Looking Back at Postgres

Joseph M. Hellerstein

Postgres was Michael Stonebraker’s most ambitious project—his grand effort to build a one-size-fits-all database system. A decade long, it generated more papers, Ph.Ds., professors, and companies than anything else he did. It also covered more technical ground than any other single system he built. Despite the risk inherent in taking on that scope, Postgres also became the most successful software artifact to come out of Stonebraker’s research groups, and his main contribution to open source. As of the time of writing, the open-source PostgreSQL system is the most popular, independent open-source database system in the world, and the fourth most popular database system in the world. Meanwhile, companies built from a Postgres base have generated a sum total of over $2.6 billion in acquisitions. By any measure, Stonebraker’s Postgres vision resulted in enormous and ongoing impact.

Context

Stonebraker had enormous success in his early career with the Ingres research project at Berkeley (see Chapter 15), and the subsequent startup he founded with Larry Rowe and Eugene Wong: Relational Technology, Inc. (RTI).

As RTI was developing in the early 1980s, Stonebraker began working on database support for data types beyond the traditional rows and columns of Codd’s original relational model. A motivating example current at the time was to provide database support for CAD tools for the microelectronics industry. In a 1983 paper, Stonebraker and students Brad Rubenstein and Antonin Guttman explained how that industry-needed support for “new data types such as polygons, rectangles, text strings, etc.,” “efficient spatial searching,” “complex integrity constraints,” and “design hierarchies and multiple representations” of the same physical constructions [Stonebraker 1983a]. Based on motivations such as these, the group started work on indexing (including Guttman’s influential R-trees for spatial indexing; [Guttman 1984]), and on adding Abstract Data Types (ADTs) to a relational
database system. ADTs were a popular new programming language construct at the time, pioneered by subsequent Turing Award winner Barbara Liskov and explored in database application programming by Stonebraker’s new collaborator, Larry Rowe. In a paper in SIGMOD Record in 1984 [Ong et al. 1984], Stonebraker and students James Ong and Dennis Fogg describe an exploration of this idea as an extension to Ingres called ADT-Ingres, which included many of the representational ideas that were explored more deeply—and with more system support—in Postgres.

**Postgres: An Overview**

As indicated by the name, Postgres was “Post-Ingres”: a system designed to take what Ingres could do and go beyond. The signature theme of Postgres was the introduction of what he eventually called Object-Relational database features: support for object-oriented programming ideas within the data model and declarative query language of a database system. But Stonebraker also decided to pursue a number of other technical challenges in Postgres that were independent of object-oriented support, including active database rules, versioned data, tertiary storage, and parallelism.

Two papers were written on the design of Postgres: an early design in SIGMOD 1986 [Stonebraker and Rowe 1986] and a “mid-flight” design description in CACM 1991 [Stonebraker and Kemnitz 1991]. The Postgres research project ramped down in 1992 with the founding of Stonebraker’s Illustra startup, which involved Stonebraker, key Ph.D. student Wei Hong, and then-chief programmer Jeff Meredith (see Chapter 25). Below, the features mentioned in the 1986 paper are marked with an asterisk (*); those from the 1991 paper that were not in the 1986 paper are marked with a plus sign (+). Other goals listed below were tackled in the system and the research literature, but not in either design paper:

1. Supporting ADTs in a Database System
   (a) Complex Objects (i.e., nested or non-first-normal form data)*
   (b) User-Defined Abstract Data Types and Functions*
   (c) Extensible Access Methods for New Data Types*
   (d) Optimizer Handling of Queries with Expensive User-Defined Functions

2. Active Databases and Rules Systems (Triggers, Alerts)*
   (a) Rules implemented as query rewrites+
   (b) Rules implemented as record-level triggers+
3. Log-centric Storage and Recovery
   (a) Reduced-complexity recovery code by treating the log as data,* using
       non-volatile memory for commit status†
   (b) No-overwrite storage and time travel queries‡
4. Support for querying data on new deep storage technologies, notably optical
   disks*
5. Support for multiprocessors or custom processors*
6. Support for a variety of language models
   (a) Minimal changes to the relational model and support for declarative
       queries*
   (b) Exposure of “fast path” access to internal APIs, bypassing the query
       language†
   (c) Multi-lingual support‡

Many of these topics were addressed in Postgres well before they were studied
or reinvented by others; in many cases, Postgres was too far ahead of its time and
the ideas caught fire later, with a contemporary twist.

We briefly discuss each of these Postgres contributions, and connections to
subsequent work in computing.

Supporting ADTs in a Database System

The signature goal of Postgres was to support new Object-Relational features: the
extension of database technology to support a combination of the benefits of relational
query processing and object-oriented programming. Over time the Object-
Relational ideas pioneered in Postgres have become standard features in most
modern database systems.

A. Complex Objects

It is quite common for data to be represented in the form of nested bundles or
“objects.” A classic example is a purchase order, which has a nested set of products,
quantities, and prices in the order. Relational modeling religion dictated that such
data should be restructured and stored in an unnested format, using multiple
flat entity tables (orders, products) with flat relationship tables (product_in_order)
connecting them. The classic reason for this flattening is that it reduces duplication
of data (a product being described redundantly in many purchase orders), which
in turn avoids complexity or errors in updating all redundant copies. But in some
cases, you want to store the nested representation, because it is natural for the
application (say, a circuit layout engine in a CAD tool), and updates are rare. This data modeling debate is at least as old as the relational model.

A key aspect of Postgres was to “have your cake and eat it too” from a data modeling perspective: Postgres retained tables as its “outermost” data type but allowed columns to have “complex” types including nested tuples or tables. One of its more esoteric implementations, first explored in the ADT-Ingres prototype, was to allow a table-typed column to be specified declaratively as a query definition: “Quel as a data type” [Stonebraker et al. 1984a].

The “post-relational” theme of supporting both declarative queries and nested data has recurred over the years—often as an outcome of arguments about which is better. At the time of Postgres in the 1980s and 1990s, some of the object-oriented database groups picked up the idea and pursued it to a standard language called OQL, which has since fallen from use.

Around the turn of the millennium, declarative queries over nested objects became a research obsession for a segment of the database community in the guise of XML databases; the resulting XQuery language (headed by Don Chamberlin of SQL fame) owes a debt to the complex object support in Postgres’ PostQuel language. XQuery had broad adoption and implementation in industry, but never caught on with users. The ideas are being revisited yet again today in query language designs for the JSON data model popular in browser-based applications. Like OQL, these languages are in many cases an afterthought in groups that originally rejected declarative queries in favor of developer-centric programming (the “NoSQL” movement), only to want to add queries back to the systems post hoc. In the meantime, as Postgres has grown over the years (and shifted syntax from PostQuel to versions of SQL that reflect many of these goals), it has incorporated support for nested data like XML and JSON into a general-purpose DBMS without requiring any significant rearchitecting. The battle swings back and forth, but the Postgres approach of extending the relational framework with extensions for nested data has shown time and again to be a natural end-state for all parties after the arguments subside.

B. User-defined Abstract Data Types and Functions

In addition to offering nested types, Postgres pioneered the idea of having opaque, extensible Abstract Data Types (ADTs), which are stored in the database but not interpreted by the core database system. In principle, this was always part of Codd’s relational model: integers and strings were traditional, but really any atomic data types with predicates can be captured in the relational model. The challenge was to
provide that mathematical flexibility in software. To enable queries that interpret and manipulate these objects, an application programmer needs to be able to register User-Defined Functions (UDFs) for these types with the system and be able to invoke those UDFs in queries. User-Defined Aggregate (UDA) functions are also desirable to summarize collections of these objects in queries. Postgres was the pioneering database system supporting these features in a comprehensive way.

Why put this functionality into the DBMS, rather than the applications above? The classic answer was the significant performance benefit of “pushing code to data,” rather than “pulling data to code.” Postgres showed that this is quite natural within a relational framework: It involved modest changes to a relational metadata catalog, and mechanisms to invoke foreign code, but the query syntax, semantics, and system architecture all worked out simply and elegantly.

Postgres was a bit ahead of its time in exploring this feature. In particular, the security implications of uploading unsafe code to a server were not an active concern in the database research community at the time. This became problematic when the technology started to get noticed in industry. Stonebraker commercialized Postgres in his Illustra startup, which was acquired by Informix in large part for its ability to support extensible “DataBlades” (extension packages) including UDFs. Informix’s Postgres-based technology, combined with its strong parallel database offering, made Informix a significant threat to Oracle. Oracle invested heavily in negative marketing about the risks of Informix’s ability to run “unprotected” user-defined C code. Some trace the demise of Informix to this campaign, although Informix’s financial shenanigans (and subsequent federal indictment of its then-CEO) were certainly more problematic. Now, decades later, all the major database vendors support the execution of user-defined functions in one or more languages, using newer technologies to protect against server crashes or data corruption.

Meanwhile, the Big Data stacks of the 2000s—including the MapReduce phenomenon that gave Stonebraker and DeWitt such heartburn [DeWitt and Stonebraker 2008]—are a re-realization of the Postgres idea of user-defined code hosted in a query framework. MapReduce looks very much like a combination of software engineering ideas from Postgres combined with parallelism ideas from systems like Gamma and Teradata, with some minor innovation around mid-query restart for extreme-scalability workloads. Postgres-based start-ups Greenplum and Aster showed around 2007 that parallelizing Postgres could result in something much higher function and practical than MapReduce for most customers, but the market still wasn’t ready for any of this technology in 2008. By now, in 2018, nearly every
Big Data stack primarily serves a workload of parallel SQL with UDFs—very much like the design Stonebraker and team pioneered in Postgres.

C. Extensible Access Methods for New Data Types

Relational databases evolved around the same time as B-trees in the early 1970s, and B-trees helped fuel Codd’s dream of “physical data independence”: B-tree indexes provide a level of indirection that adaptively reorganizes physical storage without requiring applications to change. The main limitation of B-trees and related structures was that they only support equality lookups and one-dimensional range queries. What if you have two-dimensional range queries of the kind typical in mapping and CAD applications? This problem was au courant at the time of Postgres, and the R-tree developed by Antonin Guttman in Stonebraker’s group was one of the most successful new indexes developed to solve this problem in practice. Still, the invention of an index structure does not solve the end-to-end systems problem of DBMS support for multi-dimensional range queries. Many questions arise. Can you add an access method like R-trees to your DBMS easily? Can you teach your optimizer that said access method will be useful for certain queries? Can you get concurrency and recovery correct?

This was a very ambitious aspect of the Postgres agenda: a software architecture problem affecting most of a database engine, from the optimizer to the storage layer and the logging and recovery system. R-trees became a powerful driver and the main example of the elegant extensibility of Postgres’ access method layer and its integration into the query optimizer. Postgres demonstrated—in an opaque ADT style—how to register an abstractly described access method (the R-tree, in this case), and how a query optimizer could recognize an abstract selection predicate (a range selection in this case) and match it to that abstractly described access method. Questions of concurrency control were less of a focus in the original effort: The lack of a unidimensional ordering on keys made B-tree-style locking inapplicable.¹

PostgreSQL today leverages both the original software architecture of extensible access methods (it has B-tree, GiST, SP-GiST, and Gin indexes) and the extensibility and high concurrency of the Generalized Search Tree (GiST) interface as well.

¹ The Postgres challenge of extensible access methods inspired one of my first research projects at the end of graduate school: the Generalized Search Trees (GiST) [Hellerstein et al. 1995] and subsequent notion of Indexability theory [Hellerstein et al. 2002]. I implemented GiST in Postgres during a postdoc semester, which made it even easier to add new indexing logic in Postgres. Marcel Kornacker’s thesis at Berkeley solved the difficult concurrency and recovery problems raised by extensible indexing in GiST in a templated way [Kornacker et al. 1997].
GiST indexes power the popular PostgreSQL-based PostGIS geographic information system; Gin indexes power PostgreSQL’s internal text indexing support.

**D. Optimizer Handling of Queries with Expensive UDFs**

In traditional query optimization, the challenge was generally to minimize the amount of tuple-flow (and hence I/O) you generate in processing a query. This meant that operators that filter tuples (selections) are good to do early in the query plan, while operators that can generate new tuples (join) should be done later. As a result, query optimizers would “push” selections below joins and order them arbitrarily, focusing instead on cleverly optimizing joins and disk accesses. UDFs changed this: if you have expensive UDFs in your selections, the order of executing UDFs can be critical to optimizing performance. Moreover, if a UDF in a selection is really time consuming, it’s possible that it should happen after joins (i.e., selection “pullup”). Doing this optimally complicated the optimizer space.

I took on this problem as my first challenge in graduate school and it ended up being the subject of both my M.S. with Stonebraker at Berkeley and my Ph.D. at Wisconsin under Jeff Naughton, with ongoing input from Stonebraker. Postgres was the first DBMS to capture the costs and selectivities of UDFs in the database catalog. We approached the optimization problem by coming up with an optimal ordering of selections, and then an optimal interleaving of the selections along the branches of each join tree considered during plan search. This allowed for an optimizer that maintained the textbook dynamic programming architecture of System R, with a small additional sorting cost to get the expensive selections ordered properly.\(^2\)

The expensive function optimization feature was disabled in the PostgreSQL source trees early on, in large part because there weren’t compelling use cases at that time for expensive user-defined functions.\(^3\) The examples we used revolved around image processing and are finally becoming mainstream data processing tasks in 2018. Of course, today in the era of Big Data and machine learning workloads, expensive functions have become quite common, and I expect this problem to return to the fore. Once again, Postgres was well ahead of its time.

---

2. When I started grad school, this was one of three topics that Stonebraker wrote on the board in his office as options for me to think about for a Ph.D. topic. I think the second was function indexing, but I cannot remember the third.

3. Ironically, my code from grad school was fully deleted from the PostgreSQL source tree by a young open-source hacker named Neil Conway, who some years later started a Ph.D. with me at UC Berkeley and is now one of Stonebraker’s Ph.D. grandchildren.
Active Databases and Rule Systems

The Postgres project began at the tail end of the AI community's interest in rule-based programming as a way to represent knowledge in “expert systems.” That line of thinking was not successful; many say it led to the much discussed “AI winter” that persisted through the 1990s.

However, rule programming persisted in the database community in two forms. The first was theoretical work around declarative logic programming using Datalog. This was a bugbear of Stonebraker's; he really seemed to hate the topic and famously criticized it in multiple “community” reports over the years. The second database rules agenda was pragmatic work on what was eventually dubbed Active Databases and Database Triggers, which evolved to be a standard feature of relational databases. Stonebraker characteristically voted with his feet to work on the more pragmatic variant.

Stonebraker’s work on database rules began with Eric Hanson’s Ph.D., which initially targeted Ingres but quickly transitioned to the new Postgres project. It expanded to the Ph.D. work of Spyros Potamianos on PRS2: Postgres Rules System 2. A theme in both implementations was the potential to implement rules in two different ways. One option was to treat rules as query rewrites, reminiscent of the work on rewriting views that Stonebraker pioneered in Ingres. In this scenario, a rule logic of “on condition then action” is recast as “on query then rewrite to a modified query and execute it instead.” For example, a query like “append a new row to Mike’s list of awards” might be rewritten as “raise Mike’s salary by 10%.” The other option was to implement a more physical “on condition then action,” checking conditions at a row level by using locks inside the database. When such locks were encountered, the result was not to wait (as in traditional concurrency control), but to execute the associated action.

4. Datalog survived as a mathematical foundation for declarative languages and has found application over time in multiple areas of computing including software-defined networks and compilers. Datalog is declarative querying “on steroids” as a fully expressive programming model. I was eventually drawn into it as a natural design choice and have pursued it in a variety of applied settings outside of traditional database systems.

5. The code for row-level rules in PRS2 was notoriously tricky. A bit of searching in the Berkeley Postgres archives unearthed the following source code comment—probably from Spyros Potamianos—in Postgres version 3.1, circa 1991:

* DESCRIPTION:
* Take a deeeeeeep breath & read. If you can avoid hacking the code
* below (i.e. if you have not been ‘“volunteered’" by the boss to do this
* dirty job) avoid it at all costs. Try to do something less dangerous
* for your (mental) health. Go home and watch horror movies on-TV.
In the end, neither the query rewriting scheme nor the row-level locking scheme was declared a “winner” for implementing rules in Postgres—both were kept in the released system. Eventually all of the rules code was scrapped and rewritten in PostgreSQL, but the current source still retains both the notions of per-statement and per-row triggers.

The Postgres rules systems were very influential in their day and went “head to head” with research from IBM’s Starburst project and MCC’s HiPac project. Today, “triggers” are part of the SQL standard and implemented in many of the major database engines. They are used somewhat sparingly, however. One problem is that this body of work never overcame the issues that led to AI winter: The interactions within a pile of rules can become untenably confusing as the rule set grows even modestly. In addition, triggers still tend to be relatively time consuming in practice, so database installations that have to run fast tend to avoid the use of triggers. But there has been a cottage industry in related areas like materialized view maintenance, Complex Event Processing, and stream queries, all of which are in some way extensions of ideas explored in the Postgres rules systems.

**Log-centric Storage and Recovery**

Stonebraker described his design for the Postgres storage system this way:

> “When considering the POSTGRES storage system, we were guided by a missionary zeal to do something different. All current commercial systems use a storage manager with a write-ahead log (WAL), and we felt that this technology was well understood. Moreover, the original Ingres prototype from the 1970s used a similar storage manager, and we had no desire to do another implementation.”

[Stonebraker and Kemnitz 1991]

While this is cast as pure intellectual restlessness, there were technical motivations for the work as well. Over the years, Stonebraker repeatedly expressed distaste for the complex write-ahead logging schemes pioneered at IBM and Tandem for database recovery. One of his core objections was based on a software engineering intuition that nobody should rely upon something that complicated—especially for functionality that would only be exercised in rare, critical scenarios after a crash.

---

* Read some Lovecraft. Join the Army. Go and spend a few nights in people's park. Commit suicide ...
* Hm, you keep reading, eh? Oh, well, then you deserve what you-get.
* Welcome to the gloomy labyrinth of the tuple level rule system, my poor hacker...
The Postgres storage engine unified the notion of primary storage and historical logging into a single, simple disk-based representation. At base, the idea was to keep each record in the database in a linked list of versions stamped with transaction IDs—in some sense, this is “the log as data” or “the data as a log,” depending on your point of view. The only additional metadata required is a list of committed transaction IDs and wall-clock times. This approach simplifies recovery enormously since there’s no “translating” from a log representation back to a primary representation. It also enables “time travel” queries: You can run queries “as of” some wall-clock time and access the versions of the data that were committed at that time. The original design of the Postgres storage system—which reads very much as if Stonebraker wrote it in one creative session of brainstorming—contemplated a number of efficiency problems and optimizations to this basic scheme, along with some wet-finger analyses of how performance might play out [Stonebraker 1987]. The resulting implementation in Postgres was somewhat simpler.

Stonebraker’s idea of “radical simplicity” for transactional storage was deeply countercultural at the time when the database vendors were differentiating themselves by investing heavily in the machinery of high-performance transaction processing. Benchmark winners at the time achieved high performance and recoverability via highly optimized, complex write-ahead logging systems. Once they had write-ahead logs working well, the vendors also began to innovate on follow-on ideas such as transactional replication based on log shipping, which would be difficult in the Postgres scheme. In the end, the Postgres storage system never excelled on performance; versioning and time travel were removed from PostgreSQL over time and replaced by write-ahead logging. But the time-travel functionality was interesting and remained unique. Moreover, Stonebraker’s ethos regarding

6. Unfortunately, PostgreSQL still isn’t particularly fast for transaction processing: Its embrace of write-ahead logging is somewhat half-hearted. Oddly, the PostgreSQL team kept much of the storage overhead of Postgres tuples to provide multiversion concurrency control, something that was never a goal of the Berkeley Postgres project. The result is a storage system that can emulate Oracle’s snapshot isolation with a fair bit of extra I/O overhead, but one that does not support Stonebraker’s original idea of time travel or simple recovery.

Mike Olson notes that his original intention was to replace the Postgres B-tree implementation with his own B-tree implementation from the BerkeleyDB project, which developed at Berkeley during the Postgres era. But Olson never found the time. When Berkeley DB got transactional support years later at Sleepycat Corp., Olson tried to persuade the (then-) PostgreSQL community to adopt it for recovery, in place of no-overwrite. They declined; there was a hacker on the project who desperately wanted to build an multi-version currency control system, and as that hacker was willing to do the work, he won the argument.
simple software engineering for recovery has echoes today both in the context of NoSQL systems (which choose replication rather than write-ahead logging) and main-memory databases (which often use multi-versioning and compressed commit logs). The idea of versioned relational databases and time-travel queries are still relegated to esoterica today, popping up in occasional research prototypes and minor open-source projects. It is an idea that is ripe for a comeback in our era of cheap storage and continuously streaming data.

**Queries over New Deep Storage Technologies**

In the middle of the Postgres project, Stonebraker signed on as a co-principal investigator on a large grant for digital earth science called Project Sequoia. Part of the grant proposal was to handle unprecedented volumes of digital satellite imagery requiring up to 100 terabytes of storage, far more data than could be reasonably stored on magnetic disks at the time. The center of the proposed solution was to explore the idea of a DBMS (namely Postgres) facilitating access to near-line “tertiary” storage provided by robotic “jukeboxes” for managing libraries of optical disks or tapes.

A couple different research efforts came out of this. One was the Inversion file system: an effort to provide a UNIX filesystem abstraction above an RDBMS. In an overview paper for Sequoia, Stonebraker described this in his usual cavalier style as “a straightforward exercise” [Stonebraker 1995]. In practice, this kept Stonebraker student (and subsequent Cloudera founder) Mike Olson busy for a couple years, and the final result was not exactly straightforward [Olson 1993], nor did it survive in practice.7

The other main research thrust on this front was the incorporation of tertiary storage into a more typical relational database stack, which was the subject of Sunita Sarawagi’s Ph.D. thesis. The main theme was to change the scale at which you think about managing space (i.e., data in storage and the memory hierarchy) and time (coordinating query and cache scheduling to minimize undesirable I/Os). One of the key issues in that work was to store and retrieve large multidimensional

---

7. Some years after Inversion, Bill Gates tilted against this same windmill with WinFS, an effort to rebuild the most widely used filesystem in the world over a relational database backend. WinFS was delivered in developer releases of Windows but never made it to market. Gates later called this his greatest disappointment at Microsoft.
arrays in tertiary storage—echoing work in multidimensional indexing, the basic ideas included breaking up the array into chunks and storing chunks together that are fetched together—including replicating chunks to enable multiple physical “neighbors” for a given chunk of data. A second issue was to think about how disk becomes a cache for tertiary storage. Finally, query optimization and scheduling had to take into account the long load times of tertiary storage and the importance of “hits” in the disk cache—this affects both the plan chosen by a query optimizer, and the time at which that plan is scheduled for execution.

Tape and optical disk robots are not widely used at present. But the issues of tertiary storage are very prevalent in the cloud, which has deep storage hierarchies in 2018: from attached solid-state disks to reliable disk-like storage services (e.g., AWS EBS) to archival storage (e.g., AWS S3) to deep storage (e.g., AWS Glacier). It is still the case today that these storage tiers are relatively detached, and there is little database support for reasoning about storage across these tiers. I would not be surprised if the issues explored on this front in Postgres are revisited in the near term.

**Support for Multiprocessors: XPRS**

Stonebraker never architected a large parallel database system, but he led many of the motivating discussions in the field. His “Case for Shared Nothing” paper [Stonebraker 1986d] documented the coarse-grained architectural choices in the area; it popularized the terminology used by the industry and threw support behind shared-nothing architectures like those of Gamma and Teradata, which were rediscovered by the Big Data crowd in the 2000s.

Ironically, Stonebraker’s most substantive contribution to the area of parallel databases was a “shared memory” architecture called XPRS, which stood for eXtended Postgres on RAID and Sprite. XPRS was the “Justice League” of Berkeley systems in the early 1990s: a brief combination of Stonebraker’s Postgres system, John Ousterhout’s Sprite distributed OS, and Dave Patterson’s and Randy Katz’s RAID storage architectures. Like many multi-faculty efforts, the execution of XPRS was actually determined by the grad students who worked on it. The primary contributor ended up being Wei Hong, who wrote his Ph.D. thesis on parallel query optimization in XPRS. Hence, the main contribution of XPRS to the literature and industry was parallel query optimization, with no real consideration of issues involving RAID or Sprite.8

8. Of the three projects, Postgres and RAID both had enormous impact. Sprite is best remembered for Mendel Rosenblum’s Ph.D. thesis on Log Structured File Systems (LFS), which had nothing of
In principle, parallelism “blows up” the plan space for a query optimizer by making it multiply the traditional choices made during query optimization (data access, join algorithms, join orders) against all possible ways of parallelizing each choice. The basic idea of what Stonebraker called “The Wei Hong Optimizer” was to cut the problem in two: Run a traditional single-node query optimizer in the style of System R, and then “parallelize” the resulting single-node query plan by scheduling the degree of parallelism and placement of each operator based on data layouts and system configuration. This approach is heuristic, but it makes parallelism an additive cost to traditional query optimization, rather than a multiplicative cost.

Although “The Wei Hong Optimizer” was designed in the context of Postgres, it became the standard approach for many of the parallel query optimizers in industry.

Support for a Variety of Language Models

One of Stonebraker’s recurring interests since the days of Ingres was the programmer API to a database system. In his Readings in Database Systems series, he frequently included work like Carlo Zaniolo’s GEM language as important topics for database system aficionados to understand. This interest in language undoubtedly led him to partner up with Larry Rowe on Postgres, which in turn deeply influenced the design of the Postgres data model and its Object-Relational approach. Their work focused largely on data-centric applications they saw in the commercial realm, including both business processing and emerging applications like CAD/CAM computer-aided design (and manufacturing) and Geographic Information System (GIS).

One issue that was forced upon Stonebraker at the time was the idea of “hiding” the boundary between programming language constructs and database storage. Various competing research projects and companies exploring Object-Oriented Databases (OODBs) were targeting the so-called “impedance mismatch” between imperative object-oriented programming languages like Smalltalk, C++, and Java, and the declarative relational model. The OODB idea was to make programming language objects be optionally marked “persistent,” and handled automatically by...
an embedded DBMS. Postgres supported storing nested objects and ADTs, but its relational-style declarative query interface meant that each round trip to the database was unnatural for the programmer (requiring a shift to declarative queries) and expensive to execute (requiring query parsing and optimization). To compete with the OODB vendors, Postgres exposed a so-called “Fast Path” interface: basically, a C/C++ API to the storage internals of the database. This enabled Postgres to be moderately performant in academic OODB benchmarks, but never really addressed the challenge of allowing programmers in multiple languages to avoid the impedance mismatch problem. Instead, Stonebraker branded the Postgres model as “Object-Relational” and simply sidestepped the OODB workloads as a “zero-billion-dollar” market. Today, essentially all commercial relational database systems are “Object-Relational” database systems.

This proved to be a sensible decision. Today, none of the OODB products exist in their envisioned form, and the idea of “persistent objects” in programming languages has largely been discarded. By contrast, there is widespread usage of object-relational mapping layers (fueled by early efforts like Java Hibernate and Ruby on Rails) that allow declarative databases to be tucked under nearly any imperative object-oriented programming language as a library, in a relatively seamless way. This application-level approach is different than both OODBs and Stonebraker’s definition of Object-Relational DBs. In addition, lightweight persistent key-value stores have succeeded as well, in both non-transactional and transactional forms. These were pioneered by Stonebraker’s Ph.D. student Margo Seltzer, who wrote BerkeleyDB as part of her Ph.D. thesis at the same time as the Postgres group, which presaged the rise of distributed “NoSQL” key-value stores like Dynamo, MongoDB, and Cassandra.

Software Impact

Open Source

Postgres was always an open-source project with steady releases, but in its first many years it was targeted at usage in research, not in production.

As the Postgres research project was winding down, two students in Stonebraker’s group—Andrew Yu and Jolly Chen—modified the system’s parser to accept an extensible variant of SQL rather than the original PostQuel language. The first Postgres release supporting SQL was Postgres95; the next was dubbed PostgreSQL.

A set of open-source developers became interested in PostgreSQL and “adopted” it even as the rest of the Berkeley team was moving on to other interests. Over
time, the core developers for PostgreSQL have remained fairly stable, and the open-source project has matured enormously. Early efforts focused on code stability and user-facing features, but over time the open-source community made significant modifications and improvements to the core of the system as well, from the optimizer to the access methods and the core transaction and storage system. Since the mid-1990s, very few of the PostgreSQL internals came out of the academic group at Berkeley—the last contribution may have been my GiST implementation in the latter half of the 1990s—but even that was rewritten and cleaned up substantially by open-source volunteers (from Russia, in that case). The open source community around PostgreSQL deserves enormous credit for running a disciplined process that has soldiered on over decades to produce a remarkably high-impact and long-running project.

While many things have changed in 25 years, the basic architecture of PostgreSQL remains quite similar to the university releases of Postgres in the early 1990s, and developers familiar with the current PostgreSQL source code would have little trouble wandering through the Postgres3.1 source code (c. 1991). Everything from source code directory structures to process structures to data structures remain remarkably similar. The code from the Berkeley Postgres team had excellent bones.

PostgreSQL today is without question the most high-function open-source DBMS, supporting features that are often missing from commercial products. It is also (according to one influential rankings site) the most popular widely used independent open-source database in the world9 and its impact continues to grow: In 2017 it was the fastest-growing database system in the world in popularity.10 PostgreSQL is used across a wide variety of industries and applications, which is perhaps not surprising given its ambition of broad functionality; the PostgreSQL website catalogs some of the uses at http://www.postgresql.org/about/users/. (Last accessed January 22, 2018.)

Heroku is a cloud SaaS provider that is now part of Salesforce. Postgres was adopted by Heroku in 2010 as the default database for its platform. Heroku chose

---

9. According to DB Engines (http://db-engines.com/en/ranking. Last accessed January 22, 2018), PostgreSQL today is the fourth most popular DBMS in the world, after Oracle, MySQL and MS SQL Server, all of which are corporate offerings (MySQL was acquired by Oracle many years ago). See http://db-engines.com/en/ranking_definition (Last accessed January 22, 2018) for a discussion of the rules for this ranking.

Postgres because of its operational reliability. With Heroku’s support, more major application frameworks such as Ruby on Rails and Python for Django began to recommend Postgres as their default database.

PostgreSQL today supports an extension framework that makes it easy to add additional functionality to the system via UDFs and related modifications. There is now an ecosystem of PostgreSQL extensions—akin to the Illustra vision of Data-Blades, but in open source. Some of the more interesting extensions include the Apache MADlib library for machine learning in SQL, and the Citus library for parallel query execution.

One of the most interesting open-source applications built over Postgres is the PostGIS Geographic Information System, which takes advantage of many of the features in Postgres that originally inspired Stonebraker to start the project.

Commercial Adaptations

PostgreSQL has long been an attractive starting point for building commercial database systems, given its permissive open-source license, its robust codebase, its flexibility, and breadth of functionality. Summing the acquisition prices listed below, Postgres has led to over $2.6 billion in acquisitions. Many of the commercial efforts that built on PostgreSQL have addressed what is probably its key limitation: the ability to scale out to a parallel, shared-nothing architecture.

1. Illustra was Stonebraker’s second major start-up company, founded in 1992, seeking to commercialize Postgres as RTI had commercialized Ingres. The

11. Note that this is a measure in real transaction dollars and is much more substantial than the values often thrown around in high tech. Numbers in the billions are often used to describe estimated value of stock holdings but are often inflated by $10× or more against contemporary value in hopes of future value. The transaction dollars of an acquisition measure the actual market value of the company at the time of acquisition. It is fair to say that Postgres has generated more than $2.6 billion of real commercial value.

12. Parallelizing PostgreSQL requires a fair bit of work, but is eminently doable by a small, experienced team. Today, industry-managed open-source forks of PostgreSQL such as Greenplum and CitusDB offer this functionality. It is a shame that PostgreSQL wasn’t parallelized in a true open-source way much earlier. If PostgreSQL had been extended with shared-nothing features in open source in the early 2000s, it is quite possible that the open-source Big Data movement would have evolved quite differently and more effectively.

13. Illustra was actually the third name proposed for the company. Following the painterly theme established by Ingres, Illustra was originally called Miró. For trademark reasons the name was changed to Montage, but that also ran into trademark problems.
founding team included some of the core Postgres team including recent Ph.D. alumnus Wei Hong and then-chief programmer Jeff Meredith, along with Ingres alumni Paula Hawthorn and Michael Ubell. Postgres M.S. student Mike Olson joined shortly after the founding, and I worked on the Illustra handling of optimizing expensive functions as part of my Ph.D. work. There were three main efforts in Illustra: to extend SQL92 to support user-defined types and functions as in PostQuel, to make the Postgres code base robust enough for commercial use, and to foster the market for extensible database servers via examples of “DataBlades,” domain-specific plug-in components of data types and functions (see Chapter 25). Illustra was acquired by Informix in 1997 for an estimated $400M, and its DataBlade architecture was integrated into a more mature Informix query processing codebase as Informix Universal Server.

2. Netezza was a startup founded in 1999, which forked the PostgreSQL codebase to build a high-performance parallel query processing engine on custom field-programmable-gate-array-based hardware. Netezza was quite successful as an independent company and had its IPO in 2007. It was eventually acquired by IBM, with a value of $1.7B.

3. Greenplum was the first effort to offer a shared-nothing parallel, scale-out version of PostgreSQL. Founded in 2003, Greenplum forked from the public PostgreSQL distribution, but maintained the APIs of PostgreSQL to a large degree, including the APIs for user-defined functions. In addition to parallelization, Greenplum extended PostgreSQL with an alternative high-performance compressed columnar storage engine and a parallelized rule-driven query optimizer called Orca. Greenplum was acquired by EMC in 2010 for an estimated $300M; in 2012, EMC consolidated Greenplum into its subsidiary, Pivotal. In 2015, Pivotal chose to release Greenplum and Orca back into open source. One of the efforts at Greenplum that leveraged its Postgres API was the MADlib library for machine learning in SQL; MADlib runs single-threaded in PostgreSQL and in parallel over Greenplum. MADlib lives on today as an Apache project. Another interesting open-source project based

on Greenplum is Apache HAWQ, a Pivotal design that runs the “top half” of Greenplum (i.e., the parallelized PostgreSQL query processor and extensibility APIs) in a decoupled fashion over Big Data stores such as the Hadoop File System.

4. EnterpriseDB was founded in 2004 as an open-source-based business, selling PostgreSQL in both a vanilla and enhanced edition with related services for enterprise customers. A key feature of the enhanced EnterpriseDB Advanced Server is a set of database compatibility features with Oracle to allow application migration off of Oracle.

5. Aster Data was founded in 2005 by two Stanford students to build a parallel engine for analytics. Its core single-node engine was based on PostgreSQL. Aster focused on queries for graphs and on analytics packages based on UDFs that could be programmed with either SQL or MapReduce interfaces. Aster Data was acquired by Teradata in 2011 for $263M.\(^\text{16}\) While Teradata never integrated Aster into its core parallel database engine, it still maintains Aster as a standalone product for use cases outside the core of Teradata’s warehousing market.

6. ParAccel was founded in 2006, selling a shared-nothing parallel version of PostgreSQL with column-oriented, shared-nothing storage. ParAccel enhanced the Postgres optimizer with new heuristics for queries with many joins. In 2011, Amazon invested in ParAccel, and in 2012 announced AWS Redshift, a hosted data warehouse as a service in the public cloud based on ParAccel technology. In 2013, ParAccel was acquired by Actian (which also had acquired Ingres) for an undisclosed amount—meaning it was not a material expense for Actian. Meanwhile, AWS Redshift has been an enormous success for Amazon—for many years it was the fastest-growing service on AWS, and many believe it is poised to put long-time data warehousing products like Teradata and Oracle Exadata out of business. In this sense, Postgres may achieve its ultimate dominance in the cloud.

7. CitusDB was founded in 2010 to offer a shared-nothing parallel implementation of PostgreSQL. While it started as a fork of PostgreSQL, as of 2016 CitusDB is implemented via public PostgreSQL extension APIs and can be

installed into a vanilla PostgreSQL installation. Also, as of 2016, the CitusDB extensions are available in open source.

Lessons

You can draw a host of lessons from the success of Postgres, a number of them defiant of conventional wisdom.

The highest-order lesson I draw comes from the fact that Postgres defied Fred Brooks’ “Second System Effect.” Brooks argued that designers often follow up on a successful first system with a second system that fails due to being overburdened with features and ideas. Postgres was Stonebraker’s second system, and it was certainly chock full of features and ideas. Yet the system succeeded in prototyping many of the ideas while delivering a software infrastructure that carried a number of the ideas to a successful conclusion. This was not an accident—at base, Postgres was designed for extensibility, and that design was sound. With extensibility as an architectural core, it is possible to be creative and stop worrying so much about discipline: You can try many extensions and let the strong succeed. Done well, the “second system” is not doomed; it benefits from the confidence, pet projects, and ambitions developed during the first system. This is an early architectural lesson from the more “server-oriented” database school of software engineering, which defies conventional wisdom from the “component oriented” operating systems school of software engineering.

Another lesson is that a broad focus—“one size fits many”—can be a winning approach for both research and practice. To coin some names, “MIT Stonebraker” made a lot of noise in the database world in the early 2000s that “one size doesn’t fit all.” Under this banner he launched a flotilla of influential projects and startups, but none took on the scope of Postgres. It seems that “Berkeley Stonebraker” defies the later wisdom of “MIT Stonebraker,” and I have no issue with that. Of course there’s wisdom in the “one size doesn’t fit all” motto (it’s always possible to find modest markets for custom designs!), but the success of “Berkeley Stonebraker’s” signature system—well beyond its original intents—demonstrates that a broad majority of database problems can be solved well with a good general-purpose architecture. Moreover, the design of that architecture is a technical challenge and accomplishment in its own right. In the end—as in most science and engineering debates—there isn’t only one good way to do things. Both Stonebrakers have

17. As Emerson said, “a foolish consistency is the hobgoblin of little minds.”
lessons to teach us. But at the base, I’m still a fan of the broader agenda that “Berkeley Stonebraker” embraced.

A final lesson I take from Postgres is the unpredictable potential that can come from open-sourcing your research. In his Turing talk, Stonebraker speaks about the “serendipity” of PostgreSQL succeeding in open source, largely via people outside Stonebraker’s own sphere. It’s a wonderfully modest quote:

[A] pick-up team of volunteers, none of whom have anything to do with me or Berkeley, have been shepherding that open-source system ever since 1995. The system that you get off the web for Postgres comes from this pick-up team. It is open source at its best and I want to just mention that I have nothing to do with that and that collection of folks we all owe a huge debt of gratitude to.18

I’m sure all of us who have written open source would love for that kind of “serendipity” to come our way. But it’s not all serendipity—the roots of that good luck were undoubtedly in the ambition, breadth, and vision that Stonebraker had for the project, and the team he mentored to build the Postgres prototype. If there’s a lesson there, it might be to “do something important and set it free.” It seems to me (to use a Stonebrakerism) that you can’t skip either part of that lesson.

Acknowledgments
I’m indebted to my old Postgres buddies Wei Hong, Jeff Meredith, and Mike Olson for their remembrances and input, and to Craig Kerstiens for his input on modern-day PostgreSQL.

The Collected Works of Michael Stonebraker


N. Malviya, A. Weisberg, S. Madden, and M. Stonebraker. 2014. Rethinking main memory OLTP recovery. In *Proc. 30th International Conference on Data Engineering*, pp. 604–615. DOI: [10.1109/ICDE.2014.6816685](https://doi.org/10.1109/ICDE.2014.6816685).


M. Stonebraker and D. Moore. 1996. *Object-Relational DBMSs: The Next Great Wave*. Morgan Kaufmann. 111


M. Stonebraker, E. N. Hanson, and C. Hong. 1987c. The design of the postgres rules system. In *Proc. 3th International Conference on Data Engineering*, pp. 365–374. DOI: [10.1109/ICDE.1987.7272402](https://doi.org/10.1109/ICDE.1987.7272402). 91


M. Stonebraker. 1992a. The integration of rule systems and database systems. *IEEE Transactions on Knowledge and Data Engineering*, 4(5): 415–423. DOI: 10.1109/69.166984. 91


References


References 637


IBM. 1997. Special Issue on IBM’s S/390 Parallel Sysplex Cluster. IBM Systems Journal, 36(2). 400


References


N. Tatbul, U. Çetintemel, and S. Zdonik. 2007. “Staying FIT: Efficient Load Shedding Techniques for Distributed Stream Processing.” International Conference on Very Large Data Bases (VLDB’07), Vienna, Austria. 228, 229


**Index**

Page numbers with ‘f’ indicate figures; page numbers with ‘n’ indicate footnotes.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>50-year perspective of Stonebraker</td>
<td>107–108</td>
</tr>
<tr>
<td>1970 fall, University of Michigan</td>
<td>107–108</td>
</tr>
<tr>
<td>1976 fall, Wisconsin</td>
<td>108–111</td>
</tr>
<tr>
<td>1983 fall, Berkeley</td>
<td>111</td>
</tr>
<tr>
<td>1988–1995</td>
<td>111</td>
</tr>
<tr>
<td>2000, Project Sequoia</td>
<td>112–113</td>
</tr>
<tr>
<td>2003, CIDR Conference launch</td>
<td>113</td>
</tr>
<tr>
<td>2005, MIT sabbatical</td>
<td>113–114</td>
</tr>
<tr>
<td>2008, MapReduce</td>
<td>114</td>
</tr>
<tr>
<td>2014, Turing Award</td>
<td>114</td>
</tr>
<tr>
<td>2016, MIT</td>
<td>115</td>
</tr>
<tr>
<td>2017, encounter</td>
<td>115</td>
</tr>
<tr>
<td>1000 Genomes Browser</td>
<td>267</td>
</tr>
<tr>
<td>1 Million Veterans program</td>
<td>354</td>
</tr>
<tr>
<td>Abadi, Daniel J., C-Store project perspective article</td>
<td>235–244</td>
</tr>
<tr>
<td>end of architectural era</td>
<td>463–489</td>
</tr>
<tr>
<td>H-Store prototype</td>
<td>247, 249</td>
</tr>
<tr>
<td>OLTP databases</td>
<td>411–439</td>
</tr>
<tr>
<td>Vertica Systems</td>
<td>334</td>
</tr>
<tr>
<td>Abstract data types (ADTs)</td>
<td></td>
</tr>
<tr>
<td>Ingres</td>
<td>88, 148</td>
</tr>
<tr>
<td>Ingres prototype</td>
<td>202, 203f</td>
</tr>
<tr>
<td>Postgres</td>
<td>88, 149, 206–212, 316, 523</td>
</tr>
<tr>
<td>“Access Control in a Relational Database Management System By Query Modification” (Stonebraker and Wong)</td>
<td>45</td>
</tr>
<tr>
<td>Access methods</td>
<td></td>
</tr>
<tr>
<td>Ingres</td>
<td>575–585, 577f, 582f</td>
</tr>
<tr>
<td>Postgres</td>
<td>534–535</td>
</tr>
<tr>
<td>Access Methods Interface (AMI)</td>
<td>582–585</td>
</tr>
<tr>
<td>ACM Software System Award</td>
<td>44</td>
</tr>
<tr>
<td>Actian enterprise</td>
<td>310</td>
</tr>
<tr>
<td>Active databases in Postgres</td>
<td>212–213</td>
</tr>
<tr>
<td>Active-passive replication in OLTP</td>
<td>434–435</td>
</tr>
<tr>
<td>Addmark system</td>
<td>494</td>
</tr>
<tr>
<td>Address space limitations in Ingres</td>
<td>574–575</td>
</tr>
<tr>
<td>Administration files in Ingres</td>
<td>576</td>
</tr>
<tr>
<td>ADMINS system</td>
<td>404</td>
</tr>
<tr>
<td>ADT-Ingres</td>
<td>206</td>
</tr>
<tr>
<td>Adult supervision for startup companies</td>
<td>122–123</td>
</tr>
<tr>
<td>Advanced H-Store strategy</td>
<td>478</td>
</tr>
<tr>
<td>Affero GPL license</td>
<td>262</td>
</tr>
<tr>
<td>Aggregation operators in C-Store</td>
<td>509</td>
</tr>
<tr>
<td>Aggregation systems in one size fits all</td>
<td>451–452</td>
</tr>
<tr>
<td>AI systems</td>
<td></td>
</tr>
<tr>
<td>Ingres-oriented</td>
<td>202</td>
</tr>
<tr>
<td>machine learning</td>
<td>66</td>
</tr>
</tbody>
</table>
Algorithmic complexity in Tamr, 359–361
Allman, Eric, 99, 111, 195
Allocator fragmentation in VoltDB, 342–343
AllofUs program, 354
Anchor tuples in H-Store, 474
Anchored projections in C-Store, 497
Anton, Jeff
  Miro team, 314f
  Postgres productionization, 315
AntsDBMS, 478
Anurag, Maskey, on Aurora project, 324f
Aoki, Paul, 312–313n
Apache Hadoop project, 4–5, 171
  criticism of, 69, 184
  open source impact, 171
Apache HAWQ, 222
Apache Spark, 402
Application logic in one size fits all, 452–454, 458
Area greater than (AGT) operator in Postgres, 525
ARIES (Algorithms for Recovery and isolation Exploiting Semantics), 346
Arizona State University, 149
Array Functional Language (AFL) for SciDB, 267, 353–354
Arrays
  Postgres, 537
  SciDB, 260–262, 261f
AS/400 platform, 380
Aster Data
  founding, 222
  Postgres parallelization, 209
Audio data management, 265
Aurora codelines and stream processing systems, 321–326, 324f
Aurora project
  founding, 46
  origins, 225–227
  research story, 387–391
  StreamBase based on, 89
  systems, 227–231
Aurum system
  Data Civilizer, 189
  description, 299
  AUX directory in Ingres, 576
Availability
  OLTP design, 468–470
  one size fits all, 454–455
AVL trees, 436
AWS Redshift, 222
B-trees and B-tree indexes, 90
C-Store, 503
  commercial Ingres codeline, 307
  description, 493
  OLTP, 429, 435–436
  and Postgres, 210
“B-Trees Re-examined” (Stonebraker and Held), 196–197
Bachman, Charles
  relational-CODASYL debate, 404, 406
  Turing Award, 3, 48, 87
Bailis, Peter, 159
Balakrishnan, Hari
  C-Store project, 238
  StreamBase, 232, 329
Balazinska, Magdalena
  Aurora/Borealis/StreamBase reunion, 332f
  Borealis project, 186
  stream processing era article, 225–234
Bates-Haus, Nikolaus, Tamr codeline article, 357–366, 365f
Batkin, Adam, C-Store seminal work, 491–518
Battle, Leilani, 159
Bear, Chuck, 334–336
Beaumont, Chris, 353
Begoli, Edmon, 370
Berkeley Municipal Court system, 148
Berkeley Software Distribution (BSD) license and Ingres, 166–167
  origins, 165, 398
Berkeley years
  1983 fall, 111
technical contributions, 185–186
BerkeleyDB in StreamBase, 453
Berman, Richard, 99
Bernstein, Philip A.
  on leadership and advocacy, 87–92
  relational database Ph.D, 403
Berube, Paul, 374
Beskales, George
  Data Tamer project, 269–270, 273
  Tamr co-founder, 360, 365f
  Tamr company, 120, 274
Betts, Ryan, 249
BFGoodrich company, 97–98
Bicycle story, 15–16
  Anacortes, 16, 17f
  Battle Lake, 26–27
  Carrington, 26
  difficult part, 79
  Drake, 22–24, 23f
  Ellicottville, 28
  Luddington, 28
  Marias Pass, 21, 21f
  as metaphor for building system software, 32–35
  Moultonborough, 35
  Sutton, 31–32
  Troy, 30
  Winthrop, 18
  Wollaston Beach, 31–32, 31f
Big Data era
  characteristics, 5–6
  and Postgres, 209
  stream processing in, 233–234
  volume, velocity, and variety, 357
BigDAWG codeline
  future, 376
  introduction, 367–370
  milestones, 369, 369f
  origins, 370–371, 371f
  public demonstration, 371–373, 372f
  refining, 373–374, 374f
  release, 375–376
BigDAWG polystore system
  conclusion, 288–289
  data movement, 286–287
  description, 189
  development, 284–285
  ISTC, 279–280
  one size does not fit all, 282–284
  origins, 280–282, 281–282f
  perspective on, 63–64
  query modeling and optimization, 285–286
  releases and demos, 287–288, 288f
Biller, Steve, 374
BIN directory in Ingres, 576
Biobank program, 354
Bioinformatics market, 263–267
Bitstrings in C-Store, 509
Blob storage in VoltDB, 343
Blocking, 359
Bochkov, Dmitry, 337
Bohrbugs, 34
Borealis codelines for stream processing systems, 321–326
Borealis project
  Aurora project move to, 325
  origins, 225–227
  systems, 227–231, 230f
Bottleneck studies, 436
Bowes, Ari, on Miro team, 314f
Boyce, Bruce, 406
Brooks, Fred, 223
Brown, Paul
  quad chart, 111
  SciDB codeline, 352
  scientific data management article, 253–268
Bruckner, Daniel
  Data Tamer project, 269–270, 273
  Tamr company, 274, 365f
Buffer management
  OLTP, 414
  Shore, 421, 423
Buffer Pool Manager in Shore, 419
Bulk copy of data in QUEL, 566
Business acumen on startup companies,
  teams, 122–123
Index

Business-to-consumer (B2C) space, 150
Butterworth, Paul, commercial Ingres codeline article, 303–310
C language in Postgres codelines, 311–312
C-Store project, 104
column-oriented database, 132
COMMIT statements, 506
concat operators, 509
covering sets of projections, 498
deletes, 503
launch, 46–47
one size doesn’t fit all era, 187
primary keys, 496
prototype, 121
Vertica Systems based on, 89
C-Store project perspective
Abadi and computer science, 235–238
building, 240–242, 241f
idea, evolution, and impact, 238–240
Vertica Systems founding, 242–244
C-Store seminal work
abstract, 491–492
conclusion, 516–517
data model, 496–500, 499f
introduction, 492–496, 495f
performance, 511–515
query execution, 509–510
query optimization, 510
related work, 515–516
RS column store, 500–501
storage management, 502–503
tuple movers, 508
updates and transactions, 503–508, 505f
WS column store, 502, 507–508
Cache-conscious B-trees in OLTP, 435–436
Caches in rules systems, 546
Cafarella, Michael, 6
Caltech Mead-Conway VLSI design era, 201
Career flowchart, 54–55
Carey, Michael J., 56f
data storage capabilities, 308
Ingres later years article, 193–204
Ingres project contributions, 185–186
Carnes, Donna, on Miro team, 314f
Carney, Don
Aurora project, 324f
StreamBase Systems, 232
Carter, Fred, commercial Ingres codeline article, 303–310
CASSM project, 109
Catalog relations in Ingres, 576
Çetintemel, Ugur
Aurora/Borealis/StreamBase reunion, 332f
Aurora project, 323, 324f
one size does not fit all, 282–284
one size fits all seminal work, 441–462
StreamBase Systems, 232
Chamberlin, Don
IBM Database, 382, 384
relational-CODASYL debate, 406
XQuery language, 208
Chen, Jolly
Postgres conversion, 149
Postgres parser, 218
PostgreSQL, 382, 384
SQLization project, 317–318
Chen, Peinan, 373
Cherniack, Mitch, 103
Aurora/Borealis/StreamBase reunion, 332f
Aurora project, 235, 323, 324f
C-Store seminal work, 491–518
expert sourcing, 273
StreamBase Systems, 232
Tamr project, 120
Vertica Systems, 334
Chicken Test in Postgres productionization, 315
Chisholm, Sally (Penny), 373–374
Chisholm Laboratory data for BigDAWG polystore system, 287, 374
Christiansen, Clayton, 27, 61, 100
CitusDB, 222–223
Classes, transaction, 471, 480–481
Climate change and Project Sequoia, 112–113
CLOS (Common LISP Object System), 522, 530–531
CLOSER function in Ingres, 585
Cloud Spanner, 70–71
Cloudera
  and MapReduce, 69–70
  open source impact, 171
Cluster computing in OLTP, 416
CODASYL (Conference on Data Systems Languages)
  Codd report, 43
database standard proposal, 87, 97–98
Codd, Edgar (Ted)
  Ingres development, 100
  Ingres inspired by, 43–44, 148
  Ingres platform, 101
  matrix algebra, 258
  relational-CODASYL debate, 404–406
  scientific data management, 263
  SCM SIGFIDET conference, 403
  Stonebraker influenced by, 42–44
  Turing Award, 3, 48
Cohera Corporation, 89, 401
Collected works of Stonebraker, 607–633
Column-oriented database, 132
Column store architecture, 239–240, 242, 492–493
Comeau, Al, 384
Commercial Ingres codeline, 303–304
  conclusion, 309
  open source Ingres, 309–310
  product production, 305–306
  research to commercial efforts, 304–305
  storage structures, 306–308
  user-defined types, 308–309
Commercialization
  C-Store project, 242–244
  impact for, 237–238
  Postgres, 220–223
Commuting members, 473
Compaction in VoltDB, 342–344
Companies founded by Stonebraker, 48–49
Company control in startup company guidelines, 127
Compatibility problem in one size fits all, 442
Complex objects
  Ingres, 202–203
  Postgres, 207–208
Complexity
  avoiding, 236–237
  rules systems, 541–543
  Vertica Systems, 338–339
Compression methods in C-Store project, 240–242, 241f
Computer Information and Control Engineering (CICE) program, 43
Computer science degrees need for, 81–82
  University of Michigan, 107–108
Computer Systems Research Group (CSRG), 165–167
Concepts limitation in Postgres, 522
Concurrency control
  C-Store, 505–506
  Distributed Ingres, 200
  H-Store, 478
  Ingres, 588–591
  OLTP, 433
  Postgres, 214n
CondorDB, 245
Conference on Innovative Data Systems Research (CIDR)
  BigDAWG polystore system, 288
  creation, 50, 92
  launch, 113
  success of, 76
Connection points in Aurora, 229
Consistency
  OLTP, 435
  QUEL, 567
Constrained tree applications (CTAs), 472
Constrained tree schema in H-Store project, 247
Constructed types in Postgres, 523
Control flow in Ingres, 575
Convey, Christian, on Aurora project, 324f
Copyright for Ingres project, 109–110, 166
CORBA, 112
Correct primitives in one size fits all, 451–452
Cost problem in one size fits all, 442
CREATE command in Ingres, 600
Credit, assigning, 394–395
Cueball (partner), 36
Current epochs in C-Store, 504
Customers
forgotten, 158–159
startup companies, 122, 126
StreamBase, 327–328
Cycles in OLTP, 432

Data blades in Illustra, 88
Data Civilizer, 291–292
conclusion, 300
data cleaning challenge, 296–298
data discovery challenge, 298–300
data transformation challenge, 295–296
description, 189
design, 294–295
life of analysts, 292–293
mung work automation, 64
need for, 292
purpose, 293–294
Data cleaning challenge in Data Civilizer, 296–298
Data discovery
Aurora project, 388
Data Civilizer, 298–300
Data-driven discovery paradigm, 2
Data ingest rates DBMS limitation, 226
Data Language/ALPHA, 570
Data Management Technology Kairometer, 3, 8
Data movement in BigDAWG polystore system, 286–287
Data structures in Ingres, 575–585, 577f, 582f

Data Tamer project, 269–270
creation, 105
customers, 65
description, 47
idea and prototype, 270–273, 271f
ideas source, 152
lessons, 276–277
research contributions, 188
startup, 90
Tamr company, 273–275
Data transformation challenge in Data Civilizer, 295–296
Data unification at scale. See Data Tamer project
Data warehouses
ideas source, 151
multiple data copies, 494
one size fits all, 443–445, 444–445f, 455–456
rise of, 401
schemas, 484
Database designer in H-Store, 475
Database Management Systems (DBMSs)
description, 3
Databases, brief history of, 1–6
DataBlades, 221, 317
DataBlitz system, 436
DATADIR directory in Ingres, 576
Datalog rule systems, 212
Date, Chris
referential integrity paper, 18
SCM SIGFIDET conference, 403
David, Martin, 197
DB2/400 system, 380
Db2 for i system, 380
Db2 for VSE&VM, 380
Db2 for z/OS system, 380
DB2/MVS system, 380
DBA relations in Ingres, 576–577
DBC project, 109
Deadlocks in Ingres, 590
Declarative query language, Ingres as, 196
DECOMP program, 591–597
Decompress operators in C-Store, 509
Deep storage technologies in Postgres, 215–216
Deferred update and recovery in Ingres, 600–602
DELETE function in Ingres, 584
Deleted record vectors (DRVs) in C-Store, 504
Densepack values in storage, 493, 501
Depth of transaction classes, 475
“Design and Implementation of Ingres” (Stonebraker), 47
Design problems in Postgres, 549–550
DeWitt, David J.
  50-year perspective article, 107–115
  CIDR, 92
  CondorDB version, 245
  Gamma, 151, 400
  H-Store project, 246
  Hadoop criticism, 184
  one size doesn’t fit all era, 188
  publications, 76
  Shore prototype, 152
  Vertica Systems, 136, 336
DIRECT project, 109–111
Disk orientation DBMS limitation, 226
Disk persistence in VoltDB, 346–347
Distinct values in C-Store, 500–501
Distributed COMMIT processing in C-Store, 506
Distributed computing, 400
Distributed databases, 150
Distributed Ingres, 198–200, 199f, 305
Docker tool for BigDAWG, 375
Document data management in Ingres, 201
Dozier, Jeff, 112
Du, Jiang, 376
Dynamic loading in Postgres, 551–552
Dynamic locking in H-Store, 477
Dziedzic, Adam, 373

Early years and education, 42–43
Elephants, 4, 94

Ellicott, Andy, 134
Ellison, Larry
  Oracle claims, 149
  SQL language support, 196
Elmore, Aaron J., BigDAWG polystore system article, 279–289
Emberson, Richard, on Miro team, 314f
EMP1 (friend), 27
Encoding schemes in C-Store, 500–501
“End of an Architectural Era: It’s Time for a Complete Rewrite” paper (Stonebraker), 247
End of architectural era seminal work abstract, 463–464
H-Store, 473–479, 479f
  introduction, 464–466
  OLTP design considerations, 466–470
  one size does not fit all comments, 483–486
  performance, 479–483, 479f
  summary and future work, 486–487
  transaction, processing and environment assumptions, 470–473
End of epochs in C-Store, 504
End-to-end system Data Civilizer design, 294–295
EnterpriseDB, 222
Entrepreneur-Turned-Shark (friend), 28
Epochs in C-Store, 504–505, 505f
Epstein, Bob
  BSD license, 166
  Distributed Ingres, 198
  Ingres source code, 110–111
  stored procedures, 91
  venture capitalist contact, 140
EQUEL language for Ingres
  comments, 570–571
  invocation from, 573–574
  overview, 568–570
Erwin, Christina, on Aurora project, 324f
ETL toolkit, 358
Exceptions in rules systems, 539
Excessive formalism, 162
Expanding fields, failure to cope with, 155–158, 156f
Expansive execution in BigDAWG polystore system, 285
Experimental results for OLTP, 428–432
Expert database systems, Ingres-oriented, 202
Expert sourcing, 273
Explicit parallelism in Tamr, 363
Extremely Large Data Bases (XLDB) conference and workshop, 256–257
Fabry, Bob, 165
Factoring in one size fits all, 458–459, 459f
Failover OLTP design, 469
one size fits all, 454–455
Failures consequences, 160–163
expanding fields, 155–158, 156f
forgotten customers, 158–159
paper deluge, 159–160
summary, 164
Fast path feature in Postgres, 218, 316, 530–531
Federation architecture, 401
Female Ph.D.s graduated, 394
Fernandez, Raul Castro
Aurum research story, 387–391
Data Civilizer, 189
Data Civilizer article, 291–300
Festschrift, 143–144
Files
Ingres, 576–578, 577f
UNIX environment, 571
Financial-feed processing in one size fits all, 446–447, 448f
FIND function in Ingres, 583–584
First customers for startup companies, 126
First International Conference on Expert Database Systems, 407–408
Fogg, Dennis, 206
Ford, Jim, 99
Foreign keys in C-Store, 496
Foreign-order values in C-Store, 501
Forgotten customers, 158–159
Fork-lift upgrades in OLTP design, 468
Fournier, Marc, 318
Franklin, Michael J., 56f
papers rejected by, 113
Telegraph Team, 325
Frew, Jim, 112
Freitag, Christoph, 173
Functional ideas, 184
Functions
Postgres, 208–209, 211, 523–527
POSTQUEL, 535–536, 555–556
Gadepally, Vijay
BigDAWG codeline article, 367–376
BigDAWG releases, 287
Galvez, Eddie
Aurora project, 324f
StreamBase Systems, 232, 330
Gamma project, 111, 151, 400
Garlic project, 401
Gates, Bill, 215n
GEM language, 217
Generalized Search Tree (GiST) interface, 210–211
Genomic data for SciDB, 354–355
Geographic Information Systems (GIS), 148
GET function in Ingres, 583
Gigascope project, 231
Global Biobank Engine, 267
Go, Angela, 148
Goby company
B2C space, 150
Data Tamer, 152
startup, 90
Google technologies, 70–71
Goss, Jeff, 385f
Gosselin, Dave, 352–353
Governor's Academy, 43
GPUs, 67
Graduate students, 82
Graphical user interfaces (GUIs) in prototypes, 121
Grassy Brook company
- founding, 140–141
- quad chart, 230–232, 231f

Gray, Jim, 36
- 2002 SIGMOD conference, 113
- CIDR, 92
- Project Sequoia, 112
- scientific data management, 255, 259
- System R project, 100
- Tandem Computers, 101
- Turing Award, 3, 48, 114

Great Relational-CODASYL Debate, 403–406

Greenplum startup, 209, 221–222

Grid computing in OLTP design, 468

Gupta, Ankush, 373

Gutman, Antonin
- Ingres CAD management features, 201–202
- R-Tree index structure, 203, 210

H-Store, 245–246
- basic strategy, 478
- buddies, 475
- conclusion, 251
- database designer, 475
- description, 47
- execution supervisor, 475
- founding, 104
- general transactions, 247
- ideas source, 151–152
- one size doesn’t fit all era, 187
- performance, 479–483, 479f
- prototypes, 247–250
- query execution, 474–475
- system architecture, 246–247, 473–474
- transaction classes, 471–473
- transaction management, replication and recovery, 476–478

VoltDB and PayPal, 250

VoltDB based on, 89

VoltDB executor in, 75

VoltDB split, 251

Hachem, Nabil
- Data Civilizer, 295

end of architectural era seminal work, 463–489

Haderle, Don, recollections article, 397–402

Hadoop, 4–5
- criticism of, 69, 184
- open source impact, 171

Hadoop Distributed File System (HDFS ), 70

Hagen, Dale, 384, 385f

Hamilton, James
- on 2014 ACM Turing Award, 93–95
- IBM relational database code bases article, 379–385
- on server costs, 67–68

Hammer, Joachim, 90

Hanson, Eric, 212

Harizopoulos, Stavros
- end of architectural era seminal work, 463–489

H-Store prototype, 247

OLTP databases, 411–439

Harris, Herschel, 385f

HASH structure in commercial Ingres codeline, 306–307

Hatoun, Matt, on Aurora project, 324f

Hawthorn, Paula
- Illustra, 221
- Miro team, 314f
- Postgres productionization, 315

Hearst, Marti, student perspective article, 393–396

Heath, Bobbi
- H-Store prototype, 249
- StreamBase Systems, 232

Hedges, Richard, 384

Heisenbugs, 34

Held, Gerald
- “B-Trees Re-examined,” 194–197
- Ingres implementation seminal work, 561–605
- relational database industry birth, 97–106

Helland, Pat, end of architectural era seminal work, 463–489

Hellerstein, Joseph M.
- Data Tamer, 152
Hellerstein, Joseph M. (continued)
Postgres codelines, 313
Postgres description, 186
Postgres perspective article, 205–224
Postgres project, 102
Tamr project, 120
Telegraph Team, 325
Heroku provider, 219–220
High availability
OLTP, 417, 468–470
one size fits all, 454–455
High water mark (HWM) in C-Store, 504–505, 505f
Hill, Faith, 294
Hints in Postgres, 525
HiPac project, 213
Hirohama, Michael, Postgres implementation seminal work, 519–559
Historical mode queries, 495
Hive executor, 114
Hobbib, Bill, 232, 326
Hong, Wei
Illustra, 206, 221
Miro team, 314f
Postgres and Illustra codelines article, 311–319
Postgres conversion, 149
XPRS architecture, 216–217
Horizontica version, 249, 331
Hot standby in OLTP design, 469
Howe, Bill, 370
HTCondor project, 245
Hugg, John
H-Store prototype, 249–250
VoltDB codeline article, 341–348
Huras, Matt, 383–384, 385f
Hwang, Jeong-Hyon, on Aurora project, 324f
Hypothetical relations in Ingres, 201

IBM
IMS database, 88
SQL language, 102
IBM relational database code bases
four code bases, 379–381
future, 384–385
portable code base, 381–384
IEEE International Conference on Data Engineering 2015 talk, 35
Illustra codelines, 311
overview, 313–317
Postgres and SQL, 315–316
Postgres productionization, 315
Illustra Information Technologies, Inc.
open source impact, 171
Oracle competitor, 102–103
Postgres, 112
Postgres commercial adaptations, 220–221
startup, 206
Ilyas, Ihab
Data Tamer project article, 269–277
Tamr co-founder, 120, 188, 360, 365f
Implementation efficiency in rules systems, 544
“Implementation of Integrity Constraints and Views By Query Modification” (Stonebraker), 45
“Implementation of Postgres” (Stonebraker), 47
Implementing rules, 91
IMS database, 88
In-QTel, 233
Inbound vs. outbound processing in one size fits all, 449–450f, 449–451
INDEX catalog in Ingres, 580
Indexes
C-Store, 498, 499f, 501, 509
commercial Ingres, 306–307
Postgres, 525, 550
primary and secondary, 493
VoltDB, 343
Industrial Liaison Program (ILP), 121
Industry involvement, 46
InfiniBand, 67
Informix
Illustra integrated into, 35
Illustra purchase, 102–103, 112
startups bought by, 66
Informix Universal Server, 317
Ingres implementation seminal work, 561–562
Access Methods Interface, 582–585
conclusion, 602–603
concurrency control, 588–591
data structures and access methods, 575–585, 577f, 582f
deferred update and recovery, 600–602
EQUEL, 568–571
file structure, 576–578, 577f
future extensions, 603
introduction, 562–563
invocation from EQUEL, 573–574
invocation from UNIX, 572–573
performance, 602
Process 2, 585–591
Process 3, 591–599
Process 4, 599–602
process structure, 571–575, 572f, 574f
QUEL and utility commands, 563–568
query modification, 585–588
storage structures, 580–582, 582f
system catalogs, 578–580
UNIX environment, 571–572
user feedback, 602–603
Ingres later years, 193–194
contributions, 195–198
Distributed Ingres, 198–200, 199f
relational DBMS, 194–198, 197f
support domains, 201–204
Ingres project
ATTRIBUTE catalog, 579–580
Berkeley years, 185–186
birth of, 43–45
and BSD, 166–167
commercial codeline. See Commercial Ingres codeline
competition, 100–103
COPY command, 600
copyright, 109–110
decomposition of queries, 591–597
distributed, 150
ideas source, 147–148
impact, 3
leadership and advocacy, 87–90
open source, 167–168, 309–310
platform, 101
Postgres design helped by, 27–28
process structure, 110–111
target platform, 108
team, 99–100
timing, 98–99
Wisconsin, fall 1976, 108–111
Ingres Star, 305
Inheritance in Postgres, 524, 529
Innovators Dilemma (Christiansen), 27, 61, 100
INSERT function in Ingres, 584
Insertion vectors (IVs) in C-Store, 504
Inserts in C-Store, 503
Instructions vs. cycles in OLTP, 432
INTEGRITY catalog in Ingres, 580
Integrity control in QUEL, 567
Intel Science and Technology Center (ISTC)
for Big Data, 279, 367
Intellectual property, 126
Inter-snapshot log in VoltDB, 346
Intermediate H-Store strategy, 478
Inversion file system, 215
Irrelevant theories, 161–162
ISAM (indexed sequential access method), 306–307
Islands in BigDAWG polystore system, 284
Join indexes in C-Store, 498, 499f, 501, 509
Join operators in C-Store, 509
Jones, Anita, 193
Jones, Evan, 248–249
Joy, Bill, 109, 165
JSON data model, 208
K-safe systems, 494, 499–500
Katz, Randy, 216
KDB system, 494
Kelley, Gary, 151
Kepner, Jeremy, 370
Kerschberg, Larry, 407
Kersten, Martin, 75, 151
Keyed storage structure in Ingres, 581–582, 582f

Keys
C-Store, 496–498
Postgres, 550
Kimura, Hideaki, 247–249
Kinchen, Jason, SciDB codeline article, 349–355
KISS adage (Keep it Simple, Stupid) adage, 153
Knowledge management in Postgres, 524
Knudsen, Eliot, 360
Kooi, Bob, 306
Kraska, Tim, 67
Kreps, Peter
Ingres implementation seminal work, 561–605
Ingres team, 99

Land Sharks, 15–16, 29
Langer, Robert, 137
Language constructs as DBMS limitation, 226
Language support in Postgres, 217–218
Large objects in Postgres, 537
Large Synoptic Survey Telescope, 256

Locking
H-Store, 483
OLTP, 414
Shore, 419–420, 423
Latency in VoltDB, 344–348
Lau, Edmond, C-Store seminal work, 491–518
Lawande, Shilpa
Vertica Systems, 134, 136
Vertica Systems codeline article, 333–340
Leadership, partnership approach to, 131–132
Leadership and advocacy, 87
advocacy, 91–92
mechanisms, 90–92
systems, 87–90

Least Publishable Units (LPUs)
grain optimization, 389
problems from, 76, 159, 161
Leibensperger, Mike, 354
Lew, Ivan, 385f
Lewis, Bryan, 350, 353
licenses, BSD, 165, 398
Lighthouse customers for startup companies, 122
Lightstone, Sam, 385f
Lindsay, Bruce, 382, 382–384
Lineage support in scientific data management, 266
Linear Road benchmark, 326, 446
Liquidation preference in startup company guidelines, 124–125
Liskov, Barbara, 88, 206
Lisman, John, 235
LISP for Postgres, 311–312, 553–554
Liu, Jason, at Tamr, 365f
Lock Manager in Shore, 419

Lock Manager in Shore, 419–421, 423
redo, 470
undo, 471
VoltDB, 346

Lohman, Guy, 382, 384
Lorraine, Raymond, 113

Least Publishable Units (LPUs)
grain optimization, 389
problems from, 76, 159, 161

Lightstone, Sam, 385f
Lindsay, Bruce, 382, 382–384
Lineage support in scientific data management, 266
Linear Road benchmark, 326, 446
Liquidation preference in startup company guidelines, 124–125
Liskov, Barbara, 88, 206
Lisman, John, 235
LISP for Postgres, 311–312, 553–554
Liu, Jason, at Tamr, 365f
Lock Manager in Shore, 419

Lock Manager in Shore, 419–421, 423
redo, 470
undo, 471
VoltDB, 346

Lohman, Guy, 382, 384
Lorraine, Raymond, 113
Low water mark (LWM) in C-Store, 504
LSM-tree concept, 495
Lucking, J. R., 404–405
MacAIMS Data Management System, 404
MacDonald, Nancy, 99
Machine learning, 66
Madden, Samuel, 56f
BigDAWG, 370
C-Store project, 238, 240
C-Store seminal work, 491–518
end of architectural era seminal work, 463–489
Festschrift, 143
H-Store prototype, 247
ISTC, 279, 281
OLTP databases seminal work, 411–439
research contributions article, 183–189
Vertica Systems, 334
Madden, Samuel, on Stonebraker academic career and birth of Ingres, 43–45
advocacy, 50
awards and honors, 49
companies founded, 48–49
early years and education, 42–43
industry, MIT, and new millennium, 46–47
legacy, 47–48
personal life, 50
post-Ingres years, 45–46
synopsis, 41–42
MADlib library, 221–222
Mahony, Colin, 134
Maier, David, 176, 325
Main memory
OLTP design, 466–467
studies, 436
“Making Smalltalk a Database System” (Copeland and Maier), 111
MapReduce
blog post, 114
criticism of, 5, 68–70, 136
and Postgres, 209
Mariposa system
description, 88–89
federation architecture, 401
prototype, 150
Mark, Roger, 370
Marketing problem in one size fits all, 442
MARS system, 436
Mask operators in C-Store, 509
Mattson, Tim, BigDAWG polystore system article, 279–289
McCline, Matt, 215n
McKnight, Kathy, 384
McPherson, John, 382, 384
McQueston, James, 349, 351–353, 355
MDM (master data management), 358
Meehan, John, 376
Memory
OLTP design, 466–467
studies, 436
Memory resident databases in OLTP, 416
Merck databases, 64–65
Meredith, Jeff
Illustra, 206, 221
Miro team, 314f
Postgres, 312, 314
Merge-out process, 495, 508
Message transport in one size fits all, 458
Method and System for Large Scale Data Curation patent, 275
MIMIC (Multiparameter Intelligent Monitoring in Intensive Care) dataset, 370–371, 371f
Miro team, 314f
Miso system, 284
“Mission to Planet Earth” (MTPE) effort, 112–113
Mistakes in startup company guidelines, 128
MIT
2005 sabbatical, 113–114
2016, 115
Aurora and StreamBase projects, 46
Industrial Liaison Program, 121
research contributions, 186
MIT CSAIL, 103
MODIFY command in OVQP, 599–600
Mohan, C., 382, 384
Mom (friend), 27
MonetDB project, 113, 151
Morgenthaler, Gary, on Miro team, 314f
Morpheus project
  description, 47
  prototype, 150
  startup, 90
Morris, Barry
  Aurora/Borealis/StreamBase reunion, 332f
  StreamBase Systems, 232
Mucklo, Matthew, 376
Muffin parallel databases, 151
MUFFIN prototype, 200
Multi-core support in OLTP, 434
Multi-threading in OLTP design, 467
Multilingual access in Postgres, 522
Myria project, 284
MySQL, 137

Nakerud, Jon, 149
NASA “Mission to Planet Earth” effort, 112–113
National Science Foundation (NSF)
  proposal success rate, 74
  RANN program, 147–148
Naughton, Jeff, 113
Naumann, Felix, RDBMS genealogy article, 173–179
Navigational era, 3
Naylor, Arch, 107–108
Nested queries in Postgres, 528
Netezza startup, 221
Network Time Protocol (NTP) for VoltDB, 344–345
New Order transactions in OLTP, 426, 430–432, 431–432f
NewSQL architecture, 246, 282f
No-overwrite storage manager in Postgres, 548
Non-volatile RAM, 67
Nonkeyed storage structure in Ingres, 581
Nonrepeatable errors, 34
Normal functions in Postgres, 524
NoSQL systems, 246
Novartis Institute for Biomedical Research (NIBR), 121, 152, 361
Object identifiers (OIDs) in Postgres, 532
Object Management Extension in commercial Ingres, 308
Object management in Postgres implementation, 520
“Object Management in Postgres Using Procedures” (Stonebraker), 45–46
Object-orientation in Postgres, 206–207, 531–532
Object-Oriented Databases (OODBs), 111–112, 217–218
Object-Relational DBMSs: Tracking the Next Great Wave (Stonebraker and Brown), 111
Object-Relational model, 186
O’Brien, Kyle, 375
O’Connell, Claire, 360
Olson, Mike
  Illustra, 221
  Inversion file system, 215
  open source article, 165–171
  Postgres B-tree implementation, 214n
  Postgres codelines, 313
OLTP (Online Transaction Processing)
  applications in H-Store project, 246–247
  databases seminal work abstract, 411–412
  alternative DBMS architectures, 413
  cache-conscious B-trees, 435–436
  conclusion, 436–437
  concurrency control, 433
  contributions and paper organization, 414–415
  experimental results, 428–432
  future engines, 433–436
instructions vs. cycles, 432
introduction, 412–415
multi-core support, 434
New Order transactions, 430–432, 431–432f
overheads, 414
payment, 429–430, 429f
performance study, 424–432
related work, 436
replication management, 434–435
results, 414–415, 415f
setup and measurement methodology, 427
Shore, 418–424
throughput, 428
trends, 416–418
weak consistency, 435
OLTP (Online Transaction Processing)
design considerations
grid computing and fork-lift upgrades, 468
high availability, 468–470
knobs, 470
main memory, 466–467
multi-threading and resource control, 467
payment transactions, 429–430, 429f
“OLTP: Through the Looking Glass” paper (Harizopoulos), 152
One-shot applications, 472
One-shot transactions, 247, 474, 476
One size does not fit all
BigDAWG polystore system, 282–284
in end of architectural era seminal work, 483–486
overview, 4–5
research contributions, 187–188
special-purpose database systems, 103
One size fits all: An idea whose time has come and gone seminal work
abstract, 441
conclusion, 460
correct primitives, 451–452
data warehouses, 443–445, 444–445f, 455–456
DBMS processing and application logic integration, 452–454, 453f
factoring, 458–459, 459f
financial-feed processing, 446–447, 448f
high availability, 454–455
inbound versus outbound processing, 449–450f, 449–451
introduction, 411–422
performance, 448–455
scientific databases, 457
sensor-based applications, 445–446
sensor networks, 456
stream processing, 445–447, 448f
synchronization, 455
text search, 457
XML databases, 457–458
One-variable detachment in Ingres, 592–593
One-Variable Query Processor (OVQP) in Ingres, 597–599
O’Neil, Pat, C-Store seminal work, 491–518
Ong, James, 206
Open source
BSD and Ingres, 166–167
BSD license, 165
Ingres impact, 167–168
open source Ingres, 309–310
post-Ingres, 168–169
Postgres, 218–220
PostgreSQL, 318–319
research impact, 169–171
OPENR function in Ingres, 583
“Operating System Support for Data Management” (Stonebraker and Kumar), 47
Operators
C-Store queries, 509–510
Postgres, 525–526, 528
scientific data management, 262–263
Optimizations in Shore, 423
OQL language, 208
Oracle Corporation
competition with, 102–103
Index

Oracle Corporation (continued)
performance claims, 149
Postgres attack by, 209
Tamr, 363–364
Orca optimizer, 221
OS/2 Database Manager, 381–383
OS/2 system, 381–383
Ousterhout, John, 216
Ouzzani, Mourad
Data Civilizer, 189
Data Civilizer article, 291–300
“Over My Dead Body” issues in StreamBase, 328–329
Overheads in OLTP, 414
Pagan, Alexander
Data Tamer project, 269–270, 273
Tamr company, 120, 274, 365f
Palmer, Andy, 104
2014 Turing Award Ceremony, 130f
“Cue Ball”, 144
Data Tamer project, 269
Festschrift, 143
startup company article, 129–138
Tamr CEO, 105, 123, 274
Tamr company, 365f
Vertica Systems, 142–143, 242–243
Paper deluge, 159–160
Paper requirements, 159–161
ParAccel, 222
Paradigm4, 89
Paradise project, 113
Parallel databases ideas source, 151
Parallel Data Warehouse project, Microsoft, 47
Parallel DBMS in Postgres, 316
Parallel Sysplex, 400
Parallelism in Tamr, 363
PARAMD function in Ingres, 584
PARAMI function in Ingres, 584
Parsers in Ingres, 586
Partitions in C-Store, 498
Partnerships
leadership approach, 131–132
startup companies, 132–135
Partridge, John
Aurora/Borealis/StreamBase reunion, 332f
Aurora language, 227n
connection points, 229n
RFID tagging, 225n
StreamBase customers, 327–328
StreamBase founding, 232
StreamBase issues, 328–329
Past data access as DBMS limitation, 226
Patents, 126
Path expressions in Postgres, 528
Patterson, Dave, 216
Pavlo, Andrew
H-Store project, 187
H-Store project article, 245–251
VoltDB executor in H-Store, 75
PayPal, 250
Pearson Correlation Coefficient (PCC), 292
People for startup companies, 138
Performance
BigDAWG polystore system, 287, 288f
bottleneck studies, 436
C-Store, 151, 511–515
Data Unification, 358
H-Store, 479–483, 479f
Ingres, 602
locking methods, 90–91
OLTP, 424–432
one size fits all, 448–455
Postgres, 30–32, 549–550, 554–555, 555f
Permute operators in C-Store, 509
Perna, Janet, 382, 384, 401
Persistence
Postgres, 522
VoltDB, 346–347
Persistent CLOS, 522
Persistent redo logs, 470
Personal life of Stonebraker, 50
Peterlee IS/1 System, 404
Ph.D. paper requirements, 159
Pipes, 572
Pirahesh, Hamid, 382, 384
Pitch decks in startup company guidelines, 123–125
Pivotal company, 221–222
Plans for C-Store queries, 509–510
Poliakov, Alex, SciDB codeline article, 349–355
Polybase system, 284
Polystores. See BigDAWG polystore system
Portable code base for IBM relational databases, 381–384
Post-Ingres years
open source, 168–169
overview, 45–46
Postgres codelines, 311
conclusion, 319
PostgreSQL, 317–319
prototype, 311–313
Postgres design, 15–16
base types, 523
bicycle trip metaphor, 32–35
conclusion, 35–36
Illustra buyout, 32
inheritance, 524, 529
Ingres help for, 27–28
Internet takeoff, 30
Land Sharks, 29
marketing challenge, 28–29
performance benchmark, 30–32
speedbumps, 24–26
start, 15–22, 22f
Postgres implementation seminal work
abstract, 519
conclusion, 555–557
data model, 523–528
data model and query language overview, 521–523
data model critique, 532–537
design problems, 549–550
dynamic loading and process structure, 551–552
fast path feature, 530–531
implementation introduction, 550–554
introduction, 520–521
object-orientation, 531–532
POSTQUEL query language, 528–530
programming language, 552–554
rules systems, 538–547
status and performance, 554–555, 555f
storage systems, 547–550
Postgres perspective, 205
active databases and rule systems, 212–213
commercial adaptations, 220–223
context, 205–206
deep storage technologies, 215–216
language support, 217–218
lessons, 223–224
log-centric storage and recovery, 213–215
overview, 206–207
software impact, 218–223
XPRS architecture, 216–217
Postgres project, 102
abstract data types, 88, 149
description, 186
ideas source, 149–150
Illustra purchase, 111–112
impact, 3
POSTQUEL, 400
productionization, 315
satisfaction with, 78–79
and SQL, 315–316
start of, 203–204
PostgreSQL
creation, 170, 400
impact, 3
open source, 218–220, 317–319
software architecture, 210
POSTQUEL query language
features, 528–530
functions, 526–528, 555–556, 555f
Potamianos, Spyros, 212
Pragmatism in startup companies, 135–137
Pricing models, 134
Primary-copy replication control, 90
Primary indexes, 493
Primitives in one size fits all, 451–452
Princeton University, 43
Index

Probabilistic reasoning in scientific data management, 266

Problems
ignoring, 163
solving, 276–277

Process 2 in Ingres, 585–591
Process 3 in Ingres, 591–599
Process 4 in Ingres, 599–602

Process structure
Ingres, 571–575, 572f, 574f
Postgres, 551–552

Project operators in C-Store, 509
Project Oxygen, 225
Project Sequoia
2000, 112–113
Postgres, 215

Projections in C-Store, 496–498, 510
PROTECTION catalog in Ingres, 580

Prototypes
ADT-Ingres, 202, 203f
Data Tamer project, 270–273
H-Store project, 247–250
Mariposa, 150
Morpheus, 150
MUFFIN, 200
noise in, 389
Postgres, 311–313
Shore, 152
startup companies, 120–121
Tamr project, 121

PRS2 system, 212
Punctuated Streams Team, 325
Purify tool for Postgres productionization, 315

Putzolu, Franco, 101
PVLDB 2016 paper, 297–298

Qatar Computing Research Institute (QCRI)
creation, 105
Data Civilizer project, 189
Data Tamer, 152
Tamr project, 120
Quel language, 102
comments, 570–571
complex objects, 203
description, 195–196, 197f
overview, 563–565
and standardization, 398–399
utility commands, 565–568

Query classes in H-Store, 480–482
Query decomposition in Ingres, 591–597
Query execution
C-Store, 509–510
H-Store, 474–475
Query modeling and optimization in BigDAWG, 285–286
Query modification in Ingres, 90, 585–588
Query optimization in C-Store, 510
Query rewrite implementation in rules systems, 538–541

Quiet (friend), 27

R-Tree index structure
Ingres, 201–202
and Postgres, 210

Radical simplicity for Postgres transactional storage, 214
RAID storage architectures, 216
Raising money for startup companies, 127
RAP project, 109
RARES project, 109
Rasin, Alex, 334
on Aurora project, 324f
RCA company, 97–98
Ré, Chris, 6
Read-optimized systems, 492
Real-time requirements as DBMS limitation, 227

Real-world impact in rules of thumb, 236–238
Record deduplication in Data Tamer project, 272–273
Recorded Future company, 296–297

Recovery
C-Store, 506–508
database logs for, 21–22
H-Store, 476–478
Ingres, 600–602
Postgres, 213–216
Red Brick Systems, 66
Redo logs
H-Store, 482
OLTP design, 470
“Reduction of Large Scale Markov Models for Random Chains” dissertation (Stonebraker), 43
Referential integrity, 538–539
Reformatting tuples in Ingres, 593
Relational-CODASYL debate, 404–406
Relational database industry birth, 94
Ingres competition, 100–103
Ingres team, 99–100
Ingres timing, 98–99
maturity stage, 103–105
overview, 97–98
Relational database management systems (RDBMS)
Ingres later years, 194–198, 197f
Relational databases, brief history of, 2–6
Relational era, 3
Relational models in one size does not fit all world, 483–485
Relations
Ingres, 576–580
QUEL, 566
Remote direct memory access (RDMA), 67
Rendezvous system, 404
REPLACE command in Ingres, 584, 600–601
Replication management in OLTP, 434–435
Research, open source impact on, 169–171
Research Applied to the National Needs (RANN) program, 147–148
Research contributions
2010s and beyond, 188–189
Berkeley years, 185–186
MIT, 186
one size doesn’t fit all era, 187–188
technical rules of engagement, 183–185
Research story about Aurora project, 387–391
Resident set size (RSS) in VoltDB, 342
Resource control in OLTP design, 467
RETRIEVE commands in Ingres, 573
Reviews, unsatisfactory, 160–161
RFID (radio frequency identification) tagging, 225n
Ries, Dan, 200
Rivers, Jonathan, 350
Robinson, John “JR”
Shared Nothing band, 138f
Tamr, 365f
Vertica Systems, 339
Rock fetches, 124
Rogers, Jennie, BigDAWG polystore system article, 279–289
Rollbacks in C-Store, 506
Roots in OLTP tables, 471
Route 66 TV show, 81
Row store architecture, 492
Rowe, Larry
commercial Ingres codeline, 305
Ingres founding, 303
Postgres, 36, 88, 206, 217
Postgres implementation seminal work, 519–559
RTI founding, 101, 303
RS column store in C-Store, 500–501
RTI (Relational Technology, Inc.), 101
commercial version, 111
founding, 166–167, 398
Ingres basis of, 198
Rubenstein, Brad, 205
Ruby-on-Rails system, 486
Rules systems in Postgres, 212–213
complexity, 541–543
implementation efficiency, 544
introduction, 538–541
knowledge management, 520
push for, 316
second system, 545–547
views, 543–544
S-Store project, 234, 331
Sales in startup company guidelines, 128
Sales problem in one size fits all, 442
Salz, Jon, 232, 328–330
Sarawagi, Sunita, 215
ScaLAPACK analytics, 349
Scaling in Tamr, 362–365
Schek, Hans, 92
Schema mapping in Data Tamer project, 270–272, 271f
Schieffer, Berni, 384, 385f
Schlamb, Kelly
Data Civilizer, 64, 294
Tamr, 361
Schultz, Hayden, 331
Schuster, Stu, 101
SciDB codeline, 349
connectivity, 349–351
features focus, 351–352
genomic data, 354–355
hard numbers, 352–353
languages, 353–354
security, 354
SciDB project
contributions for, 89
description, 47
one size doesn’t fit all era, 188
Scientific data management, 253–254
beginning tasks, 260–263, 261f
current users, 267–268
first users, 263–267
logistics, 259–260
mountain representation, 254–256
planning, 256–259
Scientific databases in one size fits all, 457
Scope in BigDAWG polystore system, 284
SDTM (Study Data Tabulation Model), 364
Search in one size fits all, 457
Search User Interfaces (Hearst), 394
Second System Effect, 223
Secondary indexes, 493
Secrecy in startup company guidelines, 127
Security in SciDB, 354
Segments in C-Store, 498
Self-funded companies, 74
Self-order values in C-Store, 501
Selinger, Pat, 382, 384, 407
Sensor-based applications in one size fits all, 445–446, 456
SEQUEL language, 102
Service of Stonebraker, 49
Shankland, Jim, on Miro team, 314f
Shared-nothing architecture in SciDB codeline, 351
Shared Nothing band, 80, 138f
Sharma, Kristi Sen, SciDB codeline article, 349–355
She, Zuohao (Jack), 373
Shims in BigDAWG polystore system, 284
Shore (Scalable Heterogeneous Object Repository), 418–419
architecture, 419–422, 420f
prototype, 152
removing components, 422–424
Shore Storage Manager (SSM), 418
Short One (friend), 27
Sibley, Ed, 404, 406
SIGFIDET conference, 403–408
Singer, Adam, on Aurora project, 324f
Single-partition transactions in H-Store project, 247
Single-sited transactions, 472, 474, 476
Single threading in OLTP, 413, 416–417
Skeen, Dale, 195, 200
Skok, David, 151–152
Sleepycat startup, 171
Sloan Digital Sky Survey (SDSS), 113, 255–256
Slotted pages in Shore, 419
SMALLTALK language, 553
Smooth (friend), 28
Snapshot isolation, 495, 503–505
Snodgrass, Rick, 198
Software impact in Postgres, 218–223
Solicitation in startup company guidelines, 123–125
Sort keys and operators in C-Store, 497–498,
Source code for Ingres project, 109–110
SOURCE directory in Ingres, 576
Space budget in C-Store, 500
Spanner, 70–71
Spark, 402
Spending money guidelines for startup companies, 125–126
Sprite distributed OS, 216
SQL language, 102
introduction, 404
MapReduce, 114
one size does not fit all world, 485–486
and Postgres, 315–316
vs. Quel, 196
SQuAl system, 323
Stable memory in Postgres, 548
Stanford Linear Accelerator (SLAC) facility, 257
Star schema in data warehousing, 443, 444f
Starburst project, 213
Startup companies, founded, 48–49
Startup companies, guidelines
business acumen on team, 122–123
company control, 127
first customers, 126
ideas, 119–120
intellectual property, 126
introduction, 119
lighthouse customers, 122
mistakes, 128
pitch deck and VC solicitation, 123–125
raising money, 127
sales, 128
secrecy, 127
spending money, 125–126
summary, 128
teams and prototypes, 120–121
venture capitalists, 127
Startup companies, running
introduction, 129–130
overview, 130–132
partnerships, 132–135
people in, 138
pragmatism, 135–137
State storage in one size fits all, 458
Status in Postgres, 554–555, 555f
Sterile transaction classes, 473, 476
Stonebraker, Beth, 50, 141, 144
Stonebraker, Leslie, 50, 141, 144
Stonebraker, Michael
collected works, 607–633
failures article, 155–164, 156f
ideas article, 147–153
Postgres design, construction, and commercialization story, 15–37
startup company guidelines. See Startup companies, guidelines
Winslett interview, 59–83
Stonebraker, Michael, biography overview
academic career and birth of Ingres, 43–45
academic positions, 43
advocacy, 50, 91–92
awards and honors, 49
career flowchart, 54–55
companies founded, 48–49
eyears and education, 42–43
industry, MIT, and new millennium, 46–47
legacy, 47–48
personal life, 50
post-Ingres years, 45–46
sabbatical at MIT, 113–114
student genealogy chart, 52–53, 56f
synopsis, 41–42
Stonebraker, Michael, seminal works
C-Store, 491–518
end of architectural era, 463–489
Ingres implementation, 561–605
OLTP databases, 411–439
one size fits all, 441–462
Postgres implementation, 519–559
Stonebraker’s good ideas
abstract data types, 148
Data Tamer, 152
data warehouses, 151
distributed databases, 150
H-Store/VoltDB, 151–152
Stonebraker’s good ideas (continued)
how to exploit, 153
Ingres, 147–148
parallel databases, 151
Postgres, 149–150
startup company guidelines, 119–120
Stonebraker, Sandra, 50, 141, 144
Storage allocators in C-Store, 502–503
Storage keys in C-Store, 498
Storage management and structures
C-Store, 502–503
commercial Ingres codeline, 306–308
Ingres, 580–582, 582f
Postgres, 213–216, 547–550
QUEL, 566–567
Stored procedures, 91
STRATEGY program in OVQP, 597–599
Stream processing era
Aurora and Borealis origins, 225–227
Aurora and Borealis systems, 227–231
concurrent efforts, 231–232
current systems, 233–234
StreamBase Systems, 232–233
Stream processing in one size fits all,
445–447, 448f
STREAM project, 231
Stream-SQL, enthusiasm for, 484
STREAM Team, 325
StreamBase codelines, 321–322, 322f
April Fool’s Day joke, 330–331
conclusion, 331–332
customers, 327–328
development, 326–327
issues, 328–330
StreamBase Systems
Architecture Committee, 329
aggregation systems, 451
from Aurora, 89
founding, 46, 232–233
Grassy Brook renamed to, 142
textual language, 329
Strongly two-phase applications, 473
Student genealogy chart, 52–53, 56f
Student perspective, 393–396, 395f
Subject matter experts (SMEs) in Tamr,
361–362
Survey of Income and Program Participation
(SIPP) data, 197–198
Sybase, 400
Synchronization in one size fits all, 455
Sysplex Coupling Facility, 380
System catalogs in Ingres, 578–580
System i, 380
System-level data management problems
and approaches, 91–92
System R system, 88, 100
architectural features, 465
code base, 380
development, 44
vs. Ingres, 196
Systems, leadership and advocacy, 87–90
Szalay, Alex, 112–113
Szolovits, Peter, 370
T-trees, 436
Table fragments in H-Store, 475
Tall Shark (friend), 27
Tamr codeline, 357
algorithmic complexity, 359–361
conclusion, 365–366, 366f
Data Unification, 358–359
user emphasis, 361–362
variety, 362–365
Tamr project and company
creation, 105
from Data Tamer, 90
founding, 273–275
idea for, 120, 152
prototype, 120
Tandem Computers, 101
Tang, Nan
Data Civilizer, 189
Data Civilizer article, 291–300
Tango, Jo
at Highland event, 132
venture capitalist perspective article,
139–144
Tarashansky, Igor, 352
Tatbul, Nesime
- Aurora/Borealis/StreamBase codelines article, 321–332
- Aurora/Borealis/StreamBase reunion, 332f

Taylor, Cimarron, on Miro team, 314f

Teams for startup companies, 120–121

Technical rules of engagement, 183–185

Technology Licensing Offices (TLOs), 126

Telegraph Team, 325

TelegraphCQ project, 231

Telenauv company, 90

Temporal Functional Dependencies in Data Civilizer, 297

Tenure paper requirements, 159

Teradata, 222

Term sheets in startup company guidelines, 124–125

Terminal monitor in Ingres, 572

Test-of-time award, 202

Text search in one size fits all, 457

Thomson Reuters (TR) company, 274

Thread support in Shore, 420–421

Three dimensional problems, 520

Throughput in OLTP, 428

Tibbetts, Richard
- Aurora/Borealis/StreamBase reunion, 332f

StreamBase development, 326–327

StreamBase issues, 328–330

StreamBase Systems, 232

TIBCO Software, Inc., 233, 321

Time travel feature in Postgres, 316, 529, 548, 550

TimeSeries DataBlade in Postgres, 317

Timestamp authorities (TAs) in C-Store, 504

TimesTen system, 436

TMP directory in Ingres, 576

TPC (Transaction Processing Performance Council) benchmark
- Data Unification, 358
- H-Store, 479–483, 479f
- OLTP, 424–425, 425f
- TPC-B, 382, 436

Training workloads in C-Store, 500

Trijman, Omer, 335

Tran, Nga, C-Store seminal work, 491–518

Transaction-less databases, 413

Transactions
- C-Store, 503–508, 505f
- concurrency control. See Concurrency control features, 470–471
- H-Store, 247, 476–478
- OLTP, 417–418
- rollbacks, 506
- schema characteristics, 471–473

Transitive closure in Postgres, 529

Trees schemas, 471–472

Triggers
- DBMS limitation, 226
- one size fits all, 450
- Postgres, 213
- rules systems, 212, 539–540

Triple Rock (friend), 27

Trust with venture capitalists, 140

Tsichritzis, Dennis, 403–404

Tuples in Ingres
- AMI, 583
- substitution, 592
- TIDs, 581
- variables, 563

Turing Award in 2014
- citation, 130
- overview, 114–115
- perspectives, 93–95

Two-phase applications, 473, 476

Types in Postgres, 523, 532–537

Ubell, Michael

- Illustra, 221
- Miro team, 314f
- Postgres productionization, 315
Undo logs
H-Store, 482
OLIN process structure, 471
UNIN process structure, 571–572
Union types in Postgres, 532–534
Unix platforms
Ingres, 101, 571–573
systems based on, 47
Updates
C-Store, 503–508, 505f
Ingres, 600–602
Uptone (friend), 28, 30
Urban Dynamics, 43
Urban systems, 147–148
User-Defined Aggregate (UDA) functions in Postgres, 209
User-defined extensions (UDXs)
Ingres prototype, 202
SciDB codeline, 350
User-defined functions (UDFs) in Postgres, 209, 211
User-defined types (UDTs) in commercial Ingres codeline, 308–309
User emphasis in Tamr, 361–362
User experience (UX) design and implementation in Tamr, 361
User feedback for Ingres, 602–603
Utility commands in Ingres, 563–568, 599–602

VanderPlas, Jake, 353
Varaiya, Pravin, 17, 148
Variety in Tamr, 361–365
Venture capitalists
perspective, 139–144
in startup company guidelines, 127
Verisk Health, 121, 152
Vernica, Rares, 350, 353–354
Vertica Systems
from C-Store, 89
creation, 104
founding, 133–135, 242–244
HP purchase of, 67
impact of, 113–114
patent infringement suit, 126
satisfaction with, 78–79
Tamr, 364
venture capitalist perspective, 142–143
Vertica Systems codeline, 333
architectural decisions, 336–339
building, 333–334
conclusion, 340
customers, 334–335, 339–340
features discussion, 335–336
Video data management, 265
Vietnam war, 81
Views
Ingres, 44, 586
rules systems, 543–544
Vincent, Tim, 383–384, 385f
VLDB demo paper for H-Store prototype, 249
VLSI CAD design era, 201–202
Voice-of-Experience (friend), 28, 30
VoltDB
creation, 104
from H-Store, 89
H-Store executor, 75
H-Store split, 251
PayPal interest in, 250
VoltDB codeline, 341–342
compaction, 342–344
disk persistence, 346–347
latency, 344–348
Volume in Tamr, 361–362
Weak consistency in OLTP, 435
Wei Hong Optimizer, 217
Weisberg, Ariel, 249
Whales, 158
Whitney, Kevin, 404
Whittaker, Andrew, 370
Whyte, Nick, 99
Widom, Jennifer, 228–229, 325
Winer, Mike, 385f
WinFS project, 215n
Winslet, Marianne, interview with Stonebraker, 59–83
Wisconsin, 1996 fall, 108–111
Wisconsin Benchmark, 436
Wong, Eugene, 98
   Ingres, 88
   Ingres founding, 41, 148, 303
   Ingres implementation seminal work, 561–605
   RTI founding, 398
   Stonebraker guided by, 3, 43
   Worker sites in H-Store, 477
Workflow-based diagrammatic languages, 227n
Workload in OLTP, 425–427, 426f
Write-ahead logging in Postgres, 214
Write-optimized systems, 492
WS column store in C-Store, 502, 507–508
Xiao, Min
   OMDB, 338
   Vertica Systems, 339
Xing, Ying, on Aurora project, 324f
XML databases in one size fits all, 457–458
XPRS architecture in Postgres, 216–217
XQuery language, 208
XRM-An Extended (N-ary) Relational Memory, 404
Yan, Robin, on Aurora project, 324f
Youssefi, Karel
   Ingres team, 99
   Tandem Computers, 101
Yu, Andrew
   Postgres parser, 218
   PostgreSQL, 170
   SQLization project, 317–318
Yu, Katherine, 376
Zaniolo, Carlo, 217
Zdonik, Stan, 103
   Aurora/Borealis/StreamBase reunion, 332f
   Aurora project, 322–323, 324f
   Borealis project, 186
   expert sourcing, 273
   H-Store project, 245–246
   Shared Nothing band, 138f
   stream processing era article, 225–234
   StreamBase Systems, 232
   Tamr project, 120
   Vertica Systems, 334, 339
   Zero-billion-dollar ideas, 185
Zhang, Donghui, 353–354
Zilles, Stephen, 88
Zook, Bill, 99
Editors

Michael L. Brodie

Michael L. Brodie has over 45 years of experience in research and industrial practice in databases, distributed systems, integration, artificial intelligence, and multidisciplinary problem-solving. Dr. Brodie is a research scientist at the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology; advises startups; serves on advisory boards of national and international research organizations; and is an adjunct professor at the National University of Ireland, Galway and at the University of Technology, Sydney. As Chief Scientist of IT at Verizon for over 20 years, he was responsible for advanced technologies, architectures, and methodologies for IT strategies and for guiding industrial-scale deployments of emerging technologies. He has served on several National Academy of Science committees. Current interests include Big Data, Data Science, and Information Systems evolution. Dr. Brodie holds a Ph.D. in databases from the University of Toronto and a Doctor of Science (honoris causa) from the National University of Ireland. Visit www.Michaelbrodie.com for further information.


**Authors**

**Daniel J. Abadi**

Daniel J. Abadi is the Darnell-Kanal Professor of Computer Science at the University of Maryland, College Park. He performs research on database system architecture and implementation, especially at the intersection of scalable and distributed systems. He is best known for the development of the storage and query execution engines of the C-Store (column-oriented database) prototype, which was commercialized by Vertica and eventually acquired by Hewlett-Packard, and for his HadoopDB research on fault-tolerant scalable analytical database systems, which was commercialized by Hadapt and acquired by Teradata in 2014. Abadi has been a recipient of a Churchill Scholarship, a NSF CAREER Award, a Sloan Research Fellowship, a VLDB Best Paper Award, a VLDB 10-year Best Paper Award, the 2008 SIGMOD Jim Gray Doctoral Dissertation Award, the 2013–2014 Yale Provost’s Teaching Prize, and the 2013 VLDB Early Career Researcher Award. He received his Ph.D. in 2008 from MIT. He blogs at DBMS Musings (http://dbmsmusings.blogspot.com) and Tweets at @daniel_abadi.

**Magdalena Balazinska**

Magdalena Balazinska is a professor in the Paul G. Allen School of Computer Science and Engineering at the University of Washington and is the director of the University’s eScience Institute. She’s also director of the IGERT PhD Program in Big Data and Data Science and the associated Advanced Data Science PhD Option. Her research interests are in database management systems with a current focus on data management for data science, big data systems, and cloud computing. Magdalena holds a Ph.D. from the Massachusetts Institute of Technology (2006). She is a Microsoft Research New Faculty Fellow (2007) and received the inaugural VLDB Women in Database Research Award (2016), an ACM SIGMOD Test-of-Time Award (2017), an NSF CAREER Award (2009), a 10-year most influential paper award (2010), a Google Research Award (2011),
an HP Labs Research Innovation Award (2009 and 2010), a Rogel Faculty Support Award (2006), a Microsoft Research Graduate Fellowship (2003–2005), and multiple best-paper awards.

**Nikolaus Bates-Haus**

Nikolaus Bates-Haus is Technical Lead at Tamr Inc., an enterprise-scale data unification company, where he assembled the original engineering team and led the development of the first generation of the product. Prior to joining Tamr, Nik was Lead Architect and Director of Engineering at Endeca (acquired by Oracle in 2011), where he led development of the MDEX analytical database engine, a schema-on-read column store designed for large-scale parallel query evaluation. Previously, Nik worked in data integration, machine learning, parallel computation, and real-time processing at Torrent Systems, Thinking Machines, and Philips Research North America. Nik holds an M.S. in Computer Science from Columbia University and a B.A. in Mathematics/Computer Science from Wesleyan University. Tamr is Nik’s seventh startup.

**Philip A. Bernstein**

Philip A. Bernstein is a Distinguished Scientist at Microsoft Research, where he has worked for over 20 years. He is also an Affiliate Professor of Computer Science at the University of Washington. Over the last 20 years, he has been a product architect at Microsoft and Digital Equipment Corp., a professor at Harvard University and Wang Institute of Graduate Studies, and a VP Software at Sequoia Systems. He has published over 150 papers and 2 books on the theory and implementation of database systems, especially on transaction processing and data integration, and has contributed to a variety of database products. He is an ACM Fellow, an AAAS Fellow, a winner of ACM SIGMOD’s Codd Innovations Award, a member of the Washington State Academy of Sciences, and a member of the U.S. National Academy of Engineering. He received a B.S. from Cornell and M.Sc. and Ph.D. degrees from the University of Toronto.
Janice L. Brown

Janice L. Brown is president and founder of Janice Brown & Associates, Inc., a communications consulting firm. She uses strategic communications to help entrepreneurs and visionary thinkers launch technology companies, products, and ventures, as well as sell their products and ideas. She has been involved in three ventures (so far) with 2014 Turing Award-winner Michael Stonebraker: Vertica Systems, Tamr, and the Intel Science and Technology Center for Big Data. Her background includes positions at several public relations and advertising agencies, and product PR positions at two large technology companies. Her work for the Open Software Foundation won the PRSA’s Silver Anvil Award, the “Oscar” of the PR industry. Brown has a B.A. from Simmons College. Visit www.janicebrown.com.

Paul Brown

Paul Brown first met Mike Stonebraker in early 1992 at Brewed Awakening coffee shop on Euclid Avenue in Berkeley, CA. Mike and John Forrest were interviewing Paul to take over the job Mike Olson had just left. Paul had a latte. Mike had tea. Since then, Paul has worked for two of Mike’s startups: Illustra Information Technologies and SciDB / Paradigm4. He was co-author with Mike of a book and a number of research papers. Paul has worked for a series of DBMS companies all starting with the letter “I”: Ingres, Illustra, Informix, and IBM. Alliterative ennui setting in, Paul joined Paradigm4 as SciDB’s Chief Architect. He has since moved on to work for Teradata. Paul likes dogs, DBMSs, and (void *). He hopes he might have just picked up sufficient gravitas in this industry to pull off the beard.
Paul Butterworth

Paul Butterworth served as Chief Systems Architect at Ingres from 1980–1990. He is currently co-founder and Chief Technology Officer (CTO) at VANTIQ, Inc. His past roles include Executive Vice President, Engineering at You Technology Inc., and co-founder and CTO of Emotive Communications, where he conceived and designed the Emotive Cloud Platform for enterprise mobile computing. Before that, Paul was an architect at Oracle and a founder & CTO at AmberPoint, where he directed the technical strategy for the AmberPoint SOA governance products. Prior to AmberPoint, Paul was a Distinguished Engineer and Chief Technologist for the Developer Tools Group at Sun Microsystems and a founder, Chief Architect, and Senior Vice President of Forte Software. Paul holds undergraduate and graduate degrees in Computer Science from UC Irvine.

Michael J. Carey

Michael J. Carey received his B.S. and M.S. from Carnegie-Mellon University and his Ph.D. from the University of California, Berkeley, in 1979, 1981, and 1983, respectively. He is currently a Bren Professor of Information and Computer Sciences at the University of California, Irvine (UCI) and a consulting architect at Couchbase, Inc. Before joining UCI in 2008, Mike worked at BEA Systems for seven years and led the development of BEA’s AquaLogic Data Services Platform product for virtual data integration. He also spent a dozen years teaching at the University of Wisconsin-Madison, five years at the IBM Almaden Research Center working on object-relational databases, and a year and a half at Propel Software, an e-commerce platform startup, during the infamous 2000–2001 Internet bubble. He is an ACM Fellow, an IEEE Fellow, a member of the National Academy of Engineering, and a recipient of the ACM SIGMOD E.F. Codd Innovations Award. His current interests center on data-intensive computing and scalable data management (a.k.a. Big Data).
Fred Carter

Fred Carter, a software architect in a variety of software areas, worked at Ingres Corporation in several senior positions, including Principal Scientist/Chief Architect. He is currently a principal architect at VANTIQ, Inc. Prior to VANTIQ, Fred was the runtime architect for AmberPoint, which was subsequently purchased by Oracle. At Oracle, he continued in that role, moving the AmberPoint system to a cloud-based, application performance monitoring service. Past roles included architect for EAI products at Forte (continuing at Sun Microsystems) and technical leadership positions at Oracle, where he designed distributed object services for interactive TV, online services, and content management, and chaired the Technical Committee for the Object Definition Alliance to foster standardization in the area of network-based multimedia systems. Fred has an undergraduate degree in Computer Science from Northwestern University and received his M.S. in Computer Science from UC Berkeley.

Raul Castro Fernandez

Raul Castro Fernandez is a postdoc at MIT, working with Samuel Madden and Michael Stonebraker on data discovery—how to help people find relevant data among databases, data lakes, and the cloud. Raul built Aurum, a data discovery system, to identify relevant data sets among structured data. Among other research lines, he is looking at how to incorporate unstructured data sources, such as PDFs and emails. More generally, he is interested in data-related problems, from efficient data processing to machine learning engineering. Before MIT, Raul completed his Ph.D. at Imperial College London, where he focused on designing new abstractions and building systems for large-scale data processing.
Ugur Çetintemel

Ugur Çetintemel is a professor in the department of Computer Science at Brown University. His research is on the design and engineering of high-performance, user-friendly data management and processing systems that allow users to analyze large data sets interactively. Ugur chaired SIGMOD ’09 and served on the editorial boards of VLDB Journal, Distributed and Parallel Databases, and SIGMOD Record. He is the recipient of a National Science Foundation Career Award and an IEEE 10-year test of time award in Data Engineering, among others. Ugur was a co-founder and a senior architect of StreamBase, a company that specializes in high-performance data processing. He was also a Brown Manning Assistant Professor and has been serving as the Chair of the Computer Science Department at Brown since July 2014.

Xuedong Chen

Xuedong Chen is currently an Amazon.com Web Services software developer in Andover, Massachusetts. From 2002–2007 he was a Ph.D. candidate at UMass Boston, advised by Pat and Betty O’Neil. He, along with Pat O’Neil and others, were co-authors with Mike Stonebraker.

Mitch Cherniack

Mitch Cherniack is an Associate Professor at Brandeis University. He is a previous winner of an NSF Career Award and co-founder of Vertica Systems and StreamBase Systems. His research in Database Systems has focused on query optimization, streaming data systems, and column-based database architectures. Mitch received his Ph.D. from Brown University in 1999, an M.S. from Concordia University in 1992, and a B.Ed. from McGill University in 1984.
David J. DeWitt

David J. DeWitt joined the Computer Sciences Department at the University of Wisconsin in September 1976 after receiving his Ph.D. from the University of Michigan. He served as department chair from July 1999 to July 2004. He held the title of John P. Morgridge Professor of Computer Sciences when he retired from the University of Wisconsin in 2008. In 2008, he joined Microsoft as a Technical Fellow to establish and manage the Jim Gray Systems Lab in Madison. In 2016, he moved to Boston to join the MIT Computer Science and AI Laboratory as an Adjunct Professor. Professor DeWitt is a member of the National Academy of Engineering (1998), a fellow of the American Academy of Arts and Sciences (2007), and an ACM Fellow (1995). He received the 1995 Ted Codd SIGMOD Innovations Award. His pioneering contributions to the field of scalable database systems for “big data” were recognized by ACM with the 2009 Software Systems Award.

Aaron J. Elmore

Aaron J. Elmore is an assistant professor in the Department of Computer Science and the College of the University of Chicago. Aaron was previously a postdoctoral associate at MIT working with Mike Stonebraker and Sam Madden. Aaron’s thesis on *Elasticity Primitives for Database-as-a-Service* was completed at the University of California, Santa Barbara under the supervision of Divy Agrawal and Amr El Abbadi. Prior to receiving a Ph.D., Aaron spent several years in industry and completed an M.S. at the University of Chicago.

Miguel Ferreira

Miguel Ferreira is an alumnus of MIT. He was coauthor of the paper, “Integrating Compression and Execution in Column-Oriented Database Systems,” while working with Samuel Madden and Daniel Abadi, and “C-store: A Column-Oriented DBMS,” with Mike Stonebraker, Daniel Abadi, and others.
Vijay Gadepally

Vijay Gadepally is a senior member of the technical staff at the Massachusetts Institute of Technology (MIT) Lincoln Laboratory and works closely with the Computer Science and Artificial Intelligence Laboratory (CSAIL). Vijay holds an M.Sc. and Ph.D. in Electrical and Computer Engineering from The Ohio State University and a B.Tech in Electrical Engineering from the Indian Institute of Technology, Kanpur. In 2011, Vijay received an Outstanding Graduate Student Award at The Ohio State University. In 2016, Vijay received the MIT Lincoln Laboratory’s Early Career Technical Achievement Award and in 2017 was named to AFCEA’s inaugural 40 under 40 list. Vijay’s research interests are in high-performance computing, machine learning, graph algorithms, and high-performance databases.

Nabil Hachem

Nabil Hachem is currently Vice President, Head of Data Architecture, Technology, and Standards at MassMutual. He was formerly Global Head of Data Engineering at Novartis Institute for Biomedical Research, Inc. He also held senior data engineering posts at Vertica Systems, Inc., Infinity Pharmaceuticals, Upromise Inc., Fidelity Investments Corp., and Ask Jeeves Inc. Nabil began his career as an electrical engineer and operations department manager for a data telecommunications firm in Lebanon. In addition to his commercial career, Nabil taught computer science at Worcester Polytechnic Institute. He co-authored dozens of papers on scientific databases, file structures, and join algorithms, among others. Nabil received a degree in Electrical Engineering from the American University of Beirut and earned his Ph.D. in Computer Engineering from Syracuse University.
Don Haderle

Don Haderle joined IBM in 1968 as a software developer and retired in 2005 as the software executive operating as Chief Technology Officer (CTO) for Information Management. He consulted with venture capitalists and advised startups. He currently sits on technical advisory boards for a number of companies and consults independently. Considered the father of commercial high-performance, industrial-strength relational database systems, he was the technical leader and chief architect of DB2 from 1977–1998. He led DB2’s overall architecture and development, making key personal contributions to and holding fundamental patents in all key elements, including: logging primitives, memory management, transaction fail-save and recovery techniques, query processing, data integrity, sorting, and indexing. As CTO, Haderle collaborated with researchers to incubate new product directions for the information management industry. Don was appointed an IBM Fellow in 1989 and Vice President of Advanced Technology in 1991; named an ACM Fellow in 2000; and elected to the National Academy of Engineering in 2008. He is a graduate of UC Berkeley (B.A., Economics, 1967).

James Hamilton

James Hamilton is Vice President and Distinguished Engineer on the Amazon Web Services team, where he focuses on infrastructure efficiency, reliability, and scaling. He has spent more than 20 years working on high-scale services, database management systems, and compilers. Prior to joining AWS, James was architect on the Microsoft Data Center Futures team and the Windows Live Platform Services team. He was General Manager of the Microsoft Exchange Hosted Services team and has led many of the SQL Server engineering teams through numerous releases. Before joining Microsoft, James was Lead Architect on the IBM DB2 UDB team. He holds a B.Sc. in Computer Science from the University of Victoria and a Master’s in Math, Computer Science from the University of Waterloo.
Stavros Harizopoulos

*Stavros Harizopoulos* is currently a Software Engineer at Facebook, where he leads initiatives on Realtime Analytics. Before that, he was a Principal Engineer at AWS Redshift, a petabyte-scale columnar Data Warehouse in the cloud, where he was leading efforts on performance and scalability. In 2011, he co-founded Amiato, a fully managed real-time ETL cloud service, which was later acquired by Amazon. In the past, Stavros has held research-scientist positions at HP Labs and MIT CSAIL, working on characterizing the energy efficiency of database servers, as well as dissecting the performance characteristics of modern in-memory and column-store databases. He is a Carnegie Mellon Ph.D. and a Y Combinator alumnus.

Marti Hearst

*Marti Hearst* is a professor in the School of Information and the EECS Department at UC Berkeley. She was formerly a member of the research staff at Xerox PARC and received her Ph.D. from the CS Division at UC Berkeley. Her primary research interests are user interfaces for search engines, information visualization, natural language processing, and improving education. Her book *Search User Interfaces* was the first of its kind in academics. Prof. Hearst was named a Fellow of the ACM in 2013 and a member of the CHI Academy in 2017, and is president of the Association for Computational Linguistics. She has received four student-initiated Excellence in Teaching Awards.
**Jerry Held**

Jerry Held has been a successful Silicon Valley entrepreneur, executive, and investor for over 40 years. He has managed all growth stages of companies, from conception to multi-billion-dollar global enterprise. He is currently chairman of Tamr and Madaket Health and serves on the boards of NetApp, Informatica, and Copia Global. His past board service includes roles as executive chairman of Vertica Systems and MemSQL and lead independent director of Business Objects. Previously, Dr. Held was “CEO-in-residence” at venture capital firm Kleiner Perkins Caufield & Byers. He was senior vice president of Oracle Corporation’s server product division and a member of the executive team that grew Tandem Computers from pre-revenue to multi-billion-dollar company. Among many other roles, he led pioneering work in fault-tolerant, shared-nothing, and scale-out relational database systems. He received his Ph.D. in Computer Science from the University of California, Berkeley, where he led the initial development of the Ingres relational database management system.

**Pat Helland**

Pat Helland has been building databases, transaction systems, distributed systems, messaging systems, multiprocessor hardware, and scalable cloud systems since 1978. At Tandem Computers, he was Chief Architect of the transaction engine for NonStop SQL. At Microsoft, he architected Microsoft Transaction Server, Distributed Transaction Coordinator, SQL Service Broker, and evolved the Cosmos big data infrastructure to include optimizing database features as well as petabyte-scale transactionally correct event processing. While at Amazon, Pat contributed to the design of the Dynamo eventually consistent store and also the Product Catalog. Pat attended the University of California, Irvine from 1973–1976 and was in the inaugural UC Irvine Information and Computer Science Hall of Fame. Pat chairs the Dean’s Leadership Council of the Donald Bren School of Information and Computer Sciences (ICS), UC Irvine.
Joseph M. Hellerstein

Joseph M. Hellerstein is the Jim Gray Professor of Computer Science at the University of California, Berkeley, whose work focuses on data-centric systems and the way they drive computing. He is an ACM Fellow, an Alfred P. Sloan Research Fellow, and the recipient of three ACM-SIGMOD “Test of Time” awards for his research. In 2010, Fortune Magazine included him in their list of 50 smartest people in technology, and MIT’s Technology Review magazine included his work on their TR10 list of the 10 technologies “most likely to change our world.” Hellerstein is the co-founder and Chief Strategy Officer of Trifacta, a software vendor providing intelligent interactive solutions to the messy problem of wrangling data. He serves on the technical advisory boards of a number of computing and Internet companies including Dell EMC, SurveyMonkey, Captricity, and Datometry, and previously served as the Director of Intel Research, Berkeley.

Wei Hong

Wei Hong is an engineering director in Google’s Data Infrastructure and Analysis (DIA) group, responsible for the streaming data processing area including building and maintaining the infrastructure for some of Google’s most revenue-critical data pipelines in Ads and Commerce. Prior to joining Google, he co-founded and led three startup companies: Illustra and Cohera with Mike Stonebraker in database systems and Arch Rock in Internet of Things. He also held senior engineering leadership positions at Informix, PeopleSoft, Cisco, and Nest. He was a senior researcher at Intel Research Berkeley working on sensor networks and streaming database systems and won an ACM SIGMOD Test of Time Award. He is a co-inventor of 80 patents. He received his Ph.D. from UC Berkeley and his ME, BE, and BS from Tsinghua University.
John Hugg

John Hugg has had a deep love for problems relating to data. He’s worked at three database product startups and worked on database problems within larger organizations as well. Although John dabbled in statistics in graduate school, Dr. Stonebraker lured him back to databases using the nascent VoltDB project. Working with the very special VoltDB team was an unmatched opportunity to learn and be challenged. John received an M.S in 2007 and a B.S in 2005 from Tufts University.

Ihab Ilyas

Ihab Ilyas is a professor in the Cheriton School of Computer Science at the University of Waterloo, where his main research focuses on the areas of big data and database systems, with special interest in data quality and integration, managing uncertain data, rank-aware query processing, and information extraction. Ihab is also a co-founder of Tamr, a startup focusing on large-scale data integration and cleaning. He is a recipient of the Ontario Early Researcher Award (2009), a Cheriton Faculty Fellowship (2013), an NSERC Discovery Accelerator Award (2014), and a Google Faculty Award (2014), and he is an ACM Distinguished Scientist. Ihab is an elected member of the VLDB Endowment board of trustees, elected SIGMOD vice chair, and an associate editor of ACM Transactions on Database Systems (TODS). He holds a Ph.D. in Computer Science from Purdue University and a B.Sc. and an M.Sc. from Alexandria University.
Jason Kinchen

Jason Kinchen, Paradigm4’s V.P. of Engineering, is a software professional with over 30 years’ experience in delivering highly complex products to life science, automotive, aerospace, and other engineering markets. He is an expert in leading technical teams in all facets of a project life cycle from feasibility analysis to requirements to functional design to delivery and enhancement, and experienced in developing quality-driven processes improving the software development life cycle and driving strategic planning. Jason is an avid cyclist and a Red Cross disaster action team volunteer.

Moshe Tov Kreps

Moshe Tov Kreps (formerly known as Peter Kreps) is a former researcher at the University of California at Berkeley and the Lawrence Berkeley National Laboratory. He was coauthor, with Mike Stonebraker, Eugene Wong, and Gerald Held, of the seminal paper, “The Design and Implementation of INGRES,” published in the ACM Transactions on Database Systems in September 1976.

Edmond Lau

Edmond Lau is the co-founder of Co Leadership, where his mission is to transform engineers into leaders. He runs leadership experiences, multi-week programs, and online courses to bridge people from where they are to the lives and careers they dream of. He’s the author of The Effective Engineer, the now the de facto onboarding guide for many engineering teams. He’s spent his career leading engineering teams across Silicon Valley at Quip, Quora, Google, and Ooyala. As a leadership coach, Edmond also works directly with CTO’s, directors, managers, and other emerging leaders to unlock what’s possible for them. Edmond has been featured in the New York Times, Forbes, Time, Slate, Inc., Fortune, and Wired. He blogs at coleadership.com, has a website (www.theeffectiveengineer.com), and tweets at @edmondlau.
Shilpa Lawande

Shilpa Lawande is CEO and co-founder of postscript.us, an AI startup on a mission to free doctors from clinical paperwork. Previously, she was VP/GM HPE Big Data Platform, including its flagship Vertica Analytics Platform. Shilpa was a founding engineer at Vertica and led its Engineering and Customer Success teams from startup through the company’s acquisition by HP. Shilpa has several patents and books on data warehousing to her name, and was named to the 2012 Mass High Tech Women to Watch list and Rev Boston 20 in 2015. Shilpa serves as an advisor at Tamr, and as mentor/volunteer at two educational initiatives, Year Up (Boston) and CSPathshala (India). Shilpa has a M.S. in Computer Science from the University of Wisconsin-Madison and a B.S in Computer Science and Engineering from the Indian Institute of Technology, Mumbai.

Amerson Lin

Amerson Lin received his B.S. and M.Eng both in Computer Science at MIT, the latter in 2005. He returned to Singapore to serve in the military and government before returning to the world of software. He was a consultant at Pivotal and then a business development lead at Palantir in both Singapore and the U.S. Amerson currently runs his own Insurtech startup—Gigacover—which delivers digital insurance to Southeast Asia.
Samuel Madden

Samuel Madden is a professor of Electrical Engineering and Computer Science in MIT’s Computer Science and Artificial Intelligence Laboratory. His research interests include databases, distributed computing, and networking. He is known for his work on sensor networks, column-oriented database, high-performance transaction processing, and cloud databases. Madden received his Ph.D. in 2003 from the University of California at Berkeley, where he worked on the TinyDB system for data collection from sensor networks. Madden was named one of Technology Review’s Top 35 Under 35 (2005), and is the recipient of several awards, including an NSF CAREER Award (2004), a Sloan Foundation Fellowship (2007), VLDB best paper awards (2004, 2007), and a MobiCom 2006 best paper award. He also received “test of time” awards in SIGMOD 2013 and 2017 (for his work on Acquisitional Query Processing in SIGMOD 2003 and on Fault Tolerance in the Borealis system in SIGMOD 2007), and a ten-year best paper award in VLDB 2015 (for his work on the C-Store system).

Tim Mattson

Tim Mattson is a parallel programmer. He earned his Ph.D. in Chemistry from the University of California, Santa Cruz for his work in molecular scattering theory. Since 1993, Tim has been with Intel Corporation, where he has worked on High Performance Computing: both software (OpenMP, OpenCL, RCCE, and OCR) and hardware/software co-design (ASCI Red, 80-core TFLOP chip, and the 48 core SCC). Tim’s academic collaborations include work on the fundamental design patterns of parallel programming, the BigDAWG polystore system, the TileDB array storage manager, and building blocks for graphs “in the language of linear algebra” (the GraphBLAS). Currently, he leads a team of researchers at Intel working on technologies that help application programmers write highly optimized code that runs on future parallel systems. Outside of computing, Tim fills his time with coastal sea kayaking. He is an ACA-certified kayaking coach (level 5, advanced open ocean) and instructor trainer (level three, basic coastal).
Felix Naumann

Felix Naumann studied Mathematics, Economics, and Computer Science at the University of Technology in Berlin. He completed his Ph.D. thesis on “Quality-driven Query Answering” in 2000. In 2001 and 2002, he worked at the IBM Almaden Research Center on topics of data integration. From 2003–2006, he was assistant professor for information integration at the Humboldt-University of Berlin. Since then, he has held the chair for information systems at the Hasso Plattner Institute at the University of Potsdam in Germany. He is Editor-in-Chief of Information Systems, and his research interests are in data profiling, data cleansing, and text mining.

Mike Olson

Mike Olson co-founded Cloudera in 2008 and served as its CEO until 2013 when he took on his current role of chief strategy officer (CSO). As CSO, Mike is responsible for Cloudera’s product strategy, open-source leadership, engineering alignment, and direct engagement with customers. Prior to Cloudera, Mike was CEO of Sleepycat Software, makers of Berkeley DB, the open-source embedded database engine. Mike spent two years at Oracle Corporation as Vice President for Embedded Technologies after Oracle’s acquisition of Sleepycat in 2006. Prior to joining Sleepycat, Mike held technical and business positions at database vendors Britton Lee, Illustra Information Technologies, and Informix Software. Mike has a B.S. and an M.S. in Computer Science from the University of California, Berkeley. Mike tweets at @mikeolson.

Elizabeth O’Neil

Elizabeth O’Neil (Betty) is a Professor of Computer Science at the University of Massachusetts, Boston. Her focus is research, teaching, and software development in database engines: performance analysis, transactions, XML support, Unicode support, buffering methods. In addition to her work for UMass Boston, she was, among other pursuits, a long-term (1977–1996) part-time Senior Scientist for Bolt, Beranek, and Newman, Inc., and during two sabbaticals was a full-time consultant for Microsoft Corporation. She is the owner of two patents owned by Microsoft.
Patrick O’Neil

Patrick O’Neil is Professor Emeritus at the University of Massachusetts, Boston. His research has focused on database system cost-performance, transaction isolation, data warehousing, variations of bitmap indexing, and multi-dimensional databases/OLAP. In addition to his research, teaching, and service activities, he is the coauthor—with his wife Elizabeth (Betty)—of a database management textbook, and has been active in developing database performance benchmarks and corporate database consulting. He holds several patents.

Mourad Ouzzani

Mourad Ouzzani is a principal scientist with the Qatar Computing Research Institute, HBKU. Before joining QCRI, he was a research associate professor at Purdue University. His current research interests include data integration, data cleaning, and building large-scale systems to enable science and engineering. He is the lead PI of Rayyan, a system for supporting the creation of systematic reviews, which had more than 11,000 users as of March 2017. He has extensively published in top-tier venues including SIGMOD, PVLDB, ICDE, and TKDE. He received Purdue University Seed for Success Awards in 2009 and 2012. He received his Ph.D. from Virginia Tech and his M.S. and B.S. from USTHB, Algeria.

Andy Palmer

Andy Palmer is co-founder and CEO of Tamr, Inc., the enterprise-scale data unification company that he founded with fellow serial entrepreneur and 2014 Turing Award winner Michael Stonebraker, Ph.D., and others. Previously, Palmer was co-founder and founding CEO of Vertica Systems (also with Mike Stonebraker), a pioneering analytics database company (acquired by HP). He founded Koa Labs, a seed fund supporting the Boston/Cambridge entrepreneurial ecosystem, is a founder-partner at The Founder Collective, and holds a research affiliate position at MIT CSAIL. During his career as an entrepreneur, Palmer has served as Founder, founding investor, BoD member, or advisor to more than 60 startup companies in technology, healthcare, and the...
livescience. He also served as Global Head of Software and Data Engineering at Novartis Institutes for BioMedical Research (NIBR) and as a member of the start-up team and Chief Information and Administrative Officer at Infinity Pharmaceuticals (NASDAQ: INFI). Previously, he held positions at innovative technology companies Bowstreet, pcOrder.com, and Trilogy. He holds a BA from Bowdoin (1988) and an MBA from the Tuck School of Business at Dartmouth (1994).

Andy Pavlo

Andy Pavlo is an assistant professor of Databaseology in the Computer Science Department at Carnegie Mellon University. He also used to raise clams. Andy received a Ph.D. in 2013 and an M.Sc. in 2009, both from Brown University, and an M.Sc. in 2006 and a B.Sc., both from Rochester Institute of Technology.

Alex Poliakov

Alex Poliakov has over a decade of experience developing distributed database internals. At Paradigm4, he helps set the vision for the SciDB product and leads a team of Customer Solutions experts who help researchers in scientific and commercial applications make optimal use of SciDB to create new insights, products, and services for their companies. Alex previously worked at Netezza, after graduating from MIT’s Course 6. Alex is into flying drones and producing drone videos.
Alexander Rasin

Alexander Rasin is an Associate Professor in the College of Computing and Digital Media (CDM) at DePaul University. He received his Ph.D. and M.Sc. in Computer Science from Brown University, Providence, RI. He is a co-Director of Data Systems and Optimization Lab at CDM and his primary research interest is in database forensics and cybersecurity applications of forensic analysis. Dr. Rasin's other research projects focus on building and tuning performance of domain-specific data management systems—currently in the areas of computer-aided diagnosis and software analytics. Several of his current research projects are supported by NSF.

Jennie Rogers

Jennie Rogers is the Lisa Wissner-Slivka and Benjamin Slivka Junior Professor in Computer Science and an Assistant Professor at Northwestern University. Before that she was a postdoctoral associate in the Database Group at MIT CSAIL where she worked with Mike Stonebraker and Sam Madden. She received her Ph.D. from Brown University under the guidance of Ugur Çetintemel. Her research interests include the management of science data, federated databases, cloud computing, and database performance modeling. Her Erdős number is 3.
Lawrence A. Rowe

Lawrence A. Rowe is an Emeritus Professor of Electrical Engineering and Computer Science at U.C. Berkeley. His research interests are software systems and applications. His group developed the Berkeley Lecture Webcasting System that produced 30 course lecture webcasts each week viewed by over 500,000 people per month. His publications received three “best paper” and two “test of time” awards. He is an investor/advisor in The Batchery a Berkeley-based seed-stage incubator. Rowe is an ACM Fellow, a co-recipient of the 2002 U.C. Technology Leadership Council Award for IT Innovation, the recipient of the 2007 U.C. Irvine Donald Bren School of ICS Distinguished Alumni Award, the 2009 recipient of the ACM SIGMM Technical Achievement Award, and a co-recipient of the Inaugural ACM SIGMOD Systems Award for the development of modern object-relational DBMS. Larry and his wife Jean produce and sell award-winning premium wines using Napa Valley grapes under the Greyscale Wines brand.

Kriti Sen Sharma

Kriti Sen Sharma is a Customer Solutions Architect at Paradigm4. He works on projects spanning multiple domains (genomics, imaging, wearables, finance, etc.). Using his skills in collaborative problem-solving, algorithm development, and programming, he builds end-to-end applications that address customers’ big-data needs and enable them to gain business insights rapidly. Kriti is an avid blogger and also loves biking and hiking. Kriti received a Ph.D. in 2013 and an M.Sc. in 2009, both from Virginia Polytechnic Institute and State University, and an a B.Tech. from Indian Institute of Technology, Kharagpur, in 2005.
Nan Tang

Nan Tang is a senior scientist at Qatar Computing Research Institute, HBKU, Qatar Foundation, Qatar. He received his Ph.D. from the Chinese University of Hong Kong in 2007. He worked as a research staff member at CWI, the Netherlands, from 2008–2010. He was a research fellow at University of Edinburgh from 2010–2012. His current research interests include data curation, data visualization, and intelligent and immersive data analytics.

Jo Tango

Jo Tango founded Kepha Partners. He has invested in the e-commerce, search engine, Internet ad network, wireless, supply chain software, storage, database, security, on-line payments, and data center virtualization spaces. He has been a founding investor in many Stonebraker companies: Goby (acquired by NAVTEQ), Paradigm4, StreamBase Systems (acquired by TIBCO), Vertica Systems (acquired by Hewlett-Packard), and VoltDB. Jo previously was at Highland Capital Partners for nearly nine years, where he was a General Partner. He also spent five years with Bain & Company, where he was based in Singapore, Hong Kong, and Boston, and focused on technology and startup projects. Jo attended Yale University (B.A., summa cum laude and Phi Beta Kappa) and Harvard Business School (M.B.A., Baker Scholar). He writes a personal blog at jtan-goVC.com.
Nesime Tatbul

Nesime Tatbul is a senior research scientist at the Intel Science and Technology Center at MIT CSAIL. Before joining Intel Labs, she was a faculty member at the Computer Science Department of ETH Zurich. She received her B.S. and M.S. in Computer Engineering from the Middle East Technical University (METU) and her M.S. and Ph.D. in Computer Science from Brown University. Her primary research area is database systems. She is the recipient of an IBM Faculty Award in 2008, a Best System Demonstration Award at SIGMOD 2005, and the Best Poster and the Grand Challenge awards at DEBS 2011. She has served on the organization and program committees for various conferences including SIGMOD (as an industrial program co-chair in 2014 and a group leader in 2011), VLDB, and ICDE (as a PC track chair for Streams, Sensor Networks, and Complex Event Processing in 2013).

Nga Tran

Nga Tran is currently the Director of Engineering in the server development team at Vertica, where she has worked for the last 14 years. Previously, she was a Ph.D. candidate at Brandeis University, where she participated in research that contributed to Mike Stonebraker’s research.

Marianne Winslett

Marianne Winslett has been a professor in the Department of Computer Science at the University of Illinois since 1987, and served as the Director of Illinois’s research center in Singapore, the Advanced Digital Sciences Center, from 2009–2013. Her research interests lie in information management and security, from the infrastructure level on up to the application level. She is an ACM Fellow and the recipient of a Presidential Young Investigator Award from the U.S. National Science Foundation. She is the former Vice-Chair of ACM SIGMOD and the former co-Editor-in-Chief of ACM Transactions on the Web, and has served on the editorial boards of ACM Transactions on Database Systems, IEEE
Transactions on Knowledge and Data Engineering, ACM Transactions on Information and System Security, The Very Large Data Bases Journal, and ACM Transactions on the Web. She has received two best paper awards for research on managing regulatory compliance data (VLDB, SSS), one best paper award for research on analyzing browser extensions to detect security vulnerabilities (USENIX Security), and one for keyword search (ICDE). Her Ph.D. is from Stanford University.

**Eugene Wong**

Eugene Wong is Professor Emeritus at the University of California, Berkeley. His distinguished career includes contributions to academia, business, and public service. As Department Chair of EECS, he led the department through its greatest period of growth and into one of the highest ranked departments in its field. In 2004, the Wireless Foundation was established in Cory Hall upon completion of the Eugene and Joan C. Wong Center for Communications Research. He authored or co-authored over 100 scholarly articles and published 4 books, mentored students, and supervised over 20 dissertations. In 1980, he co-founded (with Michael Stonebraker and Lawrence A. Rowe) the INGRES Corporation. He was the Associate Director of the Office of Science and Technology Policy, under George H. Bush; from 1994–1996, he was Vice President for Research and Development for Hong Kong University of Science and Technology. He received the ACM Software System Award in 1988 for his work on INGRES, and was awarded the 2005 IEEE Founders Medal, with the apt citation: “For leadership in national and international engineering research and technology policy, for pioneering contributions in relational databases.”
Biographies

Stan Zdonik

Stan Zdonik is a tenured professor of Computer Science at Brown University and a noted researcher in database management systems. Much of his work involves applying data management techniques to novel database architectures, to enable new applications. He is co-developer of the Aurora and Borealis stream processing engines, C-Store column store DBMS, and H-Store NewSQL DBMS, and has contributed to other systems including SciDB and the BigDAWG polystore system. He co-founded (with Michael Stonebraker) two startup companies: StreamBase Systems and Vertica Systems. Earlier, while at Bolt Beranek and Newman Inc., Dr. Zdonik worked on the Prophet System, a data management tool for pharmacologists. He has more than 150 peer-reviewed papers in the database field and was named an ACM Fellow in 2006. Dr. Zdonik has a B.S in Computer Science and one in Industrial Management, an M.S. in Computer Science, and the degree of Electrical Engineer, all from MIT, where he went on to receive his Ph.D. in database management under Prof. Michael Hammer.