ACM Books is a new series of high quality books for the computer science community, published by ACM in collaboration with Morgan & Claypool Publishers. ACM Books publications are widely distributed in both print and digital formats through booksellers and to libraries (and library consortia) and individual ACM members via the ACM Digital Library platform.

ABOUT ACM BOOKS

ACM Books is a new series of high quality books for the computer science community published by ACM in collaboration with Morgan & Claypool Publishers. ACM Books publications are widely distributed in both print and digital formats through booksellers and to libraries (and library consortia) and individual ACM members via the ACM Digital Library platform.
Making Databases Work

Book excerpt provided by

[Micro Focus logo]

[Vertica logo]
ACM Books

Editor in Chief
M. Tamer Özsu, University of Waterloo

ACM Books is a new series of high-quality books for the computer science community, published by ACM in collaboration with Morgan & Claypool Publishers. ACM Books publications are widely distributed in both print and digital formats through booksellers and to libraries (and library consortia) and individual ACM members via the ACM Digital Library platform.

Making Databases Work: The Pragmatic Wisdom of Michael Stonebraker
Editor: Michael L. Brodie
2018

The Handbook of Multimodal-Multisensor Interfaces, Volume 2: Signal Processing, Architectures, and Detection of Emotion and Cognition
Editors: Sharon Oviatt, Monash University
Björn Schuller, University of Augsburg and Imperial College London
Philip R. Cohen, Monash University
Daniel Sonntag, German Research Center for Artificial Intelligence (DFKI)
Gerasimos Potamianos, University of Thessaly
Antonio Krüger, Saarland University and German Research Center for Artificial Intelligence (DFKI)
2018

Declarative Logic Programming: Theory, Systems, and Applications
Editors: Michael Kifer, Stony Brook University
YanHong Annie Liu, Stony Brook University
2018

The Sparse Fourier Transform: Theory and Practice
Haitham Hassanieh, University of Illinois at Urbana-Champaign
2018

The Continuing Arms Race: Code-Reuse Attacks and Defenses
Editors: Per Larsen, Immunant, Inc.
Ahmad-Reza Sadeghi, Technische Universität Darmstadt
2018

Frontiers of Multimedia Research
Editor: Shih-Fu Chang, Columbia University
2018
Shared-Memory Parallelism Can Be Simple, Fast, and Scalable
Julian Shun, *University of California, Berkeley*
2017

Computational Prediction of Protein Complexes from Protein Interaction Networks
Sriganesh Srihari, *The University of Queensland Institute for Molecular Bioscience*
Chern Han Yong, *Duke-National University of Singapore Medical School*
Limsoon Wong, *National University of Singapore*
2017

The Handbook of Multimodal-Multisensor Interfaces, Volume 1: Foundations, User Modeling, and Common Modality Combinations
Editors: Sharon Oviatt, *Incaa Designs*
Björn Schuller, *University of Passau and Imperial College London*
Philip R. Cohen, *Voicebox Technologies*
Daniel Sonntag, *German Research Center for Artificial Intelligence (DFKI)*
Gerasimos Potamianos, *University of Thessaly*
Antonio Krüger, *Saarland University and German Research Center for Artificial Intelligence (DFKI)*
2017

Communities of Computing: Computer Science and Society in the ACM
Thomas J. Misa, Editor, *University of Minnesota*
2017

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining
ChengXiang Zhai, *University of Illinois at Urbana–Champaign*
Sean Massung, *University of Illinois at Urbana–Champaign*
2016

An Architecture for Fast and General Data Processing on Large Clusters
Matei Zaharia, *Stanford University*
2016

Reactive Internet Programming: State Chart XML in Action
Franck Barbier, *University of Pau, France*
2016

Verified Functional Programming in Agda
Aaron Stump, *The University of Iowa*
2016

The VR Book: Human-Centered Design for Virtual Reality
Jason Jerald, *NextGen Interactions*
2016
Ada’s Legacy: Cultures of Computing from the Victorian to the Digital Age
Robin Hammerman, Stevens Institute of Technology
Andrew L. Russell, Stevens Institute of Technology
2016

Edmund Berkeley and the Social Responsibility of Computer Professionals
Bernadette Longo, New Jersey Institute of Technology
2015

Candidate Multilinear Maps
Sanjam Garg, University of California, Berkeley
2015

Smarter Than Their Machines: Oral Histories of Pioneers in Interactive Computing
John Cullinane, Northeastern University; Mossavar-Rahmani Center for Business and Government, John F. Kennedy School of Government, Harvard University
2015

A Framework for Scientific Discovery through Video Games
Seth Cooper, University of Washington
2014

Trust Extension as a Mechanism for Secure Code Execution on Commodity Computers
Bryan Jeffrey Parno, Microsoft Research
2014

Embracing Interference in Wireless Systems
Shyamnath Gollakota, University of Washington
2014
Making Databases Work

The Pragmatic Wisdom of Michael Stonebraker

Michael L. Brodie
Massachusetts Institute of Technology

ACM Books #22
A publication in the ACM Books series, #22
Editor in Chief: M. Tamer Özsu, University of Waterloo

This book was typeset in Arnhem Pro 10/14 and Flama using ZZTEX.
First Edition
10 9 8 7 6 5 4 3 2 1
This book is dedicated to Michael Stonebraker, Jim Gray, Ted Codd, and Charlie Bachman, recipients of the ACM A.M. Turing Award for the management of data, one of the world's most valuable resources, and to their many collaborators, particularly the contributors to this volume.
Contents

Data Management Technology Kairometer: The Historical Context  xxvii
Foreword  xxix
Preface  xxxi
Introduction  1

Michael L. Brodie
A Brief History of Databases  1
Preparing to Read the Stories and What You Might Find There  6
A Travel Guide to Software Systems Lessons in Nine Parts  7

PART I  2014 ACM A.M. TURING AWARD PAPER AND LECTURE  13

The Land Sharks Are on the Squawk Box  15

Michael Stonebraker
Off to a Good Start  16
First Speedbumps  22
Another High  26
The High Does Not Last  28
The Future Looks Up (Again)  30
The Good Times Do Not Last Long  30
The Stories End  31
Why a Bicycle Story?  32
The Present Day  35
References  36
PART II  MIKE STONEBRAKER’S CAREER  39

Chapter 1  Make it Happen: The Life of Michael Stonebraker  41

Samuel Madden

Synopsis  41
Early Years and Education  42
Academic Career and the Birth of Ingres  43
The Post-Ingres Years  45
Industry, MIT, and the New Millennium  46
Stonebraker’s Legacy  47
Companies  48
Awards and Honors  49
Service  49
Advocacy  50
Personal Life  50
Acknowledgments  50

Mike Stonebraker’s Student Genealogy Chart  53
The Career of Mike Stonebraker: The Chart  55

PART III  MIKE STONEBRAKER SPEAKS OUT: AN INTERVIEW WITH MARIANNE WINSLETT  57

Chapter 2  Mike Stonebraker Speaks Out: An Interview  59

Marianne Winslett

PART IV  THE BIG PICTURE  85

Chapter 3  Leadership and Advocacy  87

Philip A. Bernstein

Systems  87
Mechanisms  90
Advocacy  91

Chapter 4  Perspectives: The 2014 ACM Turing Award  93

James Hamilton
Chapter 5  Birth of an Industry; Path to the Turing Award  97

Jerry Held

Birth of an Industry (1970s)  97
Ingres—Timing  98
Ingres—Team  99
Ingres—Competition  100
Ingres—Platform  101
Adolescence with Competition (1980s and 1990s)  101
Competing with Oracle  102
Competing with Oracle (Again)  102
Maturity with Variety (2000s and 2010s)  103
Vertica  104
VoltDB  104
Tamr  105
The Bottom Line  105

Chapter 6  A Perspective of Mike from a 50-Year Vantage Point  107

David J. DeWitt

Fall 1970—University of Michigan  107
Fall 1976—Wisconsin  108
Fall 1983—Berkeley  111
1988–1995—No Object Oriented DBMS Detour for Mike  111
2000—Project Sequoia  112
2003—CIDR Conference Launch  113
2005—Sabbatical at MIT  113
2008—We Blog about “MapReduce”  114
2014—Finally, a Turing Award  114
2016—I Land at MIT  115
2017  115

PART V  STARTUPS  117

Chapter 7  How to Start a Company in Five (Not So) Easy Steps  119

Michael Stonebraker

Introduction  119
Step 1: Have a Good Idea  119
Chapter 16  Looking Back at Postgres  205  
Joseph M. Hellerstein

Context 205
Postgres: An Overview 206
Log-centric Storage and Recovery 213
Software Impact 218
Lessons 223
Acknowledgments 224

Chapter 17  Databases Meet the Stream Processing Era  225  
Magdalena Balazinska, Stan Zdonik

Origins of the Aurora and Borealis Projects 225
The Aurora and Borealis Stream-Processing Systems 227
Concurrent Stream-Processing Efforts 231
Founding StreamBase Systems 232
Stream Processing Today 233
Acknowledgments 234

Chapter 18  C-Store: Through the Eyes of a Ph.D. Student  235  
Daniel J. Abadi

How I Became a Computer Scientist 235
The Idea, Evolution, and Impact of C-Store 238
Building C-Store with Mike 240
Founding Vertica Systems 242

Chapter 19  In-Memory, Horizontal, and Transactional:  
The H-Store OLTP DBMS Project  245  
Andy Pavlo

System Architecture Overview 246
First Prototype (2006) 247
VoltDB (2009–Present) 250
H-Store/VoltDB Split (2010–2016) 251
Conclusion 251
Chapter 20  Scaling Mountains: SciDB and Scientific Data Management  253

Paul Brown

Selecting Your Mountain  254
Planning the Climb  256
Expedition Logistics  259
Base Camp  260
Plans, Mountains, and Altitude Sickness  263
On Peaks  267
Acknowledgments  268

Chapter 21  Data Unification at Scale: Data Tamer  269

Ihab Ilyas

How I Got Involved  269
Data Tamer: The Idea and Prototype  270
The Company: Tamr Inc.  273
Mike's Influence: Three Lessons Learned.  276

Chapter 22  The BigDAWG Polystore System  279

Tim Mattson, Jennie Rogers, Aaron J. Elmore

Big Data ISTC  279
The Origins of BigDAWG  280
One Size Does Not Fit All and the Quest for Polystore Systems  282
Putting it All Together  284
Query Modeling and Optimization  285
Data Movement  286
BigDAWG Releases and Demos  287
Closing Thoughts  288

Chapter 23  Data Civilizer: End-to-End Support for Data Discovery, Integration, and Cleaning  291

Mourad Ouazzani, Nan Tang, Raul Castro Fernandez

We Need to Civilize the Data  292
The Day-to-Day Life of an Analyst  292
Designing an End-to-End System  294
Data Civilizer Challenges  295
Concluding Remarks  300
PART VII.B  CONTRIBUTIONS FROM BUILDING SYSTEMS  301

Chapter 24  The Commercial Ingres Codeline  303

Paul Butterworth, Fred Carter

Research to Commercial  304
Conclusions  309
Open Source Ingres  309

Chapter 25  The Postgres and Illustra Codelines  311

Wei Hong

Postgres: The Academic Prototype  311
Illustra: “Doing It for Dollars”  313
PostgreSQL and Beyond  317
Open Source PostgreSQL  318
Final Thoughts  319

Chapter 26  The Aurora/Borealis/ StreamBase Codelines: A Tale of Three Systems  321

Nesime Tatbul

Aurora/Borealis: The Dawn of Stream Processing Systems  322
From 100K+ Lines of University Code to a Commercial Product  326
Encounters with StreamBase Customers  327
“Over My Dead Body” Issues in StreamBase  328
An April Fool’s Day Joke, or the Next Big Idea?  330
Concluding Remarks  331
Acknowledgments  332

Chapter 27  The Vertica Codeline  333

Shilpa Lawande

Building a Database System from Scratch  333
Code Meets Customers  334
Don’t Reinvent the Wheel (Make It Better)  335
Architectural Decisions: Where Research Meets Real Life  336
Customers: The Most Important Members of the Dev Team  339
Conclusion  340
Acknowledgments  340
Chapter 28  The VoltDB Codeline  341  

John Hugg  

Compaction  342  
Latency  344  
Disk Persistence  346  
Latency Redux  347  
Conclusion  348  

Chapter 29  The SciDB Codeline: Crossing the Chasm  349  

Kriti Sen Sharma, Alex Poliakov, Jason Kinchen  

Playing Well with Others  349  
You Can't Have Everything (at Once)  351  
In Hard Numbers We Trust  352  
Language Matters  353  
Security is an Ongoing Process  354  
Preparing for the (Genomic) Data Deluge  354  
Crossing the Chasm: From Early Adopters to Early Majority  355  

Chapter 30  The Tamr Codeline  357  

Nikolaus Bates-Haus  

Neither Fish nor Fowl  358  
Taming the Beast of Algorithmic Complexity  359  
Putting Users Front and Center  361  
Scaling with Respect to Variety  362  
Conclusion  365  

Chapter 31  The BigDAWG Codeline  367  

Vijay Gadeppally  

Introduction  367  
BigDAWG Origins  370  
First Public BigDAWG Demonstration  371  
Refining BigDAWG  373  
BigDAWG Official Release  375  
BigDAWG Future  376
PART VIII  PERSPECTIVES  377

Chapter 32  IBM Relational Database Code Bases  379

  James Hamilton
  Why Four Code Bases?  379
  The Portable Code Base Emerges  381
  Looking Forward  384

Chapter 33  Aurum: A Story about Research Taste  387

  Raul Castro Fernandez

Chapter 34  Nice: Or What It Was Like to Be Mike's Student  393

  Marti Hearst

Chapter 35  Michael Stonebraker: Competitor, Collaborator, Friend  397

  Don Haderle

Chapter 36  The Changing of the Database Guard  403

  Michael L. Brodie
  Dinner with the Database Cognoscenti  403
  The Great Relational-CODASYL Debate  404
  Mike: More Memorable than the Debate, and Even the Cheese  406
  A Decade Later: Friend or Foe?  407

PART IX  SEMINAL WORKS OF MICHAEL STONEBRAKER AND HIS COLLABORATORS  409

OLTP Through the Looking Glass, and What We Found There  411

  Stavros Harizopoulos, Daniel J. Abadi,
  Samuel Madden, Michael Stonebraker

  Abstract  411
  1  Introduction  412
  2  Trends in OLTP  416
  3  Shore  418
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Performance Study 424</td>
</tr>
<tr>
<td>5 Implications for Future OLTP Engines 433</td>
</tr>
<tr>
<td>6 Related Work 436</td>
</tr>
<tr>
<td>7 Conclusions 436</td>
</tr>
<tr>
<td>8 Acknowledgments 437</td>
</tr>
<tr>
<td>9 Repeatability Assessment 437</td>
</tr>
<tr>
<td>References 437</td>
</tr>
</tbody>
</table>

“One Size Fits All”: An Idea Whose Time Has Come and Gone 441

*Michael Stonebraker, Ugur Çetintemel*

Abstract 441
1 Introduction 441
2 Data Warehousing 443
3 Stream Processing 445
4 Performance Discussion 448
5 One Size Fits All? 455
6 A Comment on Factoring 458
7 Concluding Remarks 460
References 460

The End of an Architectural Era (It’s Time for a Complete Rewrite) 463

*Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, Pat Helland*

Abstract 463
1 Introduction 464
2 OLTP Design Considerations 466
3 Transaction, Processing and Environment Assumptions 470
4 H-Store Sketch 473
5 A Performance Comparison 479
6 Some Comments about a “One Size Does Not Fit All” World 483
7 Summary and Future Work 486
References 488
C-Store: A Column-Oriented DBMS  491

Mike Stonebraker, Daniel J. Abadi,
Adam Batkin, Xuedong Chen,
Mitch Cherniack, Miguel Ferreira,
Edmond Lau, Amerson Lin,
Sam Madden, Elizabeth O’Neil,
Pat O’Neil, Alex Rasin,
Nga Tran, Stan Zdonik

Abstract  491
1 Introduction  492
2 Data Model  496
3 RS  500
4 WS  502
5 Storage Management  502
6 Updates and Transactions  503
7 Tuple Mover  508
8 C-Store Query Execution  509
9 Performance Comparison  511
10 Related Work  515
11 Conclusions  516

Acknowledgements and References  517

The Implementation of POSTGRES  519

Michael Stonebraker, Lawrence A. Rowe, Michael Hirohama

I Introduction  520
II The POSTGRES Data Model and Query Language  521
III The Rules System  538
IV Storage System  547
V The POSTGRES Implementation  550
VI Status and Performance  554
VII Conclusions  555

References  557

The Design and Implementation of INGRES  561

Michael Stonebraker,
Eugene Wong,
## Contents

*Peter Kreps, Gerald Held*

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>562</td>
</tr>
<tr>
<td>2</td>
<td>The INGRES Process Structure</td>
<td>571</td>
</tr>
<tr>
<td>3</td>
<td>Data Structures and Access Methods</td>
<td>575</td>
</tr>
<tr>
<td>4</td>
<td>The Structure of Process 2</td>
<td>585</td>
</tr>
<tr>
<td>5</td>
<td>Process 3</td>
<td>591</td>
</tr>
<tr>
<td>6</td>
<td>Utilities in Process 4</td>
<td>599</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion and Future Extensions</td>
<td>602</td>
</tr>
<tr>
<td></td>
<td>Acknowledgment</td>
<td>603</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>603</td>
</tr>
<tr>
<td></td>
<td>The Collected Works of Michael Stonebraker</td>
<td>607</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>635</td>
</tr>
<tr>
<td></td>
<td>Index</td>
<td>645</td>
</tr>
<tr>
<td></td>
<td>Biographies</td>
<td>671</td>
</tr>
</tbody>
</table>
The Vertica Codeline

Shilpa Lawande

The Vertica Analytic Database unequivocally established column-stores as the superior architecture for large-scale analytical workloads. Vertica’s journey started as a research project called C-Store, a collaboration by professors at MIT, Brown, Brandeis, and UMass Boston. When Michael Stonebraker and his business partner Andy Palmer decided to commercialize it in 2005, C-Store existed in the form of a research paper that had been sent for publication to VLDB (but not yet accepted) and a C++ program that ran exactly seven simple queries from TPC-H out of the box—it has no SQL front-end or query optimizer, and in order to run additional queries, you had to code the query plan in C++ using low level operators! Six years later (2011), Vertica was acquired by Hewlett-Packard Enterprise (HPE). The Vertica Analytics Engine—its code and the engineers behind it—became the foundation of HPE’s “big data” analytics solution.

What follows are some highlights from the amazing Vertica journey, as retold by members of its early engineering team. And some lessons we learned along the way.

Building a Database System from Scratch

My involvement with Vertica started in March 2005 when I came across a job ad on Monster.com that said Stonebraker Systems: “Building some interesting technology for data warehousing.” As someone who was getting bored at Oracle and had studied Mike’s Red Book1 during my DB classes at University of Wisconsin-Madison, I was intrigued, for sure. My homework after the first interview was—you guessed it—read the C-Store paper [Stonebraker et al. 2005a] and be ready to discuss it with Mike (a practice we continued to follow, except eventually the paper was replaced with the C-Store Seven Years Later paper [Lamb et al. 2012], and the

interview conducted by one or more senior developers). I do not recall much of that first interview but came away inspired by Mike’s pitch: “It doesn’t matter whether we succeed or fail. You would have built an interesting system. How many people in the world get to build a database system from scratch?” And that’s why I joined Vertica (see Chapter 18).

The early days were filled with the usual chaos that is the stuff of startups: hard stuff like getting the team to jell, easier stuff like writing code, more hard stuff like sorting through disagreements on whether to use push- or pull-based data-flow operators (and whether the building was too hot for the guys or too cold for me), writing some more code, and so on.

In the summer of 2005, we hired Chuck Bear, who at the time was living out of his last company’s basement and working his way down the Appalachian Trail. After Chuck’s interview, Mike barged into the engineering meeting saying, “We must do whatever it takes to hire this guy!” And since the team was fully staffed, Chuck got asked to do “performance testing.” It did not take long for everyone to realize that Chuck’s talents were underutilized as a “tester” (as Mike called quality assurance engineers). There was one occasion where Chuck couldn’t convince one of the engineers that we could be way faster than C-Store, so, over the next few nights, while his tests were running, he wrote a bunch of code that ran $2 \times$ faster than what was checked in!

The first commercial version of Vertica was already several times faster than C-Store, and we were only just getting going, a fantastic feat of engineering! From here on, C-Store and Vertica evolved along separate paths. Vertica went on to build a full-fledged petabyte-scale distributed database system, but we did keep in close touch with the research team, sharing ideas, especially on query execution with Daniel Abadi and Sam Madden, on query optimization with Mitch Cherniack at Brandeis, and on automatic database design with Stan Zdonik and Alex Rasin at Brown. Vertica had to evolve many of the ideas in the C-Store paper from real-world experience, but the ideas in Daniel Abadi’s Ph.D. thesis on compressed column stores still remained at the heart of Vertica’s engine, and we should all be glad he chose computer science over medicine.

**Lesson.** In effective software engineering organizations, the best ideas win. Shared ownership of the code base is essential. And, if you can’t resolve a disagreement with words, do it with code.

**Code Meets Customers**

The codeline journey of Vertica was a good example of what is called a “Lean Startup” these days—again Mike was ahead of his time (see Chapter 7). The first
version “Alpha” was supposed to only do the seven C-Store queries, but with an SQL front-end, not C++ and run on a single node. To do this, the decision was to use a “brutalized Postgres” (see Chapter 16), throwing away everything except its parser and associated data structures (why reinvent the wheel?) and converting it from a multi-process model to a single-process multi-threaded model. Also left out by choice: a lot of things that you can’t imagine a database not being able to do!

Omer Trajman was one of the early engineers. He later went on to run the Field Engineering team (charged with helping deploy Vertica in customer sites). He recalls:

One of these choices was pushing off the implementation of delete, a crazy limitation for a new high-performance database. In the first commercial versions of Vertica, if a user made a mistake loading data, the data couldn’t be changed, updated, or even deleted. The only command available to discard data was to drop the database and start over. As a workaround to having to reload data from flat files, the team later added INSERT/SELECT to order to create a copy of loaded data with some transformation applied, including removing rows. After adding the ability to rename and drop tables, the basic building blocks to automate deletes were in place. As it turns out, this was the right decision for Vertica’s target market.

The Vertica team found that there were two types of ideal early customers: those whose data almost never changed, and those whose data changed all the time. For people with relatively static data, Vertica provided the fastest and most efficient response times for analytics. For people whose data changed all the time, Vertica was able to go from raw data to fast queries more quickly than any other solution in the market. To get significant value from Vertica, neither customer type needed to delete data beyond dropping tables. Customers with data that rarely changed were able to prepare it and make sure it was properly loaded. Customers with rapidly changing data did not have the time to make corrections. Mike and the team had a genuine insight that at the time seemed ludicrous: a commercial database that can’t delete data.

Lesson. Work with customers, early and often. Listen carefully. Don’t be constrained by conventional wisdom.

Don’t Reinvent the Wheel (Make It Better)

Discussions about what to build and what not weren’t without a share of haggling between the professors who wrote the academic C-Store paper [Stonebraker et al. 2005a] and engineers who were building the real world Vertica. Here’s Chuck Bear recounting those days.
Back in 2006, the professors used to drop by Vertica every week to make sure we (the engineers) were using good designs and otherwise building the system correctly. When we told Mike and Dave DeWitt\footnote{David DeWitt (see Chapter 6), on Vertica’s technical advisory board, often visited the Vertica team.} that we were mulling approaches to multiple users and transactions, maybe some sort of optimistic concurrency control or multi-versioning, they yelled at us and said, in so many words, “Just do locking! You don’t understand locking! We’ll get you a copy of our textbook chapter on locking!” Also, they told us to look into the Shore storage manager \cite{shore1994}, thinking maybe we could reuse its locking implementation.

We read the photocopy of the chapter on locking that they provided us, and the following week we were prepared. First, we thanked the professors for their suggested reading material. But then we hit them with the hard questions . . . “How does locking work in a system like Vertica where writers don’t write to the place where readers read? If you have a highly compressed table, won’t a page-level lock on an RLE\footnote{Run Length Encoding} column essentially lock the whole table?”

In the end, they accepted our compromise idea, that we’d “just do locking” for transaction support, but at the table level, and additionally readers could take snapshots so they didn’t need any locks at all. The professors agreed that it was a reasonable design for the early versions, and in fact it remains this way over ten years later.

That’s the way lots of things worked. If you could get a design that was both professor-approved and that the engineers figured they could build, you had a winner.

**Lesson.** This decision is a great case study for “Keep it simple, stupid,” (aka KISS principle) and “Build for the common case,” two crucial systems design principles that are perhaps taught in graduate school but can only be cemented through the school of hard knocks.

**Architectural Decisions: Where Research Meets Real Life**

The decision about locking was an example of something we learned over and over during Vertica’s early years: that “professors aren’t always right” and “the customer always wins.”

The 2012 paper “The Vertica Analytic Database: C-Store 7 years later “ \cite{lamb2012} provides a comprehensive retrospective on the academic proposals from the original C-Store paper that survived the test of real-world deployments—and others that turned out to be spectacularly wrong.
For instance, the idea of permutations\(^4\) was a complete disaster. It slowed the system down to the point of being useless and was abandoned very early on. Late materialization of columns worked to an extent, for predicates and simple joins, but did not do so well once more complex joins were introduced. The original assumption that most data warehouse schemas [Kimball and Ross 2013] were “Star” or “Snowflake” served the system well in getting some early customers but soon had to be revisited. The optimizer was later adapted for “almost star” or “inverted snowflake” schemas and then was ultimately completely rewritten to be a general distributed query optimizer. Eventually, Vertica’s optimizer and execution engine did some very clever tricks, including leveraging information on data segmentation during query optimization (vs. building a single node plan first and then parallelizing it, as most commercial optimizers tend to do); delaying optimizer decisions like type of join algorithm until runtime; and so on.

Another architectural decision that took several iterations and field experience to get right was the design of the Tuple Mover. Here’s Dmitry Bochkov, the early lead engineer for this component, reminiscing about his interactions with Mike during this time.

The evolution of the Tuple Mover design in the first versions of Vertica demonstrated to me Mike’s ability to support switching from academic approach to “small matters of engineering” and back. What started as a simple implementation of an LSM (log-structured merge-tree) quickly degenerated into a complicated, low-performance component plagued by inefficient multiple rewrites of the same data and a locking system that competed with the Execution Engine and Storage Access Layer locking mechanisms.

It took a few rounds of design sessions that looked more like thesis defense, and I will forever remember the first approving nod I received from Mike. What followed was that the moveout and mergeout algorithms ended up using “our own dog food.” Our own Execution Engine was used for running the Tuple Mover operations to better handle transactions, resources planning, failover, and reconciliation among other tasks. And while it added significant pressure on other components, it allowed the Tuple Mover to become an integral part of Dr. Stonebraker’s vision of a high-performance distributed database.

Anyone who has worked with Mike knows he is a man of few words, and if you listen carefully, you can learn a massive amount from his terseness. If you

---

4. The idea that multiple projections in different sort orders could be combined at runtime to recreate the full table. Eventually, it was replaced by the notion of a super projection that contains all the columns.
worked at Vertica in the early days, you often heard Mike-isms, such as “buying a downstream farm” (referring to “engineering debt”)⁵ and the famous “over Mike’s dead body” (OMDB). These phrases referred to all the “bells and whistles” that database systems are filled with that Vertica would never build, perfectly capturing the tension between “research” and “real-life” choices that Vertica faced repeatedly over its life.

Min Xiao,⁶ founding engineer turned sales engineer, describes an OMDB encounter with Mike.

One day in 2008, I came back to the office after visiting a global bank customer. I saw that Mike, wearing a red shirt, sat in a small corner conference room working on his laptop. I stepped in and told him that the bank needed the feature of disaster recovery (DR) from Vertica. In the past, Mike had always wanted me to let him know the product requests from the customers. For this customer, their primary Vertica instance was in Manhattan and they wanted a DR instance in New Jersey. They had used Oracle for the same project prior to Vertica and therefore also hoped to have a statement-by-statement-via-change-data-capture type of DR. Mike listened to me for a minute. Apparently, he had heard the request from someone else and didn’t look surprised at all. He looked at me and calmly said “They don’t need that type of DR solution. All they need is an active replication thru parallel loading.” As always, the answer was concise as well as precise. While I took a moment to digest his answer, he noticed my hesitation and added “over my dead body.” I went back to the customer and communicated with them about the proposal of having a replicated copy. The bank wasn’t overly excited but didn’t raise the DR request anymore. Meanwhile, one of our largest (non-bank) customers, who had never used Oracle, implemented exactly what Mike had proposed and was very happy with it. They loaded into two 115-node clusters in parallel and used them to recover from each other.

Lesson. Complexity is often the Achilles’ heel of large-scale distributed systems, and as Daniel Abadi describes in vivid detail in Chapter 18, Mike hated complexity. With the liberally used phrase, OMDB, Mike forced us to think hard about every feature we added, to ensure it was truly required, a practice that served us well as our customer base grew. One of the reasons for Vertica’s success was that we thought very hard about what NOT to add, even though there was a ton of pressure from

⁵. A farm downstream along a river will always be flooded and may appear to be cheaper. This is an analogy for engineering debt, decisions made to save short-term coding work that required a ton of time and effort (i.e., cost) in the long run.
⁶. Min Xiao followed Mike and Andy Palmer to join the founding team of Tamr, Inc.
customers. Sometimes we had to relent on some earlier decisions as the system evolved to serve different classes of customers, but we still always thought long and hard about taking on complexity.

**Customers: The Most Important Members of the Dev Team**

Just as we thought hard about what features to add, we also listened very carefully to what customers were *really* asking for. Sometimes customers would ask for a feature, but we would dig into what problem they faced instead and often find that several seemingly different requests could often be fulfilled with one “feature.” Tight collaboration between engineering and customers became a key aspect of our culture from early on. Engineers thrived from hearing about the problems customers were having. Engineering, Customer Support, and Field Engineers all worked closely together to determine solutions to customer problems and the feedback often led to improvements, some incremental, but sometimes monumental.

The earliest example of such a collaboration was when one of the largest algorithm trading firms became a customer in 2008. Min Xiao recalls a day trip by the founders of this trading firm to our office in Billerica, Massachusetts, one Thursday afternoon.

Their CTO was a big fan of Mike. After several hours of intense discussions with us, we politely asked if they needed transportation to the airport. (This was before the days of Uber.) Their CEO casually brushed aside our request. Only later we found out that they had no real schedule constraints because they had flown in their own corporate jet. Not only that, but once he found out that Mike played the banjo, the next day he brought his bass guitar to the Vertica office. Mike, Stan Zdonik (a professor in Brown University), and John “JR” Robinson (a founding engineer of Vertica) played bluegrass together for several hours. This wasn’t an isolated “Mike fan”: customers loved and respected Mike for his technical knowledge and straight talk. We often joked that he was our best salesperson ever. :-)

Over time, this customer became a very close development partner to Vertica. They voluntarily helped us build Time-series Window functions, a feature-set that was originally on the “OMDB” list. Due to Vertica’s compressed and sorted columnar data storage, many of the windowing functions, which often take a long time to execute in other databases, could run blazingly fast in Vertica.

I recall the thrill that engineers felt to see the fruits of their work in practice. It was a day of great celebration for the engineering team when this customer reached a milestone running sub-second queries on 10 trillion rows of historical
trading data! These time-series functions later become one of the major performance differentiators for Vertica, and enabled very sophisticated log analytics to be expressed using rather simple SQL commands.

A big technical inflection point for Vertica came around 2009, when we started to land customers in the Web and social gaming areas. These companies really pushed Vertica’s scale to being able to handle petabytes of data in production. It took many iterations to really get “trickle loads” to work, but in the end this customer had an architecture where every click from all their games went into the database, and yet they were able to update analytical models in “near real-time.”

Another inflection point came when a very high profile social media customer decided to run Vertica on 300 nodes of very cheap and unreliable hardware. Imagine our shock when we got the first support case on a cluster of this size! This customer forced the team to really think about high availability and the idea that nodes could be down any time. As result, the entire system—from the catalog to recovery to cluster expansion—had to be reviewed for this use case. By this time, more and more customers wanted to run on the cloud, and all this work proved invaluable to support that use case.

Lesson. Keep engineers close to customers. Maybe make some music together. Listen carefully to their problems. Collaborate with them on solutions. Don’t be afraid to iterate. There is no greater motivator for an engineer than to find out his or her code didn’t work in the real world, nor greater reward than seeing their code make a difference to a customer’s business!

Conclusion
Vertica’s story is one of a lot of bold bets, some of which worked right from academic concept, and others that took a lot of hard engineering to get right. It is also a story of fruitful collaboration between professors and engineers. Most of all, it is a story of how a small startup, by working closely with customers, can change the prevailing standard of an industry, as Vertica did to the practices of data warehousing and big data analytics.

Acknowledgments
Thank you to Chuck Bear, Dmitry Bochkov, Omer Trajman, and Min Xiao of the early Vertica Engineering team for sharing their stories for this chapter.
This book celebrates Michael Stonebraker's accomplishments that led to his 2014 ACM A.M. Turing Award "for fundamental contributions to the concepts and practices underlying modern database systems."

The book describes, for the broad computing community, the unique nature, significance, and impact of Mike's achievements in advancing modern database systems over more than forty years. Today, data is considered the world's most valuable resource, whether it is in the tens of millions of databases used to manage the world's businesses and governments, in the billions of databases in our smartphones and watches, or residing elsewhere, as yet unmanaged, awaiting the elusive next generation of database systems. Every one of the millions or billions of databases includes features that are celebrated by the 2014 Turing Award and are described in this book.

Why should I care about databases? What is a database? What is data management? What is a database management system (DBMS)? These are just some of the questions that this book answers, in describing the development of data management through the achievements of Mike Stonebraker and his over 200 collaborators. In reading the stories in this book, you will discover core data management concepts that were developed over the two greatest eras—so far—of data management technology.

The book is a collection of 36 stories written by Mike and 38 of his collaborators: 23 world-leading database researchers, 11 world-class systems engineers, and 4 business partners. If you are an aspiring researcher, engineer, or entrepreneur you might read these stories to find these turning points as practice to tilt at your own computer-science windmills, to spur yourself to your next step of innovation and achievement.