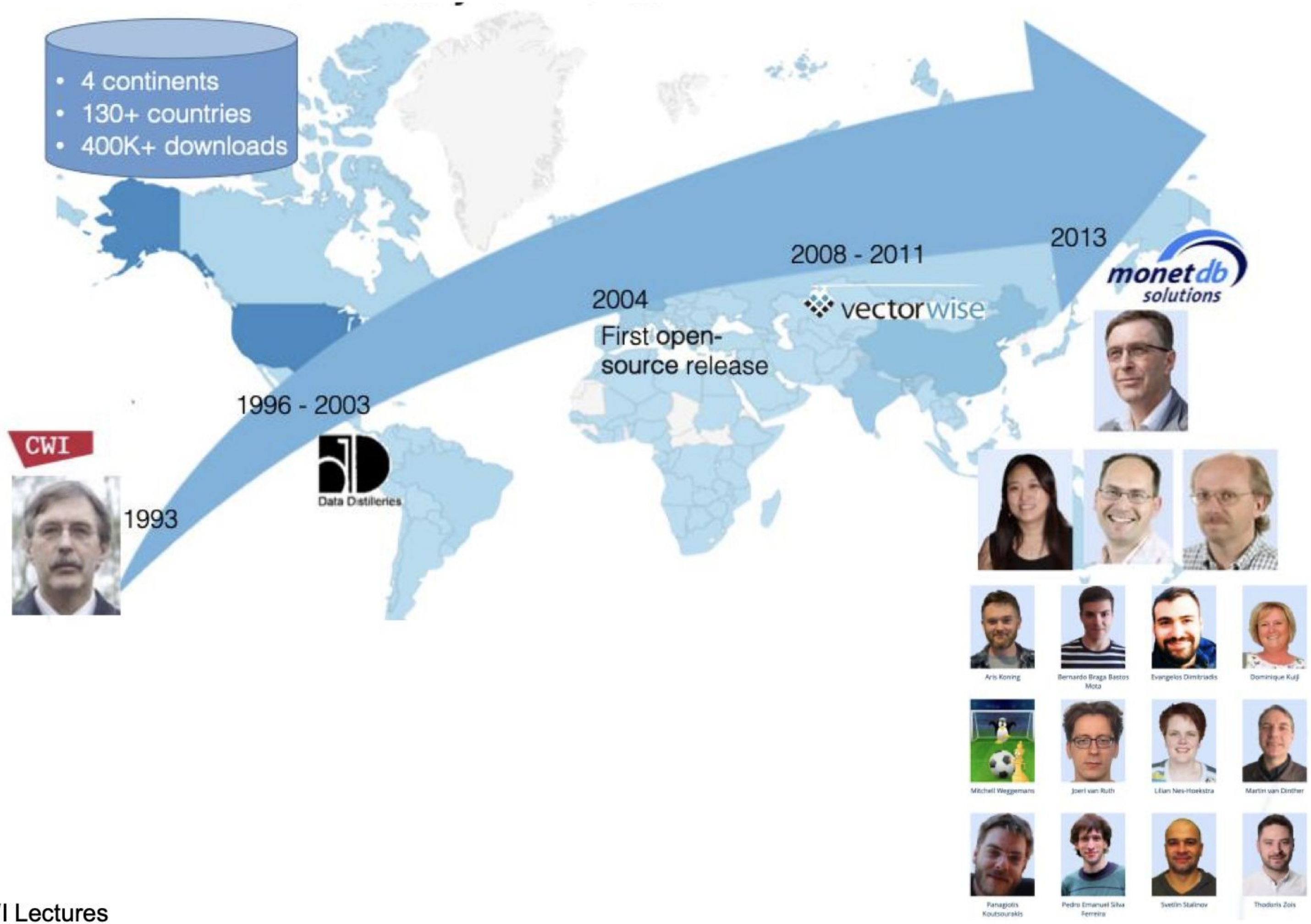
Martin Kersten Memorial

Peter Boncz

CIDR, The Mövenpick Hotel, Amsterdam, 8/1/2023


Conclusions and



• 4 continents
• 130+ countries
• 400K+ downloads
CWI
1993
1996 - 2003
Data Distilleries
2004
First open-source release
2008 - 2011
vectorwise
2013
monetdb solutions


Very Large Data Bases 2009 - Lyon, France


VIRTUS
NOBILI
TAT


monetdb


CWI
CWI Lectures on Database Research
monetdb

Very Large Data Bases 2009 - Lyon, France

Big Data Space Fungus

Martin Kersten

A Database System with Amnesia

Martin Kersten
CWI
Amsterdam, The Netherlands
martin.kersten@cwi.nl

Lefteris Sidirourgos
CWI
Amsterdam, The Netherlands
lsidir@cwi.nl

ABSTRACT

Big Data comes with huge challenges. Its volume and velocity makes handling, curating, and analytical processing a costly affair. Even to simply "look at" the data within an a priori defined budget and with a guaranteed interactive response time might be impossible to achieve. Commonly applied scale-out approaches will hit the technology and monetary wall soon, if not done so already. Likewise, blindly rejecting data when the channels are full, or reducing the data resolution at the source, might lead to loss of valuable observations.

An army of well-educated database administrators or full software stack architects might deal with these challenges albeit at substantial cost. This calls for a mostly knobless DBMS with a fundamental change in database management.

economic or scientific value. What was relevant before, becomes irrelevant and costly to keep around or digest. Data analysts are better served by deleting or moving irrelevant data out of the critical query execution path.

Furthermore, there is an end-user misconception that we can afford to keep everything around forever in (cold) storage. This is not true and only half of the story. Although storage for fast growing databases is cheap (e.g., AWS Glacier charges $48 per TB/year in 2016) using this data becomes prohibitively more expensive over time, both money wise and by input latency (AWS Glacier data retrieval cost is $ 2.5- 30 per TB and can take up to 12 hours). Although subsequent Cloud processing power is available, it is usually placed far from the cheap secondary storage. Thus, maintaining a wisely chosen subset of active records in faster memory is

An Architecture for Recycling Intermediates in a Column-store

Milena G. Ivanova Martin L. Kersten Niels J. Nes Romulo A.P. Gonçalves
{milena,mk,niels,goncalve}@cwi.nl
Centrum Wiskunde en Informatica (CWI)
P.O. Box 94079, 1090 GB Amsterdam
The Netherlands

ABSTRACT

Automatically recycling (intermediate) results is a grand challenge for state-of-the-art databases to improve both query response time and throughput. Tuples are loaded and streamed through a tuple-at-a-time processing pipeline avoiding materialization of intermediates as much as possible. This limits the opportunities for reuse of overlapping computations to DBA-defined materialized views and function/result cache tuning.

In contrast, the operator-at-a-time execution paradigm produces fully materialized results in each step of the query plan. To avoid resource contention, these intermediates are evicted as soon as possible.

In this paper we study an architecture that harvests the by-

1. INTRODUCTION

Query optimization and processing in off-the-shelf database systems is often still focused on individual queries. Queries are optimized in isolation using statistics gathered, analytical models, and heuristic rewrite rules, and ran against a kernel regardless opportunities offered by concurrent or previous invocations.

This approach is far from optimal and two directions to improve upon this situation are being actively explored: materialized views and reuse of (partial) results. Both depend and interact heavily with the underlying architecture, its execution paradigm and opportunities for optimizers to exploit transient information.

Database Cracking

Stratos Idreos
CWI Amsterdam
The Netherlands
Stratos.Idreos@cwi.nl

Martin L. Kersten
CWI Amsterdam
The Netherlands
Martin.Kersten@cwi.nl

Stefan Manegold
CWI Amsterdam
The Netherlands
Stefan.Manegold@cwi.nl

ABSTRACT

Database indices provide a non-discriminative navigational infrastructure to localize tuples of interest. Their maintenance cost is taken during database updates. In this paper, we study the complementary approach, addressing index maintenance as part of query processing using continuous physical reorganization, i.e., *cracking* the database into manageable pieces. The motivation is that by automatically organizing data the way users request it, we can achieve fast access and the much desired self-organized behavior.

We present the first mature cracking architecture and report on our implementation of cracking in the context of a full fledged relational system. It led to a minor enhancement to its relational algebra kernel, such that cracking could be piggy-backed without incurring too much processing overhead. Furthermore, we illustrate the ripple effect of dynamic reorganization on the query plans derived by the SQL opti-

In this paper, we explore a radically new approach in database architecture, called *database cracking*. The cracking approach is based on the hypothesis that index maintenance should be a byproduct of query processing, not of updates. Each query is interpreted not only as a request for a particular result set, but also as an *advice* to crack the physical database store into smaller pieces. Each piece is described by a query, all of which are assembled in a *cracker index* to speedup future search. The cracker index replaces the non-discriminative indices (e.g., B-trees and hash tables) with a discriminative index. Only database portions of past interest are easily localized. The remainder is unexplored territory and remains non-indexed until a query becomes interested. Continuously reacting on query requests brings the powerful property of self-organization. The cracker index is built dynamically while queries are processed and adapts to changing query workloads.

The Paper Pond Model

Martin Kersten

CWI, Amsterdam

Background

A growing concern in our field is the increased workload on authors and referees to get their work published. Conference PCs for conferences like VLDB and SIGMOD run into a few hundred people taking care of more than a thousand referee reports under pressure. Only 15% of the papers end up in the proceedings, and a roll-forward count from SIGMOD'05 to VLDB'05 showed that at least 30% of the drop outs were resubmitted more or less immediately.

Incremental improvements are hard to implement (shepherding, roll-forward with referees, deadline management, ...) and to assess on effectiveness (double-blind). In this short note, I propose a rad-

be consulted to improve the program (ako lightweight PC), e.g. judging and fine tuning the scientific program.

Any author has the right to refuse inclusion in a conference (or workshop) program. Their paper remains in the pool.

Authors are allowed to revise their paper twice a year to improve the chance of being picked. Each time two referees are added, while the history of the paper remains accessible to all. All previous referees can adjust their judgment as well.

Authors can try to get a paper published elsewhere, but its publication should at least be three months after the first submission to the PP. Authors can re-

Very Large Data Bases 2009 - Lyon, France


Rock and Reef Homes