C-Store: A Column-Oriented DBMS

Mike Stonebraker (MIT CSAIL), **Daniel J. Abadi** (MIT CSAIL),
Adam Batkin (Brandeis University), **Xuedong Chen** (UMass Boston),
Mitch Cherniack (Brandeis University), **Miguel Ferreira** (MIT CSAIL),
Edmond Lau (MIT CSAIL), **Amerson Lin** (MIT CSAIL),
Sam Madden (MIT CSAIL), **Elizabeth O'Neil** (UMass Boston),
Pat O'Neil (UMass Boston), **Alex Rasin** (Brown University),
Nga Tran (Brandeis University), **Stan Zdonik** (Brown University)

Abstract

This paper presents the design of a read-optimized relational DBMS that contrasts sharply with most current systems, which are write-optimized. Among the many differences in its design are: storage of data by column rather than by row, careful coding and packing of objects into storage including main memory during query processing, storing an overlapping collection of column-oriented projections, rather than the current fare of tables and indexes, a non-traditional implementation of transactions which includes high availability and snapshot isolation for read-only transactions, and the extensive use of bitmap indexes to complement B-tree structures.

We present preliminary performance data on a subset of TPC-H and show that the system we are building, C-Store, is substantially faster than popular commercial products. Hence, the architecture looks very encouraging.

1 Introduction

Most major DBMS vendors implement record-oriented storage systems, where the attributes of a record (or tuple) are placed contiguously in storage. With this *row store* architecture, a single disk write suffices to push all of the fields of a single record out to disk. Hence, high performance writes are achieved, and we call a DBMS with a row store architecture a *write-optimized* system. These are especially effective on OLTP-style applications.

In contrast, systems oriented toward ad-hoc querying of large amounts of data should be *read-optimized*. Data warehouses represent one class of read-optimized system, in which periodically a bulk load of new data is performed, followed by a relatively long period of ad-hoc queries. Other read-mostly applications include customer relationship management (CRM) systems, electronic library card catalogs, and other ad-hoc inquiry systems. In such environments, a *column store* architecture, in which the values for each single column (or attribute) are stored contiguously, should be more efficient. This efficiency has been demonstrated in the warehouse marketplace by products like Sybase IQ [FREN95, SYBA04], Addamark [ADDA04], and KDB [KDB04]. In this paper, we discuss the design of a column store called C-Store that includes a number of novel features relative to existing systems.

With a column store architecture, a DBMS need only read the values of columns required for processing a given query, and can avoid bringing into memory irrelevant attributes. In warehouse environments where typical queries involve aggregates performed over large numbers of data items, a column store has a sizeable performance advantage. However, there are several other major distinctions that can be drawn between an architecture that is read-optimized and one that is write-optimized.

Current relational DBMSs were designed to pad attributes to byte or word boundaries and to store values in their native data format. It was thought that it was too expensive to shift data values onto byte or word boundaries in main memory for processing. However, CPUs are getting faster at a much greater rate than disk bandwidth is increasing. Hence, it makes sense to trade CPU cycles, which are abundant, for disk bandwidth, which is not. This tradeoff appears especially profitable in a read-mostly environment.

There are two ways a column store can use CPU cycles to save disk bandwidth. First, it can code data elements into a more compact form. For example, if one is storing an attribute that is a customer's state of residence, then US states can be coded into six bits, whereas the two-character abbreviation requires 16 bits and a variable length character string for the name of the state requires many more. Second, one should *densepack* values in storage. For example, in a column store it is straightforward to pack N values, each K bits long, into N * K bits. The coding and compressibility advantages of a column store over a row store have been previously pointed out in [FREN95]. Of course, it is also desirable to have the DBMS query executor operate on the compressed representation whenever possible to avoid the cost of decompression, at least until values need to be presented to an application.

Commercial relational DBMSs store complete tuples of tabular data along with auxiliary B-tree indexes on attributes in the table. Such indexes can be *primary*, whereby the rows of the table are stored in as close to sorted order on the specified attribute as possible, or *secondary*, in which case no attempt is made to keep the underlying records in order on the indexed attribute. Such indexes are effective in an OLTP write-optimized environment but do not perform well in a read-optimized world. In the latter case, other data structures are advantageous, including bit map indexes [ONEI97], cross table indexes [ORAC04], and materialized views [CERI91]. In a read-optimized DBMS one can explore storing data using only these read-optimized structures, and not support write-optimized ones at all.

Hence, C-Store physically stores a collection of columns, each sorted on some attribute(s). Groups of columns sorted on the same attribute are referred to as "projections"; the same column may exist in multiple projections, possibly sorted on a different attribute in each. We expect that our aggressive compression techniques will allow us to support many column sort-orders without an explosion in space. The existence of multiple sort-orders opens opportunities for optimization.

Clearly, collections of off-the-shelf "blade" or "grid" computers will be the cheapest hardware architecture for computing and storage intensive applications such as DBMSs [DEWI92]. Hence, any new DBMS architecture should assume a grid environment in which there are G nodes (computers), each with private disk and private memory. We propose to horizontally partition data across the disks of the various nodes in a "shared nothing" architecture [STON86]. Grid computers in the near future may have tens to hundreds of nodes, and any new system should be architected for grids of this size. Of course, the nodes of a grid computer may be physically co-located or divided into clusters of co-located nodes. Since database administrators are hard pressed to optimize a grid environment, it is essential to

allocate data structures to grid nodes automatically. In addition, intra-query parallelism is facilitated by horizontal partitioning of stored data structures, and we follow the lead of Gamma [DEWI90] in implementing this construct.

Many warehouse systems (e.g. Walmart [WEST00]) maintain two copies of their data because the cost of recovery via DBMS log processing on a very large (terabyte) data set is prohibitive. This option is rendered increasingly attractive by the declining cost per byte of disks. A grid environment allows one to store such replicas on different processing nodes, thereby supporting a Tandem-style highly-available system [TAND89]. However, there is no requirement that one store multiple copies in the exact same way. C-Store allows redundant objects to be stored in different sort orders providing higher retrieval performance in addition to high availability. In general, storing overlapping projections further improves performance, as long as redundancy is crafted so that all data can be accessed even if one of the G sites fails. We call a system that tolerates K failures *K-safe*. C-Store will be configurable to support a range of values of K.

It is clearly essential to perform transactional updates, even in a read-mostly environment. Warehouses have a need to perform on-line updates to correct errors. As well, there is an increasing push toward real-time warehouses, where the delay to data visibility shrinks toward zero. The ultimate desire is on-line update to data warehouses. Obviously, in read-mostly worlds like CRM, one needs to perform general on-line updates.

There is a tension between providing updates and optimizing data structures for reading. For example, in KDB and Addamark, columns of data are maintained in entry sequence order. This allows efficient insertion of new data items, either in batch or transactionally, at the end of the column. However, the cost is a less-than-optimal retrieval structure, because most query workloads will run faster with the data in some other order. However, storing columns in non-entry sequence will make insertions very difficult and expensive.

C-Store approaches this dilemma from a fresh perspective. Specifically, we combine in a single piece of system software, both a read-optimized column store and an update/insert-oriented writeable store, connected by a *tuple mover*, as noted in Figure 1. At the top level, there is a small Writeable Store (WS) component, which is architected to support high performance inserts and updates. There is also a much larger component called the Read-optimized Store (RS), which is capable of supporting very large amounts of information. RS, as the name implies, is optimized for read and supports only a very restricted form of insert, namely the batch movement of records from WS to RS, a task that is performed by the tuple mover of Figure 1.

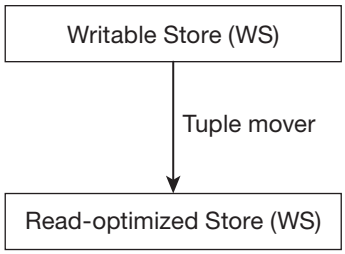


Figure 1 Architecture of C-Store

Of course, queries must access data in both storage systems. Inserts are sent to WS, while deletes must be marked in RS for later purging by the tuple mover. Updates are implemented as an insert and a delete. In order to support a high-speed tuple mover, we use a variant of the LSM-tree concept [ONEI96], which supports a *merge out* process that moves tuples from WS to RS in bulk by an efficient method of merging ordered WS data objects with large RS blocks, resulting in a new copy of RS that is installed when the operation completes.

The architecture of Figure 1 must support transactions in an environment of many large ad-hoc queries, smaller update transactions, and perhaps continuous inserts. Obviously, blindly supporting dynamic locking will result in substantial read-write conflict and performance degradation due to blocking and deadlocks.

Instead, we expect read-only queries to be run in *historical* mode. In this mode, the query selects a timestamp, T, less than the one of the most recently committed transactions, and the query is semantically guaranteed to produce the correct answer as of that point in history. Providing such *snapshot isolation* [BERE95] requires C-Store to timestamp data elements as they are inserted and to have careful programming of the runtime system to ignore elements with timestamps later than T.

Lastly, most commercial optimizers and executors are row-oriented, obviously built for the prevalent row stores in the marketplace. Since both RS and WS are column-oriented, it makes sense to build a column-oriented optimizer and executor. As will be seen, this software looks nothing like the traditional designs prevalent today.

In this paper, we sketch the design of our updatable column store, C-Store, that can simultaneously achieve very high performance on warehouse-style queries and achieve reasonable speed on OLTP-style transactions. C-Store is a column-oriented DBMS that is architected to reduce the number of disk accesses per query. The innovative features of C-Store include:

1. A hybrid architecture with a WS component optimized for frequent insert and update and an RS component optimized for query performance.
2. Redundant storage of elements of a table in several overlapping projections in different orders, so that a query can be solved using the most advantageous projection.
3. Heavily compressed columns using one of several coding schemes.
4. A column-oriented optimizer and executor, with different primitives than in a row-oriented system.
5. High availability and improved performance through K-safety using a sufficient number of overlapping projections.
6. The use of snapshot isolation to avoid 2PC and locking for queries.

It should be emphasized that while many of these topics have parallels with things that have been studied in isolation in the past, it is their combination in a real system that make C-Store interesting and unique.

The rest of this paper is organized as follows. In Section 2 we present the data model implemented by C-Store. We explore in Section 3 the design of the RS portion of C-Store, followed in Section 4 by the WS component. In Section 5 we consider the allocation of C-Store data structures to nodes in a grid, followed by a presentation of C-Store updates and transactions in Section 6. Section 7 treats the tuple mover component of C-Store, and Section 8 presents the query optimizer and executor. In Section 9 we present a comparison of C-Store performance to that achieved by both a popular commercial row store and a popular commercial column store. On TPC-H style queries, C-Store is significantly faster than either alternate system. However, it must be noted that the performance comparison is not fully completed; we have not fully integrated the WS and tuple mover, whose overhead may be significant. Finally, Sections 10 and 11 discuss related previous work and our conclusions.

2 Data Model

C-Store supports the standard relational *logical data model*, where a database consists of a collection of named tables, each with a named collection of attributes (columns). As in most relational systems, attributes (or collections of attributes) in C-Store tables can form a unique *primary key* or be a *foreign key* that references a primary key in another table. The C-Store query language is assumed to be SQL, with standard SQL semantics. Data in C-Store is not physically stored using this logical data model. Whereas most row stores implement physical tables directly and then add various indexes to speed access, C-Store implements only *projections*.

Table 1 Sample EMP data

Name	Age	Dept	Salary
Bob	25	Math	10K
Bill	27	EECS	50K
Jill	24	Biology	80K

Specifically, a C-Store projection is *anchored* on a given logical table, T, and contains one or more attributes from this table. In addition, a projection can contain any number of other attributes from other tables, as long as there is a sequence of n:1 (*i.e.*, foreign key) relationships from the anchor table to the table containing an attribute.

To form a projection, we project the attributes of interest from T, retaining any duplicate rows, and perform the appropriate sequence of value-based foreign-key joins to obtain the attributes from the non-anchor table(s). Hence, a projection has the same number of rows as its anchor table. Of course, much more elaborate projections could be allowed, but we believe this simple scheme will meet our needs while ensuring high performance. We note that we use the term projection slightly differently than is common practice, as we do not store the base table(s) from which the projection is derived.

We denote the *ith* projection over table *t* as *ti*, followed by the names of the fields in the projection. Attributes from other tables are prepended with the name of the logical table they come from. In this section, we consider an example for the standard EMP(name, age, salary, dept) and DEPT(dname, floor) relations. Sample EMP data is shown in Table 1. One possible set of projections for these tables could be as shown in Example 1.

```
EMP1  (name, age)
EMP2  (dept, age, DEPT.floor)
EMP3  (name, salary)
DEPT1 (dname, floor)
```

Example 1: Possible projections for EMP and DEPT.

Tuples in a projection are stored column-wise. Hence, if there are K attributes in a projection, there will be K data structures, each storing a single column, each of which is sorted on the same *sort key*. The sort key can be any column or columns in the projection. Tuples in a projection are sorted on the key(s) in left to right order.

We indicate the sort order of a projection by appending the sort key to the projection separated by a vertical bar. A possible ordering for the above projections would be:

```
EMP1  (name, age| age)
EMP2  (dept, age, DEPT.floor| DEPT.floor)
EMP3  (name, salary| salary)
DEPT1 (dname, floor| floor)
```

Example 2: Projections in Example 1 with sort orders.

Lastly, every projection is *horizontally partitioned* into 1 or more *segments*, which are given a *segment identifier*, Sid, where Sid > 0. C-Store supports only value-based partitioning on the sort key of a projection. Hence, each segment of a given projection is associated with a *key range* of the sort key for the projection. Moreover, the set of all key ranges *partitions* the key space.

Clearly, to answer any SQL query in C-Store, there must be a *covering set* of projections for every table in the database such that every column in every table is stored in at least one projection. However, C-Store must also be able to reconstruct complete rows of tables from the collection of stored segments. To do this, it will need to join segments from different projections, which we accomplish using *storage keys* and *join indexes*.

Storage Keys. Each segment associates every data value of every column with a storage key, SK. Values from different columns in the same segment with matching storage keys belong to the same logical row. We refer to a row of a segment using the term *record* or *tuple*. Storage keys are numbered 1, 2, 3, . . . in RS and are not physically stored, but are inferred from a tuple's physical position in the column (see Section 3 below.) Storage keys are physically present in WS and are represented as integers, larger than the largest integer storage key for any segment in RS.

Join Indices. To reconstruct all of the records in a table T from its various projections, C-Store uses *join indexes*. If *T1* and *T2* are two projections that cover a table *T*, a join index from the M segments in *T1* to the N segments in *T2* is logically a collection of M tables, one per segment, S, of *T1* consisting of rows of the form:

```
(s: SID in T2, k: Storage Key in Segment s)
```

Here, an entry in the join index for a given tuple in a segment of *T1* contains the segment ID and storage key of the corresponding (joining) tuple in *T2*. Since all join indexes are between projections anchored at the same table, this is always a one-to-one mapping. An alternative view of a join index is that it takes *T1*, sorted in some order O, and logically resorts it into the order, O' of *T2*.

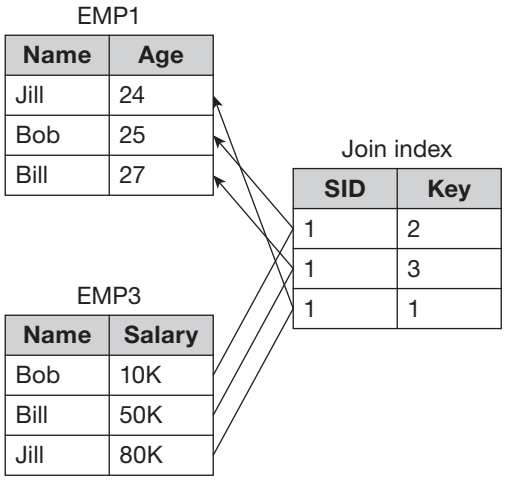


Figure 2 A join index from EMP3 to EMP1.

In order to reconstruct T from the segments of $T1, \ldots, Tk$ it must be possible to find a *path* through a set of join indices that maps each attribute of T into some sort order O*. A path is a collection of join indexes originating with a sort order specified by some projection, Ti, that passes through zero or more intermediate join indices and ends with a projection sorted in order O*. For example, to be able to reconstruct the EMP table from projections in Example 2, we need at least two join indices. If we choose age as a common sort order, we could build two indices that map EMP2 and EMP3 to the ordering of EMP1. Alternatively, we could create a join index that maps EMP2 to EMP3 and one that maps EMP3 to EMP1. Figure 2 shows a simple example of a join index that maps EMP3 to EMP1, assuming a single segment (SID = 1) for each projection. For example, the first entry of EMP3, (Bob, 10K), corresponds to the second entry of EMP1, and thus the first entry of the join index has storage key 2. In practice, we expect to store each column in several projections, thereby allowing us to maintain relatively few join indices. This is because join indexes are very expensive to store and maintain in the presence of updates, since each modification to a projection requires every join index that points into or out of it to be updated as well.

The segments of the projections in a database and their connecting join indexes must be allocated to the various nodes in a C-Store system. The C-Store administrator can optionally specify that the tables in a database must be *K-safe*. In this case,

the loss of K nodes in the grid will still allow all tables in a database to be reconstructed (i.e., despite the K failed sites, there must exist a covering set of projections and a set of join indices that map to some common sort order.) When a failure occurs, C-Store simply continues with K-1 safety until the failure is repaired and the node is brought back up to speed. We are currently working on fast algorithms to accomplish this.

Thus, the C-Store physical DBMS design problem is to determine the collection of projections, segments, sort keys, and join indices to create for the collection of logical tables in a database. This physical schema must give K-safety as well as the best overall performance for a given *training workload*, provided by the C-Store administrator, subject to requiring no more than a given *space budget*, B. Additionally, C-Store can be instructed to keep a log of all queries to be used periodically as the training workload. Because there are not enough skilled DBAs to go around, we are writing an automatic schema design tool. Similar issues are addressed in [PAPA04]

We now turn to the representation of projections, segments, storage keys, and join indexes in C-Store.

3 RS

RS is a read-optimized column store. Hence any segment of any projection is broken into its constituent columns, and each column is stored in order of the sort key for the projection. The storage key for each tuple in RS is the ordinal number of the record in the segment. This storage key is not stored but calculated as needed.

3.1 Encoding Schemes

Columns in the RS are compressed using one of 4 encodings. The encoding chosen for a column depends on its *ordering* (i.e., is the column ordered by values in that column (self-order) or by corresponding values of some other column in the same projection (foreign-order), and the proportion of *distinct values* it contains. We describe these encodings below.

Type 1: Self-order, few distinct values. A column encoded using Type 1 encoding is represented by a sequence of triples, (v, f, n) such that v is a value stored in the column, f is the position in the column where v first appears, and n is the number of times v appears in the column. For example, if a group of 4's appears in positions 12-18, this is captured by the entry, (4, 12, 7). For columns that are self-ordered, this requires one triple for each distinct value in the column. To support search queries over values in such columns, Type 1-encoded columns have clustered B-tree indexes

over their value fields. Since there are no online updates to RS, we can *densepack* the index leaving no empty space. Further, with large disk blocks (e.g., 64-128K), the height of this index can be kept small (e.g., 2 or less).

Type 2: Foreign-order, few distinct values. A column encoded using Type 2 encoding is represented by a sequence of tuples, (v, b) such that v is a value stored in the column and b is a bitmap indicating the positions in which the value is stored. For example, given a column of integers 0,0,1,1,2,1,0,2,1, we can Type 2-encode this as three pairs: (0, 110000100), (1, 001101001), and (2,000010010). Since each bitmap is sparse, it is run length encoded to save space. To efficiently find the i-th value of a type 2-encoded column, we include "offset indexes": B-trees that map positions in a column to the values contained in that column.

Type 3: Self-order, many distinct values. The idea for this scheme is to represent every value in the column as a delta from the previous value in the column. Thus, for example, a column consisting of values 1,4,7,7,8,12 would be represented by the sequence: 1,3,3,0,1,4, such that the first entry in the sequence is the first value in the column, and every subsequent entry is a delta from the previous value. Type-3 encoding is a block-oriented form of this compression scheme, such that the first entry of every block is a value in the column and its associated storage key, and every subsequent value is a delta from the previous value. This scheme is reminiscent of the way VSAM codes B-tree index keys [VSAM04]. Again, a densepack B-tree tree at the block-level can be used to index these coded objects.

Type 4: Foreign-order, many distinct values. If there are a large number of values, then it probably makes sense to leave the values unencoded. However, we are still investigating possible compression techniques for this situation. A densepack B-tree can still be used for the indexing.

3.2 Join Indexes

Join indexes must be used to connect the various projections anchored at the same table. As noted earlier, a join index is a collection of (sid, storage_key) pairs. Each of these two fields can be stored as normal columns.

There are physical database design implications concerning where to store join indexes, and we address these in the next section. In addition, join indexes must integrate RS and WS; hence, we revisit their design in the next section as well.

4 WS

In order to avoid writing two optimizers, WS is also a column store and implements the identical physical DBMS design as RS. Hence, the same projections and join indexes are present in WS. However, the storage representation is drastically different because WS must be efficiently updatable transactionally.

The storage key, SK, for each record is explicitly stored in each WS segment. A unique SK is given to each insert of a logical tuple in a table T. The execution engine must ensure that this SK is recorded in each projection that stores data for the logical tuple. This SK is an integer, larger than the number of records in the largest segment in the database.

For simplicity and scalability, WS is horizontally partitioned in the same way as RS. Hence, there is a 1:1 mapping between RS segments and WS segments. A (sid, storage_key) pair identifies a record in either of these containers.

Since we assume that WS is trivial in size relative to RS, we make no effort to compress data values; instead we represent all data directly. Therefore, each projection uses B-tree indexing to maintain a logical sort-key order.

Every column in a WS projection is represented as a collection of pairs, (v, sk), such that v is a value in the column and sk is its corresponding storage key. Each pair is represented in a conventional B-tree on the second field. The sort key(s) of each projection is additionally represented by pairs (s, sk) such that s is a sort key value and sk is the storage key describing where s first appears. Again, this structure is represented as a conventional B-tree on the sort key field(s). To perform searches using the sort key, one uses the latter B-tree to find the storage keys of interest, and then uses the former collection of B-trees to find the other fields in the record.

Join indexes can now be fully described. Every projection is represented as a collection of pairs of segments, one in WS and one in RS. For each record in the "sender," we must store the sid and storage key of a corresponding record in the "receiver." It will be useful to horizontally partition the join index in the same way as the "sending" projection and then to co-locate join index partitions with the sending segment they are associated with. In effect, each (sid, storage key) pair is a pointer to a record which can be in either the RS or WS.

5 Storage Management

The storage management issue is the allocation of segments to nodes in a grid system; C-Store will perform this operation automatically using a *storage allocator*. It seems clear that all columns in a single segment of a projection should be co-

located. As noted above, join indexes should be co-located with their "sender" segments. Also, each WS segment will be co-located with the RS segments that contain the same key range.

Using these constraints, we are working on an allocator. This system will perform initial allocation, as well as reallocation when load becomes unbalanced. The details of this software are beyond the scope of this paper.

Since everything is a column, storage is simply the persistence of a collection of columns. Our analysis shows that a raw device offers little benefit relative to today's file systems. Hence, big columns (megabytes) are stored in individual files in the underlying operating system.

6 Updates and Transactions

An insert is represented as a collection of new objects in WS, one per column per projection, plus the sort key data structure. All inserts corresponding to a single logical record have the same storage key. The storage key is allocated at the site where the update is received. To prevent C-Store nodes from needing to synchronize with each other to assign storage keys, each node maintains a locally unique counter to which it appends its local site id to generate a globally unique storage key. Keys in the WS will be consistent with RS storage keys because we set the initial value of this counter to be one larger than the largest key in RS.

We are building WS on top of BerkeleyDB [SLEE04]; we use the B-tree structures in that package to support our data structures. Hence, every insert to a projection results in a collection of physical inserts on different disk pages, one per column per projection. To avoid poor performance, we plan to utilize a very large main memory buffer pool, made affordable by the plummeting cost per byte of primary storage. As such, we expect "hot" WS data structures to be largely main memory resident.

C-Store's processing of deletes is influenced by our locking strategy. Specifically, C-Store expects large numbers of ad-hoc queries with large read sets interspersed with a smaller number of OLTP transactions covering few records. If C-Store used conventional locking, then substantial lock contention would likely be observed, leading to very poor performance.

Instead, in C-Store, we isolate read-only transactions using *snapshot isolation*. Snapshot isolation works by allowing read-only transactions to access the database as of some time in the recent past, before which we can guarantee that there are no uncommitted transactions. For this reason, when using snapshot isolation, we do not need to set any locks. We call the most recent time in the past at

which snapshot isolation can run the *high water mark* (HWM) and introduce a low-overhead mechanism for keeping track of its value in our multi-site environment. If we let read-only transactions set their effective time arbitrarily, then we would have to support general time travel, an onerously expensive task. Hence, there is also a *low water mark* (LWM) which is the earliest effective time at which a read-only transaction can run. Update transactions continue to set read and write locks and obey strict two-phase locking, as described in Section 6.2.

6.1 Providing Snapshot Isolation

The key problem in snapshot isolation is determining which of the records in WS and RS should be visible to a read-only transaction running at effective time ET. To provide snapshot isolation, we cannot perform updates in place. Instead, an update is turned into an insert and a delete. Hence, a record is visible if it was inserted before ET and deleted after ET. To make this determination without requiring a large space budget, we use coarse granularity "epochs," to be described in Section 6.1.1, as the unit for timestamps. Hence, we maintain an *insertion vector (IV)* for each projection segment in WS, which contains for each record the epoch in which the record was inserted. We program the tuple mover (described in Section 7) to ensure that no records in RS were inserted after the LWM. Hence, RS need not maintain an insertion vector. In addition, we maintain a *deleted record vector (DRV)* for each projection, which has one entry per projection record, containing a 0 if the tuple has not been deleted; otherwise, the entry contains the epoch in which the tuple was deleted. Since the DRV is very sparse (mostly zeros), it can be compactly coded using the type 2 algorithm described earlier. We store the DRV in the WS, since it must be updatable. The runtime system can now consult *IV* and *DRV* to make the visibility calculation for each query on a record-by-record basis.

6.1.1 Maintaining the High Water Mark

To maintain the HWM, we designate one site the *timestamp authority* (TA) with the responsibility of allocating timestamps to other sites. The idea is to divide time into a number of *epochs*; we define the *epoch number* to be the number of epochs that have elapsed since the beginning of time. We anticipate epochs being relatively long – e.g., many seconds each, but the exact duration may vary from deployment to deployment. We define the initial HWM to be epoch 0 and start *current epoch* at 1. Periodically, the TA decides to move the system to the next epoch; it sends a *end of epoch* message to each site, each of which increments *current epoch* from e to $e + 1$, thus causing new transactions that arrive to be run with a timestamp $e + 1$. Each site waits for all the transactions that began in epoch e (or an earlier epoch)

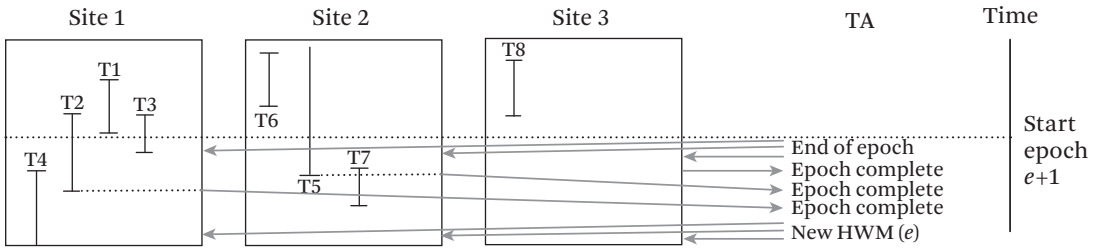


Figure 3 Illustration showing how the HWM selection algorithm works. Gray arrows indicate messages from the TA to the sites or vice versa. We can begin reading tuples with timestamp e when all transactions from epoch e have committed. Note that although T4 is still executing when the HWM is incremented, read-only transactions will not see its updates because it is running in epoch $e + 1$.

to complete and then sends an *epoch complete* message to the TA. Once the TA has received *epoch complete* messages from all sites for epoch e, it sets the HWM to be e, and sends this value to each site. Figure 3 illustrates this process.

After the TA has broadcast the new HWM with value e, read-only transactions can begin reading data from epoch e or earlier and be assured that this data has been committed. To allow users to refer to a particular real-world time when their query should start, we maintain a table mapping epoch numbers to times, and start the query as of the epoch nearest to the user-specified time.

To avoid epoch numbers from growing without bound and consuming extra space, we plan to "reclaim" epochs that are no longer needed. We will do this by "wrapping" timestamps, allowing us to reuse old epoch numbers as in other protocols, e.g., TCP. In most warehouse applications, records are kept for a specific amount of time, say 2 years. Hence, we merely keep track of the oldest epoch in any DRV, and ensure that wrapping epochs through zero does not overrun.

To deal with environments for which epochs cannot effectively wrap, we have little choice but to enlarge the "wrap length" of epochs or the size of an epoch.

6.2 Locking-based Concurrency Control

Read-write transactions use strict two-phase locking for concurrency control [GRAY92]. Each site sets locks on data objects that the runtime system reads or writes, thereby implementing a distributed lock table as in most distributed databases. Standard write-ahead logging is employed for recovery purposes; we use a NO-FORCE, STEAL policy [GRAY92] but differ from the traditional implementation of logging and locking in that we only log UNDO records, performing REDO

as described in Section 6.3, and we do not use strict two-phase commit, avoiding the PREPARE phase as described in Section 6.2.1 below.

Locking can, of course, result in deadlock. We resolve deadlock via timeouts through the standard technique of aborting one of the deadlocked transactions.

6.2.1 Distributed COMMIT Processing

In C-Store, each transaction has a *master* that is responsible for assigning units of work corresponding to a transaction to the appropriate sites and determining the ultimate commit state of each transaction. The protocol differs from two-phase commit (2PC) in that no PREPARE messages are sent. When the master receives a COMMIT statement for the transaction, it waits until all workers have completed all outstanding actions and then issues a *commit* (or *abort*) message to each site. Once a site has received a commit message, it can release all locks related to the transaction and delete the UNDO log for the transaction. This protocol differs from 2PC because the master does not PREPARE the worker sites. This means it is possible for a site the master has told to commit to crash before writing any updates or log records related to a transaction to stable storage. In such cases, the failed site will recover its state, which will reflect updates from the committed transaction, from other projections on other sites in the system during recovery.

6.2.2 Transaction Rollback

When a transaction is aborted by the user or the C-Store system, it is undone by scanning backwards in the UNDO log, which contains one entry for each logical update to a segment. We use logical logging (as in ARIES [MOHA92]), since physical logging would result in many log records, due to the nature of the data structures in WS.

6.3 Recovery

As mentioned above, a crashed site recovers by running a query (copying state) from other projections. Recall that C-Store maintains K-safety; i.e. sufficient projections and join indexes are maintained, so that K sites can fail within t, the time to recover, and the system will be able to maintain transactional consistency. There are three cases to consider. If the failed site suffered no data loss, then we can bring it up to date by executing updates that will be queued for it elsewhere in the network. Since we anticipate read-mostly environments, this roll forward operation should not be onerous. Hence, recovery from the most common type of crash is straightforward. The second case to consider is a catastrophic failure which destroys both the RS and WS. In this case, we have no choice but to reconstruct both segments from

other projections and join indexes in the system. The only needed functionality is the ability to retrieve auxiliary data structures (IV, DRV) from remote sites. After restoration, the queued updates must be run as above. The third case occurs if WS is damaged but RS is intact. Since RS is written only by the tuple mover, we expect it will typically escape damage. Hence, we discuss this common case in detail below.

6.3.1 Efficiently Recovering the WS

Consider a WS segment, Sr, of a projection with a sort key K and a key range R on a recovering site r along with a collection C of other projections, $M1, \ldots, Mb$ which contain the sort key of Sr. The tuple mover guarantees that each WS segment, S, contains all tuples with an insertion timestamp later than some time $t_{lastmove}(S)$, which represents the most recent insertion time of any record in S's corresponding RS segment.

To recover, the recovering site first inspects every projection in C for a collection of columns that covers the key range K with each segment having $t_{lastmove}(S) \leq t_{lastmove}(Sr)$. If it succeeds, it can run a collection of queries of the form:

```
SELECT desired_fields,
       insertion_epoch,
       deletion_epoch
FROM recovery_segment

WHERE insertion_epoch > t_lastmove(Sr)
       AND insertion_epoch <= HWM
       AND deletion_epoch = 0
           OR deletion_epoch >= LWM
       AND sort_key in K
```

As long as the above queries return a storage key, other fields in the segment can be found by following appropriate join indexes. As long as there is a collection of segments that cover the key range of Sr, this technique will restore Sr to the current HWM. Executing queued updates will then complete the task.

On the other hand, if there is no cover with the desired property, then some of the tuples in Sr have already been moved to RS on the remote site. Although we can still query the remote site, it is challenging to identify the desired tuples without retrieving everything in RS and differencing against the local RS segment, which is obviously an expensive operation.

To efficiently handle this case, if it becomes common, we can force the tuple mover to log, for each tuple it moves, the storage key in RS that corresponds to the storage key and epoch number of the tuple before it was moved from WS. This log can be truncated to the timestamp of the oldest tuple still in the WS on any

site, since no tuples before that will ever need to be recovered. In this case, the recovering site can use a remote WS segment, S, plus the tuple mover log to solve the query above, even though $t_{lastmove}(S)$ comes after $t_{lastmove}(Sr)$.

At r, we must also reconstruct the WS portion of any join indexes that are stored locally, i.e. for which Sr is a "sender." This merely entails querying remote "receivers," which can then compute the join index as they generate tuples, transferring the WS partition of the join index along with the recovered columns.

7 Tuple Mover

The job of the tuple mover is to move blocks of tuples in a WS segment to the corresponding RS segment, updating any join indexes in the process. It operates as a background task looking for *worthy* segment pairs. When it finds one, it performs a *merge-out process*, *MOP* on this (RS, WS) segment pair.

MOP will find all records in the chosen WS segment with an insertion time at or before the LWM, and then divides them into two groups:

- Ones deleted at or before LWM. These are discarded, because the user cannot run queries as of a time when they existed.
- Ones that were not deleted, or deleted after LWM. These are moved to RS.

MOP will create a new RS segment that we name RS'. Then, it reads in blocks from columns of the RS segment, deletes any RS items with a value in the DRV less than or equal to the LWM, and merges in column values from WS. The merged data is then written out to the new RS' segment, which grows as the merge progresses. The most recent insertion time of a record in RS' becomes the segment's new $t_{lastmove}$ and is always less than or equal to the LWM. This old-master/new-master approach will be more efficient than an update-in-place strategy, since essentially all data objects will move. Also, notice that records receive new storage keys in RS', thereby requiring join index maintenance. Since RS items may also be deleted, maintenance of the DRV is also mandatory. Once RS' contains all the WS data and join indexes are modified on RS', the system cuts over from RS to RS'. The disk space used by the old RS can now be freed.

Periodically the timestamp authority sends out to each site a new LWM epoch number. Hence, LWM "chases" HWM, and the delta between them is chosen to mediate between the needs of users who want historical access and the WS space constraints.

8 C-Store Query Execution

The query optimizer will accept a SQL query and construct a query plan of execution nodes. In this section, we describe the nodes that can appear in a plan and then the architecture of the optimizer itself.

8.1 Query Operators and Plan Format

There are 10 node types and each accepts operands or produces results of type projection (`Proj`), column (`Col`), or bitstring (`Bits`). A projection is simply a set of columns with the same cardinality and ordering. A bitstring is a list of zeros and ones indicating whether the associated values are present in the record subset being described. In addition, C-Store query operators accept predicates (`Pred`), join indexes (`JI`), attribute names (`Att`), and expressions (`Exp`) as arguments.

Join indexes and bitstrings are simply special types of columns. Thus, they also can be included in projections and used as inputs to operators where appropriate.

We briefly summarize each operator below.

1. **Decompress** converts a compressed column to an uncompressed (Type 4) representation.
2. **Select** is equivalent to the selection operator of the relational algebra (σ), but rather than producing a restriction of its input, instead produces a bitstring representation of the result.
3. **Mask** accepts a bitstring `B` and projection `Cs`, and restricts `Cs` by emitting only those values whose corresponding bits in `B` are `1`.
4. **Project** equivalent to the projection operator of the relational algebra (π).
5. **Sort** sorts all columns in a projection by some subset of those columns (the *sort* columns).
6. **Aggregation Operators** compute SQL-like aggregates over a named column, and for each group identified by the values in a projection.
7. **Concat** combines one or more projections sorted in the same order into a single projection
8. **Permute** permutes a projection according to the ordering defined by a join index.
9. **Join** joins two projections according to a predicate that correlates them.
10. **Bitstring Operators** `BAnd` produces the bitwise AND of two bitstrings. `BOr` produces a bitwise OR. `BNot` produces the complement of a bitstring.

A C-Store query plan consists of a tree of the operators listed above, with access methods at the leaves and iterators serving as the interface between connected nodes. Each non-leaf plan node consumes the data produced by its children via a modified version of the standard iterator interface [GRAE93] via calls of "get_next." To reduce communication overhead (i.e., number of calls of "get_next") between plan nodes, C-Store iterators return 64K blocks from a single column. This approach preserves the benefit of using iterators (coupling data flow with control flow), while changing the granularity of data flow to better match the column-based model.

8.2 Query Optimization

We plan to use a Selinger-style [SELI79] optimizer that uses cost-based estimation for plan construction. We anticipate using a two-phase optimizer [HONG92] to limit the complexity of the plan search space. Note that query optimization in this setting differs from traditional query optimization in at least two respects: the need to consider compressed representations of data and the decisions about when to mask a projection using a bitstring.

C-Store operators have the capability to operate on both compressed and uncompressed input. As will be shown in Section 9, the ability to process compressed data is the key to the performance benefits of C-Store. An operator's execution cost (both in terms of I/O and memory buffer requirements) is dependent on the compression type of the input. For example, a `Select` over Type 2 data (foreign order/few values, stored as a delta-encoded bitmaps, with one bitmap per value) can be performed by reading only those bitmaps from disk whose values match the predicate (despite the column itself not being sorted). However, operators that take Type 2 data as input require much larger memory buffer space (one page of memory for each possible value in the column) than any of the other three types of compression. Thus, the cost model must be sensitive to the representations of input and output columns.

The major optimizer decision is which set of projections to use for a given query. Obviously, it will be time consuming to construct a plan for each possibility, and then select the best one. Our focus will be on pruning this search space. In addition, the optimizer must decide where in the plan to mask a projection according to a bitstring. For example, in some cases it is desirable to push the `Mask` early in the plan (e.g, to avoid producing a bitstring while performing selection over Type 2 compressed data) while in other cases it is best to delay masking until a point where it is possible to feed a bitstring to the next operator in the plan (e.g., `COUNT`) that can produce results solely by processing the bitstring.

9 Performance Comparison

At the present time, we have a storage engine and the executor for RS running. We have an early implementation of the WS and tuple mover; however they are not at the point where we can run experiments on them. Hence, our performance analysis is limited to read-only queries, and we are not yet in a position to report on updates. Moreover, RS does not yet support segments or multiple grid nodes. As such, we report single-site numbers. A more comprehensive performance study will be done once the other pieces of the system have been built.

Our benchmarking system is a 3.0 Ghz Pentium, running RedHat Linux, with 2 Gbytes of memory and 750 Gbytes of disk.

In the decision support (warehouse) market TPC-H is the gold standard, and we use a simplified version of this benchmark, which our current engine is capable of running. Specifically, we implement the **lineitem**, **order**, and **customer** tables as follows:

```
CREATE TABLE LINEITEM (
L_ORDERKEY INTEGER NOT NULL,
L_PARTKEY  INTEGER NOT NULL,
L_SUPPKEY  INTEGER NOT NULL,
L_LINENUMBER       INTEGER NOT NULL,
L_QUANTITY INTEGER NOT NULL,
L_EXTENDEDPRICE    INTEGER NOT NULL,
L_RETURNFLAG       CHAR(1) NOT NULL,
L_SHIPDATE INTEGER NOT NULL);

CREATE TABLE ORDERS (
O_ORDERKEY  INTEGER NOT NULL,
O_CUSTKEY   INTEGER NOT NULL,
O_ORDERDATE INTEGER NOT NULL);

CREATE TABLE CUSTOMER (
C_CUSTKEY   INTEGER NOT NULL,
C_NATIONKEY INTEGER NOT NULL);
```

We chose columns of type INTEGER and CHAR(1) to simplify the implementation. The standard data for the above table schema for TPC-H scale_10 totals 60,000,000 line items (1.8GB), and was generated by the data generator available from the TPC website.

We tested three systems and gave each of them a storage budget of 2.7 GB (roughly 1.5 times the raw data size) for all data plus indices. The three systems were C-Store as described above and two popular commercial relational DBMS systems, one that implements a row store and another that implements a column

C-Store	Row Store	Column Store
1.987 GB	4.480 GB	2.650 GB

store. In both of these systems, we turned off locking and logging. We designed the schemas for the three systems in a way to achieve the best possible performance given the above storage budget. The row-store was unable to operate within the space constraint so we gave it 4.5 GB which is what it needed to store its tables plus indices. The actual disk usage numbers are shown below. Obviously, C-Store uses 40% of the space of the row store, even though it uses redundancy and the row store does not. The main reasons are C-Store compression and absence of padding to word or block boundaries. The column store requires 30% more space than C-Store. Again, C-Store can store a redundant schema in less space because of superior compression and absence of padding.

We ran the following seven queries on each system:

Q1. *Determine the total number of lineitems shipped for each day after day D.*

```
SELECT l_shipdate, COUNT (*)
FROM lineitem
WHERE l_shipdate > D
GROUP BY l_shipdate
```

Q2. *Determine the total number of lineitems shipped for each supplier on day D.*

```
SELECT l_suppkey, COUNT (*)
FROM lineitem
WHERE l_shipdate = D
GROUP BY l_suppkey
```

Q3. *Determine the total number of lineitems shipped for each supplier after day D.*

```
SELECT l_suppkey, COUNT (*)
FROM lineitem
WHERE l_shipdate > D
GROUP BY l_suppkey
```

Q4. *For every day after D, determine the latest shipdate of all items ordered on that day.*

```
SELECT o_orderdate, MAX (l_shipdate)
FROM lineitem, orders
WHERE l_orderkey = o_orderkey AND
      o_orderdate > D
GROUP BY o_orderdate
```

Q5. *For each supplier, determine the latest shipdate of an item from an order that was made on some date, D.*

```
SELECT l_suppkey, MAX (l_shipdate)
FROM lineitem, orders
WHERE l_orderkey = o_orderkey AND
      o_orderdate = D
GROUP BY l_suppkey
```

Q6. *For each supplier, determine the latest shipdate of an item from an order made after some date, D.*

```
SELECT l_suppkey, MAX (l_shipdate)
FROM lineitem, orders
WHERE l_orderkey = o_orderkey AND
      o_orderdate > D
GROUP BY l_suppkey
```

Q7. *Return a list of identifiers for all nations represented by customers along with their total lost revenue for the parts they have returned.* This is a simplified version of query 10 (Q10) of TPC-H.

```
SELECT c_nationkey, sum(l_extendedprice)
FROM lineitem, orders, customers
WHERE l_orderkey=o_orderkey AND
      o_custkey=c_custkey AND
      l_returnflag='R'
GROUP BY c_nationkey
```

We constructed schemas for each of the three systems that best matched our seven-query workload. These schema were tuned individually for the capabilities of each system. For C-Store, we used the following schema:

```
D1: (l_orderkey, l_partkey, l_suppkey,
     l_linenumber, l_quantity,
     l_extendedprice, l_returnflag, l_shipdate
     | l_shipdate, l_suppkey)

D2: (o_orderdate, l_shipdate, l_suppkey |
     o_orderdate, l_suppkey)

D3: (o_orderdate, o_custkey, o_orderkey |
     o_orderdate)

D4: (l_returnflag, l_extendedprice,
     c_nationkey | l_returnflag)

D5: (c_custkey, c_nationkey | c_custkey)
```

D2 and D4 are materialized (join) views. D3 and D5 are added for completeness since we don't use them in any of the seven queries. They are included so that we can answer arbitrary queries on this schema as is true for the product schemas.

On the commercial row-store DBMS, we used the common relational schema given above with a collection of system-specific tuning parameters. We also used system-specific tuning parameters for the commercial column-store DBMS. Although we believe we chose good values for the commercial systems, obviously, we cannot guarantee they are optimal.

The following table indicates the performance that we observed. All measurements are in seconds and are taken on a dedicated machine.

Query	C-Store	Row Store	Column Store
Q1	0.03	6.80	2.24
Q2	0.36	1.09	0.83
Q3	4.90	93.26	29.54
Q4	2.09	722.90	22.23
Q5	0.31	116.56	0.93
Q6	8.50	652.90	32.83
Q7	2.54	265.80	33.24

As can be seen, C-Store is much faster than either commercial product. The main reasons are:

- *Column representation* – avoids reads of unused attributes (same as competing column store).
- *Storing overlapping projections, rather than the whole table* – allows storage of multiple orderings of a column as appropriate.
- *Better compression of data* – allows more orderings in the same space.
- *Query operators operate on compressed representation* – mitigates the storage barrier problem of current processors.

In order to give the other systems every possible advantage, we tried running them with the materialized views that correspond to the projections we used with C-Store. This time, the systems used space as follows (C-Store numbers, which did not change, are included as a reference):

C-Store	Row Store	Column Store
1.987 GB	11.900 GB	4.090 GB

The relative performance numbers in seconds are as follows:

Query	C-Store	Row Store	Column Store
Q1	0.03	0.22	2.34
Q2	0.36	0.81	0.83
Q3	4.90	49.38	29.10
Q4	2.09	21.76	22.23
Q5	0.31	0.70	0.63
Q6	8.50	47.38	25.46
Q7	2.54	18.47	6.28

As can be seen, the performance gap closes, but at the same time, the amount of storage needed by the two commercial systems grows quite large.

In summary, for this seven query benchmark, C-Store is on average 164 times faster than the commercial row-store and 21 times faster than the commercial column-store in the space-constrained case. For the case of unconstrained space, C-Store is 6.4 times faster than the commercial row-store, but the row-store takes 6 times the space. C-Store is on average 16.5 times faster than the commercial column-store, but the column-store requires 1.83 times the space.

Of course, this performance data is very preliminary. Once we get WS running and write a tuple mover, we will be in a better position to do an exhaustive study.

10 Related Work

One of the thrusts in the warehouse market is in maintaining so-called "data cubes." This work dates from Essbase by Arbor software in the early 1990's, which was effective at "slicing and dicing" large data sets [GRAY97]. Efficiently building and maintaining specific aggregates on stored data sets has been widely studied [KOTI99, ZHAO97]. Precomputation of such aggregates as well as more general materialized views [STAU96] is especially effective when a prespecified set of queries is run at regular intervals. On the other hand, when the workload cannot be anticipated in advance, it is difficult to decide what to precompute. C-Store is aimed entirely at this latter problem.

Including two differently architected DBMSs in a single system has been studied before in data mirrors [RAMA02]. However, the goal of data mirrors was to achieve better query performance than could be achieved by either of the two underlying systems alone in a warehouse environment. In contrast, our goal is to simultaneously achieve good performance on update workloads and ad-hoc queries. Consequently, C-Store differs dramatically from a data mirror in its design.

Storing data via columns has been implemented in several systems, including Sybase IQ, Addamark, Bubba [COPE88], Monet [BONC04], and KDB. Of these, Monet is probably closest to C-Store in design philosophy. However, these systems typically store data in entry sequence and do not have our hybrid architecture nor do they have our model of overlapping materialized projections.

Similarly, storing tables using an inverted organization is well known. Here, every attribute is stored using some sort of indexing, and record identifiers are used to find corresponding attributes in other columns. C-Store uses this sort of organization in WS but extends the architecture with RS and a tuple mover.

There has been substantial work on using compressed data in databases; Roth and Van Horn [ROTH93] provide an excellent summary of many of the techniques that have been developed. Our coding schemes are similar to some of these techniques, all of which are derived from a long history of work on the topic in the broader field of computer science [WITT87]. Our observation that it is possible to operate directly on compressed data has been made before [GRAE91, WESM00].

Lastly, materialized views, snapshot isolation, transaction management, and high availability have also been extensively studied. The contribution of C-Store is an innovative combination of these techniques that simultaneously provides improved performance, K-safety, efficient retrieval, and high performance transactions.

11 Conclusions

This paper has presented the design of C-Store, a radical departure from the architecture of current DBMSs. Unlike current commercial systems, it is aimed at the "read-mostly" DBMS market. The innovative contributions embodied in C-Store include:

- A column store representation, with an associated query execution engine.
- A hybrid architecture that allows transactions on a column store.
- A focus on economizing the storage representation on disk, by coding data values and dense-packing the data.

- A data model consisting of overlapping projections of tables, unlike the standard fare of tables, secondary indexes, and projections.
- A design optimized for a shared nothing machine environment.
- Distributed transactions without a redo log or two phase commit.
- Efficient snapshot isolation.

Acknowledgements and References

We would like to thank David DeWitt for his helpful feedback and ideas.

This work was supported by the National Science Foundation under NSF Grant numbers IIS-0086057 and IIS-0325525.

[ADDA04] http://www.addamark.com/products/sls.htm

[BERE95] Hal Berenson et al. A Critique of ANSI SQL Isolation Levels. In *Proceedings of SIGMOD*, 1995.

[BONC04] Peter Boncz et. al. MonetDB/X100: Hyper-pipelining Query Execution. In *Proceedings CIDR 2004*.

[CERI91] S. Ceri and J. Widom. Deriving Production Rules for Incremental View Maintenance. In *VLDB*, 1991.

[COPE88] George Copeland et. al. Data Placement in Bubba. *In Proceedings SIGMOD 1988*.

[DEWI90] David Dewitt et. al. The GAMMA Database machine Project. *IEEE Transactions on Knowledge and Data Engineering*, 2(1), March, 1990.

[DEWI92] David Dewitt and Jim Gray. Parallel Database Systems: The Future of High Performance Database Processing. *Communications of the ACM*, 1992.

[FREN95] Clark D. French. One Size Fits All Database Architectures Do Not Work for DSS. In *Proceedings of SIGMOD*, 1995.

[GRAE91] Goetz Graefe, Leonard D. Shapiro. Data Compression and Database Performance. In *Proceedings of the Symposium on Applied Computing*, 1991.

[GRAE93] G. Graefe. Query Evaluation Techniques for Large Databases. *Computing Surveys*, 25(2), 1993.

[GRAY92] Jim Gray and Andreas Reuter. *Transaction Processing Concepts and Techniques*, Morgan Kaufman, 1992.

[GRAY97] Gray et al. DataCube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery*, 1(1), 1997.

[HONG92] Wei Hong and Michael Stonebraker. Exploiting Interoperator Parallelism in XPRS. In *SIGMOD*, 1992.

[KDB04] http://www.kx.com/products/database.php

[KOTI99] Yannis Kotidis, Nick Roussopoulos. DynaMat: A Dynamic View Management System for Data Warehouses. In *Proceedings of SIGMOD*, 1999.

[MOHA92] C. Mohan et. al: ARIES: A Transaction Recovery Method Supporting Fine-granularity Locking and Partial Rollbacks Using Write-ahead Logging. *TODS*, March 1992.

[ONEI96] Patrick O'Neil, Edward Cheng, Dieter Gawlick, and Elizabeth O'Neil, The Log-Structured Merge-Tree. *Acta Informatica* 33, June 1996.

[ONEI97] P. O'Neil and D. Quass. Improved Query Performance with Variant Indexes, In *Proceedings of SIGMOD*, 1997.

[ORAC04] Oracle Corporation. *Oracle 9i Database for Data Warehousing and Business Intelligence*. White Paper. http://www.oracle.com/solutions/business_intelligence/Oracle9idw_bwp.

[PAPA04] Stratos Papadomanolakis and Anastassia Ailamaki. AutoPart: Automating Schema Design for Large Scientific Databases Using Data Partitioning. In *SSDBM 2004*.

[RAMA02] Ravishankar Ramamurthy, David Dewitt. Qi Su: A Case for Fractured Mirrors. In *Proceedings of VLDB*, 2002.

[ROTH93] Mark A. Roth, Scott J. Van Horn: Database Compression. *SIGMOD Record* 22(3). 1993.

[SELI79] Patricia Selinger, Morton Astrahan, Donald Chamberlain, Raymond Lorie, Thomas Price. Access Path Selection in a Relational Database. In *Proceedings of SIGMOD*, 1979.

[SLEE04] http://www.sleepycat.com/docs/

[STAU96] Martin Staudt, Matthias Jarke. Incremental Maintenance of Externally Materialized Views. In *VLDB*, 1996.

[STON86] Michael Stonebraker. The Case for Shared Nothing. In *Database Engineering*, 9(1), 1986.

[SYBA04] http://www.sybase.com/products/databaseservers/sybaseiq

[TAND89] Tandem Database Group: NonStop SQL, A Distributed High Performance, High Availability Implementation of SQL. In Proceedings *of HPTPS*, 1989.

[VSAM04] http://www.redbooks.ibm.com/redbooks.nsf/0/8280b48d5e3997bf85256cbd007e4a96?OpenDocument

[WESM00] Till Westmann, Donald Kossmann, Sven Helmer, Guido Moerkotte. The Implementation and Performance of Compressed Databases. *SIGMOD Record* 29(3), 2000.

[WEST00] Paul Westerman. *Data Warehousing: Using the Wal-Mart Model*. Morgan-Kaufmann Publishers , 2000.

[WITT87] I. Witten, R. Neal, and J. Cleary. Arithmetic coding for data compression. *Comm. of the ACM*, 30(6), June 1987.

[ZHAO97] Y. Zhao, P. Deshpande, and J. Naughton. An Array-Based Algorithm for Simultaneous Multidimensional Aggregates. In *Proceedings of SIGMOD*, 1997.

The Collected Works of Michael Stonebraker

D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, C. Erwin, E. F. Galvez, M. Hatoun, A. Maskey, A. Rasin, A. Singer, M. Stonebraker, N. Tatbul, Y. Xing, R. Yan, and S. B. Zdonik. 2003a. Aurora: A data stream management system. In *Proc. ACM SIGMOD International Conference on Management of Data*, p. 666. DOI: 10.1145/872757.872855. 225, 228, 229, 230, 232

D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. B. Zdonik. 2003b. Aurora: a new model and architecture for data stream management. *VLDB Journal*, 12(2): 120–139. DOI: 10.1007/s00778-003-0095-z. 228, 229, 324

D. J. Abadi, R. Agrawal, A. Ailamaki, M. Balazinska, P. A. Bernstein, M. J. Carey, S. Chaudhuri, J. Dean, A. Doan, M. J. Franklin, J. Gehrke, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, H. V. Jagadish, D. Kossmann, S. Madden, S. Mehrotra, T. Milo, J. F. Naughton, R. Ramakrishnan, V. Markl, C. Olston, B. C. Ooi, C. Ré, D. Suciu, M. Stonebraker, T. Walter, and J. Widom. 2014. The Beckman report on database research. *ACM SIGMOD Record*, 43(3): 61–70. DOI: 10.1145/2694428.2694441. 92

D. Abadi, R. Agrawal, A. Ailamaki, M. Balazinska, P. A. Bernstein, M. J. Carey, S. Chaudhuri, J. Dean, A. Doan, M. J. Franklin, J. Gehrke, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, H. V. Jagadish, D. Kossmann, S. Madden, S. Mehrotra, T. Milo, J. F. Naughton, R. Ramakrishnan, V. Markl, C. Olston, B. C. Ooi, C. Ré, D. Suciu, M. Stonebraker, T. Walter, and J. Widom. 2016. The Beckman report on database research. *Communications of the ACM*, 59(2): 92–99. DOI: 10.1145/2845915. 92

Z. Abedjan, C. G. Akcora, M. Ouzzani, P. Papotti, and M. Stonebraker. 2015a. Temporal rules discovery for web data cleaning. *Proc. VLDB Endowment*, 9(4): 336–347. http://www.vldb.org/pvldb/vol9/p336-abedjan.pdf. 297

Z. Abedjan, J. Morcos, M. N. Gubanov, I. F. Ilyas, M. Stonebraker, P. Papotti, and M. Ouzzani. 2015b. Dataxformer: Leveraging the web for semantic transformations. In *Proc. 7th Biennial Conference on Innovative Data Systems Research*. http://www.cidrdb.org/cidr2015/Papers/CIDR15_Paper31.pdf. 296, 297

Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. 2016a. Detecting data errors: Where are we and what needs to be done? *Proc. VLDB Endowment*, 9(12): 993–1004. http://www.vldb.org/pvldb/vol9/p993-abedjan.pdf. 298

Z. Abedjan, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, and M. Stonebraker. 2016b. Dataxformer: A robust transformation discovery system. In *Proc. 32nd International Conference on Data Engineering*, pp. 1134–1145. DOI: 10.1109/ICDE.2016.7498319. 296

S. Abiteboul, R. Agrawal, P. A. Bernstein, M. J. Carey, S. Ceri, W. B. Croft, D. J. DeWitt, M. J. Franklin, H. Garcia-Molina, D. Gawlick, J. Gray, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, M. L. Kersten, M. J. Pazzani, M. Lesk, D. Maier, J. F. Naughton, H. Schek, T. K. Sellis, A. Silberschatz, M. Stonebraker, R. T. Snodgrass, J. D. Ullman, G. Weikum, J. Widom, and S. B. Zdonik. 2003. The Lowell database research self assessment. *CoRR*, cs.DB/0310006. http://arxiv.org/abs/cs.DB/0310006. 92

S. Abiteboul, R. Agrawal, P. A. Bernstein, M. J. Carey, S. Ceri, W. B. Croft, D. J. DeWitt, M. J. Franklin, H. Garcia-Molina, D. Gawlick, J. Gray, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, M. L. Kersten, M. J. Pazzani, M. Lesk, D. Maier, J. F. Naughton, H. Schek, T. K. Sellis, A. Silberschatz, M. Stonebraker, R. T. Snodgrass, J. D. Ullman, G. Weikum, J. Widom, and S. B. Zdonik. 2005. The Lowell database research self-assessment. *Communications of the ACM*, 48(5): 111–118. DOI: 10.1145/1060710.1060718. 92

R. Agrawal, A. Ailamaki, P. A. Bernstein, E. A. Brewer, M. J. Carey, S. Chaudhuri, A. Doan, D. Florescu, M. J. Franklin, H. Garcia-Molina, J. Gehrke, L. Gruenwald, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, H. F. Korth, D. Kossmann, S. Madden, R. Magoulas, B. C. Ooi, T. O'Reilly, R. Ramakrishnan, S. Sarawagi, M. Stonebraker, A. S. Szalay, and G. Weikum. 2008. The Claremont report on database research. *ACM SIGMOD Record*, 37(3): 9–19. DOI: 10.1145/1462571.1462573. 92

R. Agrawal, A. Ailamaki, P. A. Bernstein, E. A. Brewer, M. J. Carey, S. Chaudhuri, A. Doan, D. Florescu, M. J. Franklin, H. Garcia-Molina, J. Gehrke, L. Gruenwald, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, H. F. Korth, D. Kossmann, S. Madden, R. Magoulas, B. C. Ooi, T. O'Reilly, R. Ramakrishnan, S. Sarawagi, M. Stonebraker, A. S. Szalay, and G. Weikum. 2009. The Claremont report on database research. *Communications of the ACM*, 52(6): 56–65. DOI: 10.1145/1516046.1516062. 92

A. Aiken, J. Chen, M. Lin, M. Spalding, M. Stonebraker, and A. Woodruff. 1995. The Tioga-2 database visualization environment. In *Proc. Workshop on Database Issues for Data Visualization*, pp. 181–207. DOI: 10.1007/3-540-62221-7_15.

A. Aiken, J. Chen, M. Stonebraker, and A. Woodruff. 1996. Tioga-2: A direct manipulation database visualization environment. In *Proc. 12th International Conference on Data Engineering*, pp. 208–217. DOI: 10.1109/ICDE.1996.492109.

E. Allman and M. Stonebraker. 1982. Observations on the evolution of a software system. *IEEE Computer*, 15(6): 27–32. DOI: 10.1109/MC.1982.1654047.

E. Allman, M. Stonebraker, and G. Held. 1976. Embedding a relational data sublanguage in a general purpose programming language. In *Proc. SIGPLAN Conference on Data: Abstraction, Definition and Structure*, pp. 25–35. DOI: 10.1145/800237.807115. 195

J. T. Anderson and M. Stonebraker. 1994. SEQUOIA 2000 metadata schema for satellite images. *ACM SIGMOD Record*, 23(4): 42–48. DOI: 10.1145/190627.190642.

A. Arasu, M. Cherniack, E. F. Galvez, D. Maier, A. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. 2004. Linear road: A stream data management benchmark. In *Proc. 30th International Conference on Very Large Data Bases*, pp. 480–491. http://www.vldb.org/conf/2004/RS12P1.pdf. 326

T. Atwoode, J. Dash, J. Stein, M. Stonebraker, and M. E. S. Loomis. 1994. Objects and databases (panel). In *Proc. 9th Annual Conference on Object-Oriented Programming Systems, Languages, and Applications*, pp. 371–372. DOI: 10.1145/191080.191138.

H. Balakrishnan, M. Balazinska, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, E. F. Galvez, J. Salz, M. Stonebraker, N. Tatbul, R. Tibbetts, and S. B. Zdonik. 2004. Retrospective on Aurora. *VLDB Journal*, 13(4): 370–383. DOI: 10.1007/s00778-004-0133-5. 228, 229

M. Balazinska, H. Balakrishnan, and M. Stonebraker. 2004b. Load management and high availability in the Medusa distributed stream processing system. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 929–930. DOI: 10.1145/1007568.1007701. 325

M. Balazinska, H. Balakrishnan, and M. Stonebraker. 2004a. Contract-based load management in federated distributed systems. In *Proc. 1st USENIX Symposium on Networked Systems Design and Implementation*. http://www.usenix.org/events/nsdi04/tech/balazinska.html. 228, 230

M. Balazinska, H. Balakrishnan, S. Madden, and M. Stonebraker. 2005. Fault-tolerance in the Borealis distributed stream processing system. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 13–24. DOI: 10.1145/1066157.1066160. 228, 230, 234, 325

M. Balazinska, H. Balakrishnan, S. Madden, and M. Stonebraker. 2008. Fault-tolerance in the borealis distributed stream processing system. *ACM Transactions on Database Systems*, 33(1): 3:1–3:44. DOI: 10.1145/1331904.1331907.

D. Barbará, J. A. Blakeley, D. H. Fishman, D. B. Lomet, and M. Stonebraker. 1994. The impact of database research on industrial products (panel summary). *ACM SIGMOD Record*, 23(3): 35–40. DOI: 10.1145/187436.187455.

V. Barr and M. Stonebraker. 2015a. A valuable lesson, and whither hadoop? *Communications of the ACM*, 58(1): 18–19. DOI: 10.1145/2686591. 50

V. Barr and M. Stonebraker. 2015b. How men can help women in cs; winning 'computing's nobel prize'. *Communications of the ACM*, 58(11): 10–11. DOI: 10.1145/2820419.

V. Barr, M. Stonebraker, R. C. Fernandez, D. Deng, and M. L. Brodie. 2017. How we teach cs2all, and what to do about database decay. *Communications of the ACM*, 60(1): 10–11. http://dl.acm.org/citation.cfm?id=3014349.

L. Battle, M. Stonebraker, and R. Chang. 2013. Dynamic reduction of query result sets for interactive visualizaton. In *Proc. 2013 IEEE International Conference on Big Data*, pp. 1–8. DOI: 10.1109/BigData.2013.6691708.

L. Battle, R. Chang, and M. Stonebraker. 2016. Dynamic prefetching of data tiles for interactive visualization. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 1363–1375. DOI: 10.1145/2882903.2882919.

R. Berman and M. Stonebraker. 1977. GEO-OUEL: a system for the manipulation and display of geographic data. In *Proc. 4th Annual Conference Computer Graphics and Interactive Techniques*, pp. 186–191. DOI: 10.1145/563858.563892.

P. A. Bernstein, U. Dayal, D. J. DeWitt, D. Gawlick, J. Gray, M. Jarke, B. G. Lindsay, P. C. Lockemann, D. Maier, E. J. Neuhold, A. Reuter, L. A. Rowe, H. Schek, J. W. Schmidt, M. Schrefl, and M. Stonebraker. 1989. Future directions in DBMS research—the Laguna Beach participants. *ACM SIGMOD Record*, 18(1): 17–26. 92

P. A. Bernstein, M. L. Brodie, S. Ceri, D. J. DeWitt, M. J. Franklin, H. Garcia-Molina, J. Gray, G. Held, J. M. Hellerstein, H. V. Jagadish, M. Lesk, D. Maier, J. F. Naughton, H. Pirahesh, M. Stonebraker, and J. D. Ullman. 1998a. The Asilomar report on database research. *ACM SIGMOD Record*, 27(4): 74–80. DOI: 10.1145/306101.306137. 92

P. A. Bernstein, M. L. Brodie, S. Ceri, D. J. DeWitt, M. J. Franklin, H. Garcia-Molina, J. Gray, G. Held, J. M. Hellerstein, H. V. Jagadish, M. Lesk, D. Maier, J. F. Naughton, H. Pirahesh, M. Stonebraker, and J. D. Ullman. 1998b. The Asilomar report on database research. *CoRR*, cs.DB/9811013. http://arxiv.org/abs/cs.DB/9811013. 92

A. Bhide and M. Stonebraker. 1987. Performance issues in high performance transaction processing architectures. In *Proc. 2nd International Workshop High Performance Transaction Systems*, pp. 277–300. DOI: 10.1007/3-540-51085-0_51. 91

A. Bhide and M. Stonebraker. 1988. A performance comparison of two architectures for fast transaction processing. In *Proc. 4th International Conference on Data Engineering*, pp. 536–545. DOI: 10.1109/ICDE.1988.105501. 91

M. L. Brodie and M. Stonebraker. 1993. Darwin: On the incremental migration of legacy information systems. Technical Report TR-0222-10-92-165, GTE Laboratories Incorporated.

M. L. Brodie and M. Stonebraker. 1995a. *Migrating Legacy Systems: Gateways, Interfaces, and the Incremental Approach*. Morgan Kaufmann. 91

M. L. Brodie and M. Stonebraker. 1995b. *Legacy Information Systems Migration: Gateways, Interfaces, and the Incremental Approach*. Morgan Kaufmann.

M. L. Brodie, R. M. Michael Stonebraker, and J. Pei. 2018. The case for the co-evolution of applications and data. In *New England Database Days*.

P. Brown and M. Stonebraker. 1995. Bigsur: A system for the management of earth science data. In *Proc. 21th International Conference on Very Large Data Bases*, pp. 720–728. http://www.vldb.org/conf/1995/P720.pdf.

M. J. Carey and M. Stonebraker. 1984. The performance of concurrency control algorithms for database management systems. In *Proc. 10th International Conference on Very Large Data Bases*, pp. 107–118. http://www.vldb.org/conf/1984/P107.pdf. 91, 200

D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. B. Zdonik. 2002. Monitoring streams—A new class of data management applications. In *Proc. 28th International Conference on Very Large Data Bases*, pp. 215–226. DOI: 10.1016/B978-155860869-6/50027-5. 228, 229, 324

D. Carney, U. Çetintemel, A. Rasin, S. B. Zdonik, M. Cherniack, and M. Stonebraker. 2003. Operator scheduling in a data stream manager. In *Proc. 29th International Conference on Very Large Data Bases*, pp. 838–849. http://www.vldb.org/conf/2003/papers/S25P02.pdf. 228, 229

U. Çetintemel, J. Du, T. Kraska, S. Madden, D. Maier, J. Meehan, A. Pavlo, M. Stonebraker, E. Sutherland, N. Tatbul, K. Tufte, H. Wang, and S. B. Zdonik. 2014. S-store: A streaming NewSQL system for big velocity applications. *Proc. VLDB Endowment*, 7(13): 1633–1636. http://www.vldb.org/pvldb/vol7/p1633-cetintemel.pdf. 234, 251

U. Çetintemel, D. J. Abadi, Y. Ahmad, H. Balakrishnan, M. Balazinska, M. Cherniack, J. Hwang, S. Madden, A. Maskey, A. Rasin, E. Ryvkina, M. Stonebraker, N. Tatbul, Y. Xing, and S. Zdonik. 2016. The Aurora and Borealis stream processing engines. In M. N. Garofalakis, J. Gehrke, and R. Rastogi, editors, *Data Stream Management—Processing High-Speed Data Streams*, pp. 337–359. Springer. ISBN 978-3-540-28607-3. DOI: 10.1007/978-3-540-28608-0_17.

R. Chandra, A. Segev, and M. Stonebraker. 1994. Implementing calendars and temporal rules in next generation databases. In *Proc. 10th International Conference on Data Engineering*, pp. 264–273. DOI: 10.1109/ICDE.1994.283040. 91

S. Chaudhuri, A. K. Chandra, U. Dayal, J. Gray, M. Stonebraker, G. Wiederhold, and M. Y. Vardi. 1996. Database research: Lead, follow, or get out of the way?—panel abstract. In *Proc. 12th International Conference on Data Engineering*, p. 190.

P. Chen, V. Gadepally, and M. Stonebraker. 2016. The BigDAWG monitoring framework. In *Proc. 2016 IEEE High Performance Extreme Computing Conference*, pp. 1–6. DOI: 10.1109/HPEC.2016.7761642. 373

Y. Chi, C. R. Mechoso, M. Stonebraker, K. Sklower, R. Troy, R. R. Muntz, and E. Mesrobian. 1997. ESMDIS: earth system model data information system. In *Proc. 9th International Conference on Scientific and Statistical Database Management*, pp. 116–118. DOI: 10.1109/SSDM.1997.621169.

P. Cudré-Mauroux, H. Kimura, K. Lim, J. Rogers, R. Simakov, E. Soroush, P. Velikhov, D. L. Wang, M. Balazinska, J. Becla, D. J. DeWitt, B. Heath, D. Maier, S. Madden, J. M. Patel, M. Stonebraker, and S. B. Zdonik. 2009. A demonstration of SciDB: A science-oriented DBMS. *Proc. VLDB Endowment*, 2(2): 1534–1537. DOI: 10.14778/1687553.1687584.

J. DeBrabant, A. Pavlo, S. Tu, M. Stonebraker, and S. B. Zdonik. 2013. Anti-caching: A new approach to database management system architecture. *Proc. VLDB Endowment*, 6(14): 1942–1953. http://www.vldb.org/pvldb/vol6/p1942-debrabant.pdf.

J. DeBrabant, J. Arulraj, A. Pavlo, M. Stonebraker, S. B. Zdonik, and S. Dulloor. 2014. A prolegomenon on OLTP database systems for non-volatile memory. In *Proc. 5th International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures*, pp. 57–63. http://www.adms-conf.org/2014/adms14_debrabant.pdf.

D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang. 2017a. The data civilizer system. In *Proc. 8th Biennial Conference on Innovative Data Systems Research*. http://cidrdb.org/cidr2017/papers/p44-deng-cidr17.pdf. 293

D. Deng, A. Kim, S. Madden, and M. Stonebraker. 2017b. SILKMOTH: an efficient method for finding related sets with maximum matching constraints. *CoRR*, abs/1704.04738. http://arxiv.org/abs/1704.04738.

D. J. DeWitt, R. H. Katz, F. Olken, L. D. Shapiro, M. Stonebraker, and D. A. Wood. 1984. Implementation techniques for main memory database systems. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 1–8. DOI: 10.1145/602259.602261. 111

D. J. DeWitt and M. Stonebraker. January 2008. MapReduce: A major step backwards. *The Database Column*. http://homes.cs.washington.edu/~billhowe/mapreduce_a_major_step_backwards.html. Accessed April 8, 2018. 50, 114, 136, 184, 209

D. J. DeWitt, I. F. Ilyas, J. F. Naughton, and M. Stonebraker. 2013. We are drowning in a sea of least publishable units (lpus). In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 921–922. DOI: 10.1145/2463676.2465345.

P. Dobbins, T. Dohzen, C. Grant, J. Hammer, M. Jones, D. Oliver, M. Pamuk, J. Shin, and M. Stonebraker. 2007. Morpheus 2.0: A data transformation management system. In *Proc. 3rd International Workshop on Database Interoperability*.

T. Dohzen, M. Pamuk, J. Hammer, and M. Stonebraker. 2006. Data integration through transform reuse in the Morpheus project. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 736–738. DOI: 10.1145/1142473.1142571.

J. Dozier, M. Stonebraker, and J. Frew. 1994. Sequoia 2000: A next-generation information system for the study of global change. In *Proc. 13th IEEE Symposium Mass Storage Systems*, pp. 47–56. DOI: 10.1109/MASS.1994.373028.

J. Duggan and M. Stonebraker. 2014. Incremental elasticity for array databases. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 409–420. DOI: 10.1145/2588555.2588569.

J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. B. Zdonik. 2015a. The BigDAWG polystore system. *ACM SIGMOD Record*, 44(2): 11–16. DOI: 10.1145/2814710.2814713. 284

J. Duggan, O. Papaemmanouil, L. Battle, and M. Stonebraker. 2015b. Skew-aware join optimization for array databases. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 123–135. DOI: 10.1145/2723372.2723709.

A. Dziedzic, J. Duggan, A. J. Elmore, V. Gadepally, and M. Stonebraker. 2015. BigDAWG: a polystore for diverse interactive applications. *Data Systems for Interactive Analysis Workshop*.

A. Dziedzic, A. J. Elmore, and M. Stonebraker. 2016. Data transformation and migration in polystores. In *Proc. 2016 IEEE High Performance Extreme Computing Conference*, pp. 1–6. DOI: 10.1109/HPEC.2016.7761594. 372

A. J. Elmore, J. Duggan, M. Stonebraker, M. Balazinska, U. Çetintemel, V. Gadepally, J. Heer, B. Howe, J. Kepner, T. Kraska, S. Madden, D. Maier, T. G. Mattson, S. Papadopoulos, J. Parkhurst, N. Tatbul, M. Vartak, and S. Zdonik. 2015. A demonstration of the BigDAWG polystore system. *Proc. VLDB Endowment*, 8(12): 1908–1911. http://www.vldb.org/pvldb/vol8/p1908-Elmore.pdf. 287, 371

R. S. Epstein and M. Stonebraker. 1980. Analysis of distributed data base processing strategies. In *Proc. 6th International Conference on Very Data Bases*, pp. 92–101.

R. S. Epstein, M. Stonebraker, and E. Wong. 1978. Distributed query processing in a relational data base system. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 169–180. DOI: 10.1145/509252.509292. 198

R. C. Fernandez, Z. Abedjan, S. Madden, and M. Stonebraker. 2016. Towards large-scale data discovery: position paper. In *Proc. 3rd International Workshop on Exploratory Search in Databases and the Web*, pp. 3–5. DOI: 10.1145/2948674.2948675.

R. C. Fernandez, D. Deng, E. Mansour, A. A. Qahtan, W. Tao, Z. Abedjan, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. 2017b. A demo of the data civilizer system. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 1639–1642. DOI: 10.1145/3035918.3058740.

R. C. Fernandez, D. Deng, E. Mansour, A. A. Qahtan, W. Tao, Z. Abedjan, A. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. 2017a. A demo of the data civilizer system. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 1636–1642. 293

R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. 2018a. Aurum: A data discovery system. In *Proc. 34th International Conference on Data Engineering*, pp. 1001–1012.

R. C. Fernandez, E. Mansour, A. Qahtan, A. Elmagarmid, I. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. 2018b. Seeping semantics: Linking datasets using word embeddings for data discovery. In *Proc. 34th International Conference on Data Engineering*, pp. 989–1000.

V. Gadepally, P. Chen, J. Duggan, A. J. Elmore, B. Haynes, J. Kepner, S. Madden, T. Mattson, and M. Stonebraker. 2016a. The BigDAWG polystore system and architecture. In *Proc. 2016 IEEE High Performance Extreme Computing Conference*, pp. 1–6. DOI: 10.1109/HPEC.2016.7761636. 287, 373

V. Gadepally, P. Chen, J. Duggan, A. J. Elmore, B. Haynes, J. Kepner, S. Madden, T. Mattson, and M. Stonebraker. 2016b. The BigDAWG polystore system and architecture. *CoRR*, abs/1609.07548. http://arxiv.org/abs/1609.07548.

V. Gadepally, P. Chen, J. Duggan, A. Elmore, B. Haynes, J. Kepnera, S. Madden, T. Mattson, and M. Stonebraker. 2016c. The BigDAWG polystore system and architecture. In *Proc. 2016 IEEE High Performance Extreme Computing Conference*, pp. 1–6. DOI: 10.1109/HPEC.2016.7761636.

V. Gadepally, J. Duggan, A. J. Elmore, J. Kepner, S. Madden, T. Mattson, and M. Stonebraker. 2016d. The BigDAWG architecture. *CoRR*, abs/1602.08791. http://arxiv.org/abs/1602.08791.

A. Go, M. Stonebraker, and C. Williams. 1975. An approach to implementing a geo-data system. In *Proc. Workshop on Data Bases for Interactive Design*, pp. 67–77.

J. Gray, H. Schek, M. Stonebraker, and J. D. Ullman. 2003. The Lowell report. In *Proc. ACM SIGMOD International Conference on Management of Data*, p. 680. DOI: 10.1145/872757.872873. 92

M. N. Gubanov and M. Stonebraker. 2013. Bootstraping synonym resolution at web scale. In *Proc. DIMACS/CCICADA Workshop on Big Data Integration*.

M. N. Gubanov and M. Stonebraker. 2014. Large-scale semantic profile extraction. In *Proc. 17th International Conference on Extending Database Technology*, pp. 644–647. DOI: 10.5441/002/edbt.2014.64.

M. N. Gubanov, M. Stonebraker, and D. Bruckner. 2014. Text and structured data fusion in data tamer at scale. In *Proc. 30th International Conference on Data Engineering*, pp. 1258–1261. DOI: 10.1109/ICDE.2014.6816755.

A. M. Gupta, V. Gadepally, and M. Stonebraker. 2016. Cross-engine query execution in federated database systems. In *Proc. 2016 IEEE High Performance Extreme Computing Conference*, pp. 1–6. DOI: 10.1109/HPEC.2016.7761648. 373

A. Guttman and M. Stonebraker. 1982. Using a relational database management system for computer aided design data. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 5(2): 21–28. http://sites.computer.org/debull/82JUN-CD.pdf. 201

J. Hammer, M. Stonebraker, and O. Topsakal. 2005. THALIA: test harness for the assessment of legacy information integration approaches. In *Proc. 21st International Conference on Data Engineering*, pp. 485–486. DOI: 10.1109/ICDE.2005.140.

R. Harding, D. V. Aken, A. Pavlo, and M. Stonebraker. 2017. An evaluation of distributed concurrency control. *Proc. VLDB Endowment*, 10(5): 553–564. DOI: 10.14778/3055540.3055548.

S. Harizopoulos, D. J. Abadi, S. Madden, and M. Stonebraker. 2008. OLTP through the looking glass, and what we found there. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 981–992. DOI: 10.1145/1376616.1376713. 152, 246, 251, 346

P. B. Hawthorn and M. Stonebraker. 1979. Performance analysis of a relational data base management system. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 1–12. DOI: 10.1145/582095.582097.

G. Held and M. Stonebraker. 1975. Storage structures and access methods in the relational data base management system INGRES. In *Proc. ACM Pacific 75—Data: Its Use, Organization and Management*, pp. 26–33. 194

G. Held and M. Stonebraker. 1978. B-trees re-examined. *Communications of the ACM*, 21(2): 139–143. DOI: 10.1145/359340.359348. 90, 197

G. Held, M. Stonebraker, and E. Wong. 1975. INGRES: A relational data base system. In *National Computer Conference*, pp. 409–416. DOI: 10.1145/1499949.1500029. 102, 397

J. M. Hellerstein and M. Stonebraker. 1993. Predicate migration: Optimizing queries with expensive predicates. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 267–276. DOI: 10.1145/170035.170078.

J. M. Hellerstein and M. Stonebraker. 2005. *Readings in Database Systems*, 4. MIT Press. ISBN 978-0-262-69314-1. http://mitpress.mit.edu/books/readings-database-systems.

J. M. Hellerstein, M. Stonebraker, and R. Caccia. 1999. Independent, open enterprise data integration. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 22(1): 43–49. http://sites.computer.org/debull/99mar/cohera.ps.

J. M. Hellerstein, M. Stonebraker, and J. R. Hamilton. 2007. Architecture of a database system. *Foundations and Trends in Databases*, 1(2): 141–259. DOI: 10.1561/1900000002.

W. Hong and M. Stonebraker. 1991. Optimization of parallel query execution plans in XPRS. In *Proc. 1st International Conference on Parallel and Distributed Information Systems*, pp. 218–225. DOI: 10.1109/PDIS.1991.183106.

W. Hong and M. Stonebraker. 1993. Optimization of parallel query execution plans in XPRS. *Distributed and Parallel Databases*, 1(1): 9–32. DOI: 10.1007/BF01277518.

J. Hwang, M. Balazinska, A. Rasin, U. Çetintemel, M. Stonebraker, and S. B. Zdonik. 2005. High-availability algorithms for distributed stream processing. In *Proc. 21st International Conference on Data Engineering*, pp. 779–790. DOI: 10.1109/ICDE.2005.72. 228, 230, 325

A. Jhingran and M. Stonebraker. 1990. Alternatives in complex object representation: A performance perspective. In *Proc. 6th International Conference on Data Engineering*, pp. 94–102. DOI: 10.1109/ICDE.1990.113458.

A. Jindal, P. Rawlani, E. Wu, S. Madden, A. Deshpande, and M. Stonebraker. 2014. VERTEXICA: your relational friend for graph analytics! *Proc. VLDB Endowment*, 7(13): 1669–1672. http://www.vldb.org/pvldb/vol7/p1669-jindal.pdf.

R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. B. Zdonik, E. P. C. Jones, S. Madden, M. Stonebraker, Y. Zhang, J. Hugg, and D. J. Abadi. 2008. H-store: a high-performance, distributed main memory transaction processing system. *Proc. VLDB Endowment*, 1(2): 1496–1499. DOI: 10.14778/1454159.1454211. 247, 249, 341

R. H. Katz, J. K. Ousterhout, D. A. Patterson, and M. Stonebraker. 1988. A project on high performance I/O subsystems. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 11(1): 40–47. http://sites.computer.org/debull/88MAR-CD.pdf.

J. T. Kohl, C. Staelin, and M. Stonebraker. 1993a. Highlight: Using a log-structured file system for tertiary storage management. In *Proc. of the Usenix Winter 1993 Technical Conference*, pp. 435–448.

J. T. Kohl, M. Stonebraker, and C. Staelin. 1993b. Highlight: a file system for tertiary storage. In *Proc. 12th IEEE Symposium Mass Storage Systems*, pp. 157–161. DOI: 10.1109/MASS.1993.289765.

C. P. Kolovson and M. Stonebraker. 1989. Indexing techniques for historical databases. In *Proc. 5th International Conference on Data Engineering*, pp. 127–137. DOI: 10.1109/ICDE.1989.47208.

C. P. Kolovson and M. Stonebraker. 1991. Segment indexes: Dynamic indexing techniques for multi-dimensional interval data. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 138–147. DOI: 10.1145/115790.115807.

R. A. Kowalski, D. B. Lenat, E. Soloway, M. Stonebraker, and A. Walker. 1988. Knowledge management—panel report. In *Proc. 2nd International Conference on Expert Database Systems*, pp. 63–69.

A. Kumar and M. Stonebraker. 1987a. The effect of join selectivities on optimal nesting order. *ACM SIGMOD Record*, 16(1): 28–41. DOI: 10.1145/24820.24822.

A. Kumar and M. Stonebraker. 1987b. Performance evaluation of an operating system transaction manager. In *Proc. 13th International Conference on Very Large Data Bases*, pp. 473–481. http://www.vldb.org/conf/1987/P473.pdf.

A. Kumar and M. Stonebraker. 1988. Semantics based transaction management techniques for replicated data. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 117–125. DOI: 10.1145/50202.50215.

A. Kumar and M. Stonebraker. 1989. Performance considerations for an operating system transaction manager. *IEEE Transactions on Software Engineering*, 15(6): 705–714. DOI: 10.1109/32.24724.

R. Kung, E. N. Hanson, Y. E. Ioannidis, T. K. Sellis, L. D. Shapiro, and M. Stonebraker. 1984. Heuristic search in data base systems. In *Proc. 1st International Workshop on Expert Database Systems*, pp. 537–548.

C. A. Lynch and M. Stonebraker. 1988. Extended user-defined indexing with application to textual databases. In *Proc. 14th International Conference on Very Large Data Bases*, pp. 306–317. http://www.vldb.org/conf/1988/P306.pdf.

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.

E. Mansour, D. Deng, A. Qahtan, R. C. Fernandez, Wenbo, Z. Abedjan, A. Elmagarmid, I. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. 2018. Building data civilizer pipelines with an advanced workflow engine. In *Proc. 34th International Conference on Data Engineering*, pp. 1593–1596.

T. Mattson, D. A. Bader, J. W. Berry, A. Buluç, J. Dongarra, C. Faloutsos, J. Feo, J. R. Gilbert, J. Gonzalez, B. Hendrickson, J. Kepner, C. E. Leiserson, A. Lumsdaine, D. A. Padua, S. Poole, S. P. Reinhardt, M. Stonebraker, S. Wallach, and A. Yoo. 2013. Standards for graph algorithm primitives. In *Proc. 2013 IEEE High Performance Extreme Computing Conference*, pp. 1–2. DOI: 10.1109/HPEC.2013.6670338.

T. Mattson, D. A. Bader, J. W. Berry, A. Buluç, J. J. Dongarra, C. Faloutsos, J. Feo, J. R. Gilbert, J. Gonzalez, B. Hendrickson, J. Kepner, C. E. Leiserson, A. Lumsdaine, D. A. Padua, S. W. Poole, S. P. Reinhardt, M. Stonebraker, S. Wallach, and A. Yoo. 2014. Standards for graph algorithm primitives. *CoRR*, abs/1408.0393. DOI: 10.1109/HPEC.2013.6670338.

N. H. McDonald and M. Stonebraker. 1975. CUPID - the friendly query language. In *Proc. ACM Pacific 75—Data: Its Use, Organization and Management*, pp. 127–131.

J. Meehan, N. Tatbul, S. B. Zdonik, C. Aslantas, U. Çetintemel, J. Du, T. Kraska, S. Madden, D. Maier, A. Pavlo, M. Stonebraker, K. Tufte, and H. Wang. 2015a. S-store: Streaming meets transaction processing. *CoRR*, abs/1503.01143. DOI: 10.14778/2831360.2831367. 234

J. Meehan, N. Tatbul, S. Zdonik, C. Aslantas, U. Çetintemel, J. Du, T. Kraska, S. Madden, D. Maier, A. Pavlo, M. Stonebraker, K. Tufte, and H. Wang. 2015b. S-store: Streaming meets transaction processing. *Proc. VLDB Endowment*, 8(13): 2134–2145. DOI: 10.14778/2831360.2831367. 234, 288, 331, 374

J. Morcos, Z. Abedjan, I. F. Ilyas, M. Ouzzani, P. Papotti, and M. Stonebraker. 2015. Dataxformer: An interactive data transformation tool. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 883–888. DOI: 10.1145/2723372.2735366. 296

B. Muthuswamy, L. Kerschberg, C. Zaniolo, M. Stonebraker, D. S. P. Jr., and M. Jarke. 1985. Architectures for expert-DBMS (panel). In *Proc. 1985 ACM Annual Conference on the Range of Computing: Mid-80's*, pp. 424–426. DOI: 10.1145/320435.320555.

K. O'Brien, V. Gadepally, J. Duggan, A. Dziedzic, A. J. Elmore, J. Kepner, S. Madden, T. Mattson, Z. She, and M. Stonebraker. 2017. BigDAWG polystore release and demonstration. *CoRR*, abs/1701.05799. http://arxiv.org/abs/1701.05799.

V. E. Ogle and M. Stonebraker. 1995. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9): 40–48. DOI: 10.1109/2.410150.

M. A. Olson, W. Hong, M. Ubell, and M. Stonebraker. 1996. Query processing in a parallel object-relational database system. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 19(4): 3–10. http://sites.computer.org/debull/96DEC-CD.pdf.

C. Olston, M. Stonebraker, A. Aiken, and J. M. Hellerstein. 1998a. VIQING: visual interactive querying. In *Proc. 1998 IEEE Symposium on Visual Languages*, pp. 162–169. DOI: 10.1109/VL.1998.706159.

C. Olston, A. Woodruff, A. Aiken, M. Chu, V. Ercegovac, M. Lin, M. Spalding, and M. Stonebraker. 1998b. Datasplash. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 550–552. DOI: 10.1145/276304.276377.

J. Ong, D. Fogg, and M. Stonebraker. 1984. Implementation of data abstraction in the relational database system ingres. *ACM SIGMOD Record*, 14(1): 1–14. DOI: 10.1145/984540.984541. 201, 202, 206

R. Overmyer and M. Stonebraker. 1982. Implementation of a time expert in a data base system. *ACM SIGMOD Record*, 12(3): 51–60. DOI: 10.1145/984505.984509.

A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker. 2009. A comparison of approaches to large-scale data analysis. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 165–178. DOI: 10.1145/1559845.1559865.

H. Pirk, S. Madden, and M. Stonebraker. 2015. By their fruits shall ye know them: A data analyst's perspective on massively parallel system design. In *Proc. 11th Workshop on Data Management on New Hardware*, pp. 5:1–5:6. DOI: 10.1145/2771937.2771944.

G. Planthaber, M. Stonebraker, and J. Frew. 2012. Earthdb: scalable analysis of MODIS data using SciDB. In *Proc. 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pp. 11–19. DOI: 10.1145/2447481.2447483.

S. Potamianos and M. Stonebraker. 1996. The POSTGRES rules system. In *Active Database Systems: Triggers and Rules For Advanced Database Processing*, pp. 43–61. Morgan Kaufmann. 91, 168

C. Ré, D. Agrawal, M. Balazinska, M. Cafarella, M. Jordan, T. Kraska, and R. Ramakrishnan. 2015. Machine learning and databases: The sound of things to come or a cacophony of hype? In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 283–284.

D. R. Ries and M. Stonebraker. 1977a. Effects of locking granularity in a database management system. *ACM Transactions on Database Systems*, 2(3): 233–246. DOI: 10.1145/320557.320566. 91, 198

D. R. Ries and M. Stonebraker. 1977b. A study of the effects of locking granularity in a data base management system (abstract). In *Proc. ACM SIGMOD International Conference on Management of Data*, p. 121. DOI: 10.1145/509404.509422. 91

D. R. Ries and M. Stonebraker. 1979. Locking granularity revisited. *ACM Transactions on Database Systems*, 4(2): 210–227. http://doi.acm.org/10.1145/320071.320078. DOI: 10.1145/320071.320078. 91

L. A. Rowe and M. Stonebraker. 1981. Architecture of future data base systems. *ACM SIGMOD Record*, 11(1): 30–44. DOI: 10.1145/984471.984473.

L. A. Rowe and M. Stonebraker. 1986. The commercial INGRES epilogue. In M. Stonebraker, editor, *The INGRES Papers: Anatomy of a Relational Database System*, pp. 63–82. Addison-Wesley.

L. A. Rowe and M. Stonebraker. 1987. The POSTGRES data model. In *Proc. 13th International Conference on Very Large Data Bases*, pp. 83–96. http://www.vldb.org/conf/1987/P083.pdf. 258

L. A. Rowe and M. Stonebraker. 1990. The POSTGRES data model. In A. F. Cardenas and D. McLeod, editors, *Research Foundations in Object-Oriented and Semantic Database Systems*, pp. 91–110. Prentice Hall.

S. Sarawagi and M. Stonebraker. 1994. Efficient organization of large multidimensional arrays. In *Proc. 10th International Conference on Data Engineering*, pp. 328–336. DOI: 10.1109/ICDE.1994.283048.

S. Sarawagi and M. Stonebraker. 1996. Reordering query execution in tertiary memory databases. In *Proc. 22th International Conference on Very Large Data Bases*. http://www.vldb.org/conf/1996/P156.pdf.

G. A. Schloss and M. Stonebraker. 1990. Highly redundant management of distributed data. In *Proc. Workshop on the Management of Replicated Data*, pp. 91–92.

A. Seering, P. Cudré-Mauroux, S. Madden, and M. Stonebraker. 2012. Efficient versioning for scientific array databases. In *Proc. 28th International Conference on Data Engineering*, pp. 1013–1024. DOI: 10.1109/ICDE.2012.102.

L. J. Seligman, N. J. Belkin, E. J. Neuhold, M. Stonebraker, and G. Wiederhold. 1995. Metrics for accessing heterogeneous data: Is there any hope? (panel). In *Proc. 21th International Conference on Very Large Data Bases*, p. 633. http://www.vldb.org/conf/1995/P633.pdf.

M. I. Seltzer and M. Stonebraker. 1990. Transaction support in read optimizied and write optimized file systems. In *Proc. 16th International Conference on Very Large Data Bases*, pp. 174–185. http://www.vldb.org/conf/1990/P174.pdf.

M. I. Seltzer and M. Stonebraker. 1991. Read optimized file system designs: A performance evaluation. In *Proc. 7th International Conference on Data Engineering*, pp. 602–611. DOI: 10.1109/ICDE.1991.131509.

M. Serafini, R. Taft, A. J. Elmore, A. Pavlo, A. Aboulnaga, and M. Stonebraker. 2016. Clay: Fine-grained adaptive partitioning for general database schemas. *Proc. VLDB Endowment*, 10(4): 445–456. DOI: 10.14778/3025111.3025125.

J. Sidell, P. M. Aoki, A. Sah, C. Staelin, M. Stonebraker, and A. Yu. 1996. Data replication in mariposa. In *Proc. 12th International Conference on Data Engineering*, pp. 485–494. DOI: 10.1109/ICDE.1996.492198.

A. Silberschatz, M. Stonebraker, and J. D. Ullman. 1990. Database systems: Achievements and opportunities—the "Lagunita" report of the NSF invitational workshop on the future of database system research held in Palo Alto, CA, February 22–23, 1990. *ACM SIGMOD Record*, 19(4): 6–22. DOI: 10.1145/122058.122059. 92

A. Silberschatz, M. Stonebraker, and J. D. Ullman. 1991. Database systems: Achievements and opportunities. *Communications of the ACM*, 34(10): 110–120. DOI: 10.1145/125223.125272.

A. Silberschatz, M. Stonebraker, and J. D. Ullman. 1996. Database research: Achievements and opportunities into the 21st century. *ACM SIGMOD Record*, 25(1): 52–63. DOI: 10.1145/381854.381886.

D. Skeen and M. Stonebraker. 1981. A formal model of crash recovery in a distributed system. In *Proc. 5th Berkeley Workshop on Distributed Data Management and Computer Networks*, pp. 129–142.

D. Skeen and M. Stonebraker. 1983. A formal model of crash recovery in a distributed system. *IEEE Transactions on Software Engineering*, 9(3): 219–228. DOI: 10.1109/TSE.1983.236608. 199

M. Stonebraker and R. Cattell. 2011. 10 rules for scalable performance in "simple operation" datastores. *Communications of the ACM*, 54(6): 72–80. DOI: 10.1145/1953122.1953144.

M. Stonebraker and U. Çetintemel. 2005. "One size fits all": An idea whose time has come and gone (abstract). In *Proc. 21st International Conference on Data Engineering*, pp. 2–11. DOI: 10.1109/ICDE.2005.1. 50, 92, 103, 131, 152, 367, 401

M. Stonebraker and D. J. DeWitt. 2008. A tribute to Jim Gray. *Communications of the ACM*, 51(11): 54–57. DOI: 10.1145/1400214.1400230.

M. Stonebraker and A. Guttman. 1984. Using a relational database management system for computer aided design data—an update. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 7(2): 56–60. http://sites.computer.org/debull/84JUN-CD.pdf.

M. Stonebraker and A. Guttman. 1984. R-trees: a dynamic index structure for spatial searching. In *Proc. of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD '84)*, pp. 47–57. ACM, New York. DOI:10.1145/602259.602266. 201

M. Stonebraker and M. A. Hearst. 1988. Future trends in expert data base systems. In *Proc. 2nd International Conference on Expert Database Systems*, pp. 3–20. 395

M. Stonebraker and G. Held. 1975. Networks, hierarchies and relations in data base management systems. In *Proc. ACM Pacific 75—Data: Its Use, Organization and Management*, pp. 1–9.

M. Stonebraker and J. M. Hellerstein, editors. 1998. *Readings in Database Systems*, 3. Morgan Kaufmann.

M. Stonebraker and J. M. Hellerstein. 2001. Content integration for e-business. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 552–560. DOI: 10.1145/375663.375739.

M. Stonebraker and J. Hong. 2009. Saying good-bye to DBMSS, designing effective interfaces. *Communications of the ACM*, 52(9): 12–13. DOI: 10.1145/1562164.1562169.

M. Stonebraker and J. Hong. 2012. Researchers' big data crisis; understanding design and functionality. *Communications of the ACM*, 55(2): 10–11. DOI: 10.1145/2076450.2076453.

M. Stonebraker and J. Kalash. 1982. TIMBER: A sophisticated relation browser (invited paper). In *Proc. 8th International Conference on Very Data Bases*, pp. 1–10. http://www.vldb.org/conf/1982/P001.pdf.

M. Stonebraker and K. Keller. 1980. Embedding expert knowledge and hypothetical data bases into a data base system. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 58–66. DOI: 10.1145/582250.582261. 200

M. Stonebraker and G. Kemnitz. 1991. The Postgres next generation database management system. *Communications of the ACM*, 34(10): 78–92. DOI: 10.1145/125223.125262. 168, 206, 213

M. Stonebraker and A. Kumar. 1986. Operating system support for data management. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 9(3): 43–50. http://sites.computer.org/debull/86SEP-CD.pdf. 47

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

M. Stonebraker and E. J. Neuhold. 1977. A distributed database version of INGRES. In *Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks*, pp. 19–36. 109, 198, 199

M. Stonebraker and M. A. Olson. 1993. Large object support in POSTGRES. In *Proc. 9th International Conference on Data Engineering*, pp. 355–362. DOI: 10.1109/ICDE.1993.344046.

M. Stonebraker and J. Robertson. 2013. Big data is "buzzword du jour;" CS academics "have the best job". *Communications of the ACM*, 56(9): 10–11. DOI: 10.1145/2500468.2500471.

M. Stonebraker and L. A. Rowe. 1977. Observations on data manipulation languages and their embedding in general purpose programming languages. In *Proc. 3rd International Conference on Very Data Bases*, pp. 128–143.

M. Stonebraker and L. A. Rowe. 1984. Database portals: A new application program interface. In *Proc. 10th International Conference on Very Large Data Bases*, pp. 3–13. http://www.vldb.org/conf/1984/P003.pdf.

M. Stonebraker and L. A. Rowe. 1986. The design of Postgres. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 340–355. DOI: 10.1145/16894.16888. 149, 203, 206

M. Stonebraker and P. Rubinstein. 1976. The INGRES protection system. In *Proc. 1976 ACM Annual Conference*, pp. 80–84. DOI: 10.1145/800191.805536. 398

M. Stonebraker and G. A. Schloss. 1990. Distributed RAID—A new multiple copy algorithm. In *Proc. 6th International Conference on Data Engineering*, pp. 430–437. DOI: 10.1109/ICDE.1990.113496.

M. Stonebraker and A. Weisberg. 2013. The VoltDB main memory DBMS. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 36(2): 21–27. http://sites.computer.org/debull/A13june/VoltDB1.pdf.

M. Stonebraker and E. Wong. 1974b. Access control in a relational data base management system by query modification. In *Proc. 1974 ACM Annual Conference, Volume 1*, pp. 180–186. DOI: 10.1145/800182.810400. 45

M. Stonebraker, P. Rubinstein, R. Conway, D. Strip, H. R. Hartson, D. K. Hsiao, and E. B. Fernandez. 1976a. SIGBDP (paper session). In *Proc. 1976 ACM Annual Conference*, p. 79. DOI: 10.1145/800191.805535.

M. Stonebraker, E. Wong, P. Kreps, and G. Held. 1976b. The design and implementation of INGRES. *ACM Transactions on Database Systems*, 1(3): 189–222. DOI: 10.1145/320473.320476. 47, 148, 398

M. Stonebraker, R. R. Johnson, and S. Rosenberg. 1982a. A rules system for a relational data base management system. In *Proc. 2nd International Conference on Databases: Improving Database Usability and Responsiveness*, pp. 323–335. 91, 202

M. Stonebraker, J. Woodfill, J. Ranstrom, M. C. Murphy, J. Kalash, M. J. Carey, and K. Arnold. 1982b. Performance analysis of distributed data base systems. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 5(4): 58–65. http://sites.computer.org/debull/82DEC-CD.pdf.

M. Stonebraker, W. B. Rubenstein, and A. Guttman. 1983a. Application of abstract data types and abstract indices to CAD data bases. In *Engineering Design Applications*, pp. 107–113.

M. Stonebraker, H. Stettner, N. Lynn, J. Kalash, and A. Guttman. 1983b. Document processing in a relational database system. *ACM Transactions on Information Systems*, 1(2): 143–158. DOI: 10.1145/357431.357433.

M. Stonebraker, J. Woodfill, and E. Andersen. 1983c. Implementation of rules in relational data base systems. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 6(4): 65–74. http://sites.computer.org/debull/83DEC-CD.pdf. 91, 202

M. Stonebraker, J. Woodfill, J. Ranstrom, J. Kalash, K. Arnold, and E. Andersen. 1983d. Performance analysis of distributed data base systems. In *Proc. 2nd Symposium on Reliable Distributed Systems*, pp. 135–138.

M. Stonebraker, J. Woodfill, J. Ranstrom, M. C. Murphy, M. Meyer, and E. Allman. 1983e. Performance enhancements to a relational database system. *ACM Transactions on Database Systems*, 8(2): 167–185. DOI: 10.1145/319983.319984.

M. Stonebraker, E. Anderson, E. N. Hanson, and W. B. Rubenstein. 1984a. Quel as a data type. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 208–214. DOI: 10.1145/602259.602287. 208

M. Stonebraker, J. Woodfill, J. Ranstrom, J. Kalash, K. Arnold, and E. Andersen. 1984b. Performance analysis of distributed data base systems. *Performance Evaluation*, 4(3): 220. DOI: 10.1016/0166-5316(84)90036-1.

M. Stonebraker, D. DuBourdieux, and W. Edwards. 1985. Problems in supporting data base transactions in an operating system transaction manager. *Operating Systems Review*, 19(1): 6–14. DOI: 10.1145/1041490.1041491.

M. Stonebraker, T. K. Sellis, and E. N. Hanson. 1986. An analysis of rule indexing implementations in data base systems. In *Proc. 1st International Conference on Expert Database Systems*, pp. 465–476. 91

M. Stonebraker, J. Anton, and E. N. Hanson. 1987a. Extending a database system with procedures. *ACM Transactions on Database Systems*, 12(3): 350–376. DOI: 10.1145/27629.27631.

M. Stonebraker, J. Anton, and M. Hirohama. 1987b. Extendability in POSTGRES. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 10(2): 16–23. http://sites.computer.org/debull/87JUN-CD.pdf.

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. 91

M. Stonebraker, E. N. Hanson, and S. Potamianos. 1988a. The POSTGRES rule manager. *IEEE Transactions on Software Engineering*, 14(7): 897–907. DOI: 10.1109/32.42733. 91, 168

M. Stonebraker, R. H. Katz, D. A. Patterson, and J. K. Ousterhout. 1988b. The design of XPRS. In *Proc. 14th International Conference on Very Large Data Bases*, pp. 318–330. http://www.vldb.org/conf/1988/P318.pdf.

M. Stonebraker, M. A. Hearst, and S. Potamianos. 1989. A commentary on the POSTGRES rule system. *ACM SIGMOD Record*, 18(3): 5–11. DOI: 10.1145/71031.71032. 91, 168, 395

M. Stonebraker, A. Jhingran, J. Goh, and S. Potamianos. 1990a. On rules, procedures, caching and views in data base systems. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 281–290. DOI: 10.1145/93597.98737.

M. Stonebraker, L. A. Rowe, and M. Hirohama. 1990b. The implementation of postgres. *IEEE Transactions on Knowledge and Data Engineering*, 2(1): 125–142. DOI: 10.1109/69.50912. 47, 168

M. Stonebraker, L. A. Rowe, B. G. Lindsay, J. Gray, M. J. Carey, and D. Beech. 1990e. Third generation data base system manifesto—The committee for advanced DBMS function. In *Proc. ACM SIGMOD International Conference on Management of Data*, p. 396.

M. Stonebraker, L. A. Rowe, B. G. Lindsay, J. Gray, M. J. Carey, M. L. Brodie, P. A. Bernstein, and D. Beech. 1990c. Third-generation database system manifesto—The committee for advanced DBMS function. *ACM SIGMOD Record*, 19(3): 31–44. DOI: 10.1145/101077.390001. 91

M. Stonebraker, L. A. Rowe, B. G. Lindsay, J. Gray, M. J. Carey, M. L. Brodie, P. A. Bernstein, and D. Beech. 1990d. Third-generation database system manifesto—The committee for advanced DBMS function. In *Proc. IFIP TC2/WG 2.6 Working Conference on Object-Oriented Databases: Analysis, Design & Construction*, pp. 495–511. 91

M. Stonebraker, R. Agrawal, U. Dayal, E. J. Neuhold, and A. Reuter. 1993a. DBMS research at a crossroads: The Vienna update. In *Proc. 19th International Conference on Very Large Data Bases*, pp. 688–692. http://www.vldb.org/conf/1993/P688.pdf.

M. Stonebraker, J. Chen, N. Nathan, C. Paxson, A. Su, and J. Wu. 1993b. Tioga: A database-oriented visualization tool. In *Proceedings IEEE Conference Visualization*, pp. 86–93. DOI: 10.1109/VISUAL.1993.398855. 393

M. Stonebraker, J. Chen, N. Nathan, C. Paxson, and J. Wu. 1993c. Tioga: Providing data management support for scientific visualization applications. In *Proc. 19th International Conference on Very Large Data Bases*, pp. 25–38. http://www.vldb.org/conf/1993/P025.pdf. 393

M. Stonebraker, J. Frew, and J. Dozier. 1993d. The SEQUOIA 2000 project. In *Proc. 3rd International Symposium Advances in Spatial Databases*, pp. 397–412. DOI: 10.1007/3-540-56869-7_22.

M. Stonebraker, J. Frew, K. Gardels, and J. Meredith. 1993e. The Sequoia 2000 benchmark. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 2–11. DOI: 10.1145/170035.170038.

M. Stonebraker, P. M. Aoki, R. Devine, W. Litwin, and M. A. Olson. 1994a. Mariposa: A new architecture for distributed data. In *Proc. 10th International Conference on Data Engineering*, pp. 54–65. DOI: 10.1109/ICDE.1994.283004. 401

M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, and C. Staelin. 1994b. An economic paradigm for query processing and data migration in Mariposa. In *Proc. 3rd International Conference on Parallel and Distributed Information Systems*, pp. 58–67. DOI: 10.1109/PDIS.1994.331732.

M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. 1996. Mariposa: A wide-area distributed database system. *VLDB Journal*, 5(1): 48–63. DOI: 10.1007/s007780050015.

M. Stonebraker, P. Brown, and M. Herbach. 1998a. Interoperability, distributed applications and distributed databases: The virtual table interface. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 21(3): 25–33. http://sites.computer.org/debull/98sept/informix.ps.

M. Stonebraker, P. Brown, and D. Moore. 1998b. *Object-Relational DBMSs*, 2. Morgan Kaufmann.

M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik. 2005a. C-store: A column-oriented DBMS. In *Proc. 31st International Conference on Very Large Data Bases*, pp. 553–564. http://www.vldb2005.org/program/paper/thu/p553-stonebraker.pdf. 104, 132, 151, 238, 242, 258, 333, 335, 402

M. Stonebraker, U. Çetintemel, and S. B. Zdonik. 2005b. The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, 34(4): 42–47. DOI: 10.1145/1107499.1107504. 282

M. Stonebraker, C. Bear, U. Çetintemel, M. Cherniack, T. Ge, N. Hachem, S. Harizopoulos, J. Lifter, J. Rogers, and S. B. Zdonik. 2007a. One size fits all? Part 2: Benchmarking studies. In *Proc. 3rd Biennial Conference on Innovative Data Systems Research*, pp. 173–184. http://www.cidrdb.org/cidr2007/papers/cidr07p20.pdf. 103, 282

M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. 2007b. The end of an architectural era (it's time for a complete rewrite). In *Proc. 33rd International Conference on Very Large Data Bases*, pp. 1150–1160. http://www.vldb.org/conf/2007/papers/industrial/p1150-stonebraker.pdf. 247, 341, 344

M. Stonebraker, J. Becla, D. J. DeWitt, K. Lim, D. Maier, O. Ratzesberger, and S. B. Zdonik. 2009. Requirements for science data bases and SciDB. In *Proc. 4th Biennial Conference on Innovative Data Systems Research*. http://www-db.cs.wisc.edu/cidr/cidr2009/Paper_26.pdf. 257

M. Stonebraker, D. J. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin. 2010. Mapreduce and parallel DBMSS: friends or foes? *Communications of the ACM*, 53(1): 64–71. DOI: 10.1145/1629175.1629197. 50, 136, 251

M. Stonebraker, P. Brown, A. Poliakov, and S. Raman. 2011. The architecture of SciDB. In *Proc. 23rd International Conference on Scientific and Statistical Database Management*, pp. 1–16. DOI: 10.1007/978-3-642-22351-8_1.

M. Stonebraker, A. Ailamaki, J. Kepner, and A. S. Szalay. 2012. The future of scientific data bases. In *Proc. 28th International Conference on Data Engineering*, pp. 7–8. DOI: 10.1109/ICDE.2012.151.

M. Stonebraker, P. Brown, D. Zhang, and J. Becla. 2013a. SciDB: A database management system for applications with complex analytics. *Computing in Science and Engineering*, 15(3): 54–62. DOI: 10.1109/MCSE.2013.19.

M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. 2013b. Data curation at scale: The data tamer system. In *Proc. 6th Biennial Conference on Innovative Data Systems Research*. http://www.cidrdb.org/cidr2013/Papers/CIDR13_Paper28.pdf. 105, 150, 269, 297, 357, 358

M. Stonebraker, J. Duggan, L. Battle, and O. Papaemmanouil. 2013c. SciDB DBMS research at M.I.T. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 36(4): 21–30. http://sites.computer.org/debull/A13dec/p21.pdf.

M. Stonebraker, S. Madden, and P. Dubey. 2013d. Intel "big data" science and technology center vision and execution plan. *ACM SIGMOD Record*, 42(1): 44–49. DOI: 10.1145/2481528.2481537.

M. Stonebraker, A. Pavlo, R. Taft, and M. L. Brodie. 2014. Enterprise database applications and the cloud: A difficult road ahead. In *Proc. 7th IEEE International Conference on Cloud Computing*, pp. 1–6. DOI: 10.1109/IC2E.2014.97.

M. Stonebraker, D. Deng, and M. L. Brodie. 2016. Database decay and how to avoid it. In *Proc. 2016 IEEE International Conference on Big Data*, pp. 7–16. DOI: 10.1109/BigData.2016.7840584.

M. Stonebraker, D. Deng, and M. L. Brodie. 2017. Application-database co-evolution: A new design and development paradigm. In *North East Database Day*, pp. 1–3.

M. Stonebraker. 1972a. Retrieval efficiency using combined indexes. In *Proc. 1972 ACM-SIGFIDET Workshop on Data Description, Access and Control*, pp. 243–256.

M. Stonebraker. 1972b. A simplification of forrester's model of an urban area. *IEEE Transactions on Systems, Man, and Cybernetics*, 2(4): 468–472. DOI: 10.1109/TSMC.1972.4309156.

M. Stonebraker. 1974a. The choice of partial inversions and combined indices. *International Journal Parallel Programming*, 3(2): 167–188. DOI: 10.1007/BF00976642.

M. Stonebraker. 1974b. A functional view of data independence. In *Proc. 1974 ACM SIGMOD Workshop on Data Description, Access and Control*, pp. 63–81. DOI: 10.1145/800296.811505. 404, 405

M. Stonebraker. 1975. Getting started in INGRES—A tutorial, Memorandum No. ERL-M518, Electronics Research Laboratory, College of Engineering, UC Berkeley. 196

M. Stonebraker. 1975. Implementation of integrity constraints and views by query modification. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 65–78. DOI: 10.1145/500080.500091. 45, 90

M. Stonebraker. 1976a. Proposal for a network INGRES. In *Proc. 1st Berkeley Workshop on Distributed Data Management and Computer Networks*, p. 132.

M. Stonebraker. 1976b. The data base management system INGRES. In *Proc. 1st Berkeley Workshop on Distributed Data Management and Computer Networks*, p. 336. 195

M. Stonebraker. 1976c. A comparison of the use of links and secondary indices in a relational data base system. In *Proc. 2nd International Conference on Software Engineering*, pp. 527–531. http://dl.acm.org/citation.cfm?id=807727.

M. Stonebraker. 1978. Concurrency control and consistency of multiple copies of data in distributed INGRES. In *Proc. 3rd Berkeley Workshop on Distributed Data Management and Computer Networks*, pp. 235–258. 90, 398

M. Stonebraker. May 1979a. Muffin: A distributed database machine. Technical Report ERL Technical Report UCB/ERL M79/28, University of California at Berkeley. 151

M. Stonebraker. 1979b. Concurrency control and consistency of multiple copies of data in distributed INGRES. *IEEE Transactions on Software Engineering*, 5(3): 188–194. DOI: 10.1109/TSE.1979.234180. 398

M. Stonebraker. 1980. Retrospection on a database system. *ACM Transactions on Database Systems*, 5(2): 225–240. DOI: 10.1145/320141.320158.

M. Stonebraker. 1981a. Operating system support for database management. *Communications of the ACM*, 24(7): 412–418. DOI: 10.1145/358699.358703.

M. Stonebraker. 1981b. Chairman's column. *ACM SIGMOD Record*, 11(3): i–iv.

M. Stonebraker. 1981c. Chaiman's column. *ACM SIGMOD Record*, 11(4): 2–4.

M. Stonebraker. 1981d. Chairman's column. *ACM SIGMOD Record*, 12(1): 1–3.

M. Stonebraker. 1981e. In memory of Kevin Whitney. *ACM SIGMOD Record*, 12(1): 7.

M. Stonebraker. 1981f. Chairman's column. *ACM SIGMOD Record*, 11(1): 1–4.

M. Stonebraker. 1981g. Hypothetical data bases as views. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 224–229. DOI: 10.1145/582318.582352.

M. Stonebraker. 1982a. Chairman's column. *ACM SIGMOD Record*, 12(3): 2–4.

M. Stonebraker. 1982b. Letter to Peter Denning (two VLDB conferences). *ACM SIGMOD Record*, 12(3): 6–7.

M. Stonebraker. 1982c. Chairman's column. *ACM SIGMOD Record*, 12(4): a–c.

M. Stonebraker. 1982d. Chairman's column. *ACM SIGMOD Record*, 13(1): 2–3&4.

M. Stonebraker. 1982e. Adding semantic knowledge to a relational database system. In M. L. Brodie, M. John, and S. J. W., editors, *On Conceptual Modelling*, pp. 333–352. Springer. DOI: 10.1007/978-1-4612-5196-5_12.

M. Stonebraker. 1982f. A database perspective. In M. L. Brodie, M. John, and S. J. W., editors, *On Conceptual Modelling*, pp. 457–458. Springer. DOI: 10.1007/978-1-4612-5196-5_18.

M. Stonebraker. 1983a. DBMS and AI: is there any common point of view? In *Proc. ACM SIGMOD International Conference on Management of Data*, p. 134. DOI: 10.1145/582192.582215. 201, 205

M. Stonebraker. April 1983b. Chairman's column. *ACM SIGMOD Record*, 13(3): 1–3.

M. Stonebraker. January 1983c. Chairman's column. *ACM SIGMOD Record*, 13(2): 1–3.

M. Stonebraker. 1984. Virtual memory transaction management. *Operating Systems Review*, 18(2): 8–16. DOI: 10.1145/850755.850757. 203

M. Stonebraker. 1985a. Triggers and inference in data base systems. In *Proc. 1985 ACM Annual Conference on the Range of Computing: Mid-80's Perspective*, p. 426. DOI: 10.1145/320435.323372.

M. Stonebraker. 1985b. Triggers and inference in database systems. In M. L. Brodie and J. Mylopoulos, editors, *On Knowledge Base Management Systems*, pp. 297–314. Springer. 202

M. Stonebraker. 1985c. Expert database systems/bases de données et systèmes experts. In *Journées Bases de Données Avancés*.

M. Stonebraker. 1985d. The case for shared nothing. In *Proc. International Workshop on High-Performance Transaction Systems*, p. 0. 91

M. Stonebraker. 1985e. Tips on benchmarking data base systems. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 8(1): 10–18. http://sites.computer.org/debull/85MAR-CD.pdf.

M. Stonebraker, editor. 1986a. *The INGRES Papers: Anatomy of a Relational Database System*. Addison-Wesley.

M. Stonebraker. 1986b. Inclusion of new types in relational data base systems. In *Proc. 2nd International Conference on Data Engineering*, pp. 262–269. DOI: 10.1109/ICDE.1986.7266230. 88, 202, 258

M. Stonebraker. 1986c. Object management in Postgres using procedures. In *Proc. International Workshop on Object-Oriented Database Systems*, pp. 66–72. http://dl.acm.org/citation.cfm?id=318840. 45, 88, 399

M. Stonebraker. 1986d. The case for shared nothing. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 9(1): 4–9. http://sites.computer.org/debull/86MAR-CD.pdf. 91, 216

M. Stonebraker. 1986e. Design of relational systems (introduction to section 1). In M. Stonebraker, editor, *The INGRES Papers: Anatomy of a Relational Database System*, pp. 1–3. Addison-Wesley.

M. Stonebraker. 1986f. Supporting studies on relational systems (introduction to section 2). In M. Stonebraker, editor, *The INGRES Papers*, pp. 83–85. Addison-Wesley.

M. Stonebraker. 1986g. Distributed database systems (introduction to section 3). In M. Stonebraker, editor, *The INGRES Papers: Anatomy of a Relational Database System*, pp. 183–186. Addison-Wesley.

M. Stonebraker. 1986h. The design and implementation of distributed INGRES. In M. Stonebraker, editor, *The INGRES Papers: Anatomy of a Relational Database System*, pp. 187–196. Addison-Wesley.

M. Stonebraker. 1986i. User interfaces for database systems (introduction to section 4). In M. Stonebraker, editor, *The INGRES Papers: Anatomy of a Relational Database System*, pp. 243–245. Addison-Wesley.

M. Stonebraker. 1986j. Extended semantics for the relational model (introduction to section 5). In M. Stonebraker, editor, *The INGRES Papers: Anatomy of a Relational Database System*, pp. 313–316. Addison-Wesley.

M. Stonebraker. 1986k. Database design (introduction to section 6). In M. Stonebraker, editor, *The INGRES Papers: Anatomy of a Relational Database System*, pp. 393–394. Addison-Wesley.

M. Stonebraker. 1986l. Object management in a relational data base system. In *Digest of Papers - COMPCON*, pp. 336–341.

M. Stonebraker. 1987. The design of the POSTGRES storage system. In *Proc. 13th International Conference on Very Large Data Bases*, pp. 289–300. http://www.vldb.org/conf/1987/P289.pdf. 168, 214, 258

M. Stonebraker, editor. 1988a. *Readings in Database Systems*. Morgan Kaufmann.

M. Stonebraker. 1988b. Future trends in data base systems. In *Proc. 4th International Conference on Data Engineering*, pp. 222–231. DOI: 10.1109/ICDE.1988.105464.

M. Stonebraker. 1989a. The case for partial indexes. *ACM SIGMOD Record*, 18(4): 4–11. DOI: 10.1145/74120.74121.

M. Stonebraker. 1989b. Future trends in database systems. *IEEE Transactions on Knowledge and Data Engineering*, 1(1): 33–44. DOI: 10.1109/69.43402.

M. Stonebraker. 1990a. The third-generation database manifesto: A brief retrospection. In *Proc. IFIP TC2/WG 2.6 Working Conference on Object-Oriented Databases: Analysis, Design & Construction*, pp. 71–72.

M. Stonebraker. 1990b. Architecture of future data base systems. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 13(4): 18–23. http://sites.computer.org/debull/90DEC-CD.pdf.

M. Stonebraker. 1990c. Data base research at Berkeley. *ACM SIGMOD Record*, 19(4): 113–118. DOI: 10.1145/122058.122072.

M. Stonebraker. 1990d. Introduction to the special issue on database prototype systems. *IEEE Transactions on Knowledge and Data Engineering*, 2(1): 1–3. DOI: 10.1109/TKDE.1990.10000.

M. Stonebraker. 1990e. The Postgres DBMS. In *Proc. ACM SIGMOD International Conference on Management of Data*, p. 394.

M. Stonebraker. 1991a. Managing persistent objects in a multi-level store. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 2–11. DOI: 10.1145/115790.115791.

M. Stonebraker. 1991b. Object management in Postgres using procedures. In K. R. Dittrich, U. Dayal, and A. P. Buchmann, editors, *On Object-Oriented Database System*, pp. 53–64. Springer. DOI: http://10.1007/978-3-642-84374-7_5.

M. Stonebraker. 1971. The reduction of large scale Markov models for random chains. Ph.D. Dissertation. University of Michigan, Ann Arbor, MI. AAI7123885. 43

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

M. Stonebraker, editor. 1992b. *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*. ACM Press.

M. Stonebraker. 1993a. The SEQUOIA 2000 project. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 16(1): 24–28. http://sites.computer.org/debull/93MAR-CD.pdf.

M. Stonebraker. 1993b. Are we polishing a round ball? (panel abstract). In *Proc. 9th International Conference on Data Engineering*, p. 606.

M. Stonebraker. 1993c. The miro DBMS. In *Proc. ACM SIGMOD International Conference on Management of Data*, p. 439. DOI: 10.1145/170035.170124. 314

M. Stonebraker. 1994a. SEQUOIA 2000: A reflection of the first three years. In *Proc. 7th International Working Conference on Scientific and Statistical Database Management*, pp. 108–116. DOI: 10.1109/SSDM.1994.336956.

M. Stonebraker, editor. 1994b. *Readings in Database Systems*, 2. Morgan Kaufmann.

M. Stonebraker. 1994c. Legacy systems—the Achilles heel of downsizing (panel). In *Proc. 3rd International Conference on Parallel and Distributed Information Systems*, p. 108.

M. Stonebraker. 1994d. In memory of Bob Kooi (1951-1993). *ACM SIGMOD Record*, 23(1): 3. DOI: 10.1145/181550.181551.

M. Stonebraker. 1995. An overview of the Sequoia 2000 project. *Digital Technical Journal*, 7(3). http://www.hpl.hp.com/hpjournal/dtj/vol7num3/vol7num3art3.pdf. 215, 255

M. Stonebraker. 1998. Are we working on the right problems? (panel). In *Proc. ACM SIGMOD International Conference on Management of Data*, p. 496. DOI: 10.1145/276304.276348.

M. Stonebraker. 2002. Too much middleware. *ACM SIGMOD Record*, 31(1): 97–106. DOI: 10.1145/507338.507362. 91

M. Stonebraker. 2003. Visionary: A next generation visualization system for databases. In *Proc. ACM SIGMOD International Conference on Management of Data*, p. 635. http://www.acm.org/sigmod/sigmod03/eproceedings/papers/ind00.pdf.

M. Stonebraker. 2004. Outrageous ideas and/or thoughts while shaving. In *Proc. 20th International Conference on Data Engineering*, p. 869. DOI: 10.1109/ICDE.2004.1320096.

M. Stonebraker. 2008a. Why did Jim Gray win the Turing Award? *ACM SIGMOD Record*, 37(2): 33–34. DOI: 10.1145/1379387.1379398.

M. Stonebraker. 2008b. Technical perspective—one size fits all: An idea whose time has come and gone. *Communications of the ACM*, 51(12): 76. DOI: 10.1145/1409360.1409379. 92

M. Stonebraker. 2009a. Stream processing. In L. Liu and M. T. Özsu, editors. *Encyclopedia of Database Systems*, pp. 2837–2838. Springer. DOI: 10.1007/978-0-387-39940-9_371.

M. Stonebraker. 2009b. A new direction for tpc? In *Proc. 1st TPC Technology Conference on Performance Evaluation and Benchmarking*, pp. 11–17. DOI: 10.1007/978-3-642-10424-4_2.

M. Stonebraker. 2010a. SQL databases v. nosql databases. *Communications of the ACM*, 53(4): 10–11. DOI: 10.1145/1721654.1721659. 50

M. Stonebraker. 2010b. In search of database consistency. *Communications of the ACM*, 53(10): 8–9. DOI: 10.1145/1831407.1831411.

M. Stonebraker. 2011a. Stonebraker on data warehouses. *Communications of the ACM*, 54(5): 10–11. DOI: 10.1145/1941487.1941491.

M. Stonebraker. 2011b. Stonebraker on nosql and enterprises. *Communications of the ACM*, 54(8): 10–11. DOI: 10.1145/1978542.1978546. 50

M. Stonebraker. 2012a. SciDB: An open-source DBMS for scientific data. *ERCIM News*, 2012(89). http://ercim-news.ercim.eu/en89/special/scidb-an-open-source-dbms-for-scientific-data.

M. Stonebraker. 2012b. New opportunities for new SQL. *Communications of the ACM*, 55(11): 10–11. DOI: 10.1145/2366316.2366319.

M. Stonebraker. 2013. We are under attack; by the least publishable unit. In *Proc. 6th Biennial Conference on Innovative Data Systems Research*. http://www.cidrdb.org/cidr2013/Talks/CIDR13_Gongshow16.ppt. 273

M. Stonebraker. 2015a. Turing lecture. In *Proc. Federated Computing Research Conference*, pp. 2–2. DOI: 10.1145/2820468.2820471.

M. Stonebraker. 2015b. What it's like to win the Turing Award. *Communications of the ACM*, 58(11): 11. xxxi, xxxiii

M. Stonebraker. 2015c. The Case for Polystores. ACM SIGMOD Blog, http://wp.sigmod.org/?p=1629. 370, 371

M. Stonebraker. 2016. The land sharks are on the squawk box. *Communications of the ACM*, 59(2): 74–83. DOI: 10.1145/2869958. 50, 129, 139, 260, 319

M. Stonebraker. 2018. My top ten fears about the DBMS field. In *Proc. 34th International Conference on Data Engineering*, pp. 24–28.

M. Sullivan and M. Stonebraker. 1991. Using write protected data structures to improve software fault tolerance in highly available database management systems. In *Proc. 17th International Conference on Very Large Data Bases*, pp. 171–180. http://www.vldb.org/conf/1991/P171.pdf.

R. Taft, E. Mansour, M. Serafini, J. Duggan, A. J. Elmore, A. Aboulnaga, A. Pavlo, and M. Stonebraker. 2014a. E-store: Fine-grained elastic partitioning for distributed transaction processing. *Proc. VLDB Endowment*, 8(3): 245–256. http://www.vldb.org/pvldb/vol8/p245-taft.pdf. 188, 251

R. Taft, M. Vartak, N. R. Satish, N. Sundaram, S. Madden, and M. Stonebraker. 2014b. Genbase: a complex analytics genomics benchmark. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 177–188. DOI: 10.1145/2588555.2595633.

R. Taft, W. Lang, J. Duggan, A. J. Elmore, M. Stonebraker, and D. J. DeWitt. 2016. Step: Scalable tenant placement for managing database-as-a-service deployments. In *Proc. 7th ACM Symposium on Cloud Computing*, pp. 388–400. DOI: 10.1145/2987550.2987575.

R. Taft, N. El-Sayed, M. Serafini, Y. Lu, A. Aboulnaga, M. Stonebraker, R. Mayerhofer, and F. Andrade. 2018. P-Store: an elastic database system with predictive provisioning. In *Proc. ACM SIGMOD International Conference on Management of Data*. 188

W. Tao, D. Deng, and M. Stonebraker. 2017. Approximate string joins with abbreviations. *Proc. VLDB Endowment*, 11(1): 53–65.

N. Tatbul, U. Çetintemel, S. B. Zdonik, M. Cherniack, and M. Stonebraker. 2003. Load shedding in a data stream manager. In *Proc. 29th International Conference on Very Large Data Bases*, pp. 309–320. http://www.vldb.org/conf/2003/papers/S10P03.pdf. 228, 229

N. Tatbul, S. Zdonik, J. Meehan, C. Aslantas, M. Stonebraker, K. Tufte, C. Giossi, and H. Quach. 2015. Handling shared, mutable state in stream processing with correctness guarantees. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 38(4): 94–104. http://sites.computer.org/debull/A15dec/p94.pdf.

T. J. Teorey, J. W. DeHeus, R. Gerritsen, H. L. Morgan, J. F. Spitzer, and M. Stonebraker. 1976. SIGMOD (paper session). In *Proc. 1976 ACM Annual Conference*, p. 275. DOI: 10.1145/800191.805596.

M. S. Tuttle, S. H. Brown, K. E. Campbell, J. S. Carter, K. Keck, M. J. Lincoln, S. J. Nelson, and M. Stonebraker. 2001a. The semantic web as "perfection seeking": A view from drug terminology. In *Proc. 1st Semantic Web Working Symposium*, pp. 5–16. http://www.semanticweb.org/SWWS/program/full/paper49.pdf.

M. S. Tuttle, S. H. Brown, K. E. Campbell, J. S. Carter, K. Keck, M. J. Lincoln, S. J. Nelson, and M. Stonebraker. 2001b. The semantic web as "perfection seeking": A view from drug terminology. In I. F. Cruz, S. Decker, J. Euzenat, and D. L. McGuinness, editors, *The Emerging Semantic Web, Selected Papers from the 1st Semantic Web Working Symposium*, volume 75 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

J. Widom, A. Bosworth, B. Lindsey, M. Stonebraker, and D. Suciu. 2000. Of XML and databases (panel session): Where's the beef? In *Proc. ACM SIGMOD International Conference on Management of Data*, p. 576. DOI: 10.1145/335191.335476.

M. W. Wilkins, R. Berlin, T. Payne, and G. Wiederhold. 1985. Relational and entity-relationship model databases and specialized design files in vlsi design. In *Proc. 22nd ACM/IEEE Design Automation Conference*, pp. 410–416.

J. Woodfill and M. Stonebraker. 1983. An implementation of hypothetical relations. In *Proc. 9th International Conference on Very Data Bases*, pp. 157–166. http://www.vldb.org/conf/1983/P157.pdf.

A. Woodruff and M. Stonebraker. 1995. Buffering of intermediate results in dataflow diagrams. In *Proc. IEEE Symposium on Visual Languages*, p. 187. DOI: 10.1109/VL.1995.520808.

A. Woodruff and M. Stonebraker. 1997. Supporting fine-grained data lineage in a database visualization environment. In *Proc. 13th International Conference on Data Engineering*, pp. 91–102. DOI: 10.1109/ICDE.1997.581742.

A. Woodruff, P. Wisnovsky, C. Taylor, M. Stonebraker, C. Paxson, J. Chen, and A. Aiken. 1994. Zooming and tunneling in Tioga: Supporting navigation in multimedia space. In *Proc. IEEE Symposium on Visual Languages*, pp. 191–193. DOI: 10.1109/VL.1994.363622.

A. Woodruff, A. Su, M. Stonebraker, C. Paxson, J. Chen, A. Aiken, P. Wisnovsky, and C. Taylor. 1995. Navigation and coordination primitives for multidimensional visual browsers. In *Proc. IFIP WG 2.6 3rd Working Conference Visual Database Systems*, pp. 360–371. DOI: 10.1007/978-0-387-34905-3_23.

A. Woodruff, J. A. Landay, and M. Stonebraker. 1998a. Goal-directed zoom. In *CHI '98 Conference Summary on Human Factors in Computing Systems*, pp. 305–306. DOI: 10.1145/286498.286781.

A. Woodruff, J. A. Landay, and M. Stonebraker. 1998b. Constant density visualizations of non-uniform distributions of data. In *Proc. 11th Annual ACM Symposium on User Interface Software and Technology*, pp. 19–28. DOI: 10.1145/288392.288397.

A. Woodruff, J. A. Landay, and M. Stonebraker. 1998c. Constant information density in zoomable interfaces. In *Proc. Working Conference on Advanced Visual Interfaces*, pp. 57–65. DOI: 10.1145/948496.948505.

A. Woodruff, J. A. Landay, and M. Stonebraker. 1999. VIDA: (visual information density adjuster). In *CHI '99 Extended Abstracts on Human Factors in Computing Systems*, pp. 19–20. DOI: 10.1145/632716.632730.

A. Woodruff, C. Olston, A. Aiken, M. Chu, V. Ercegovac, M. Lin, M. Spalding, and M. Stonebraker. 2001. Datasplash: A direct manipulation environment for programming semantic zoom visualizations of tabular data. *Journal of Visual Languages and Computing*, 12(5): 551–571. DOI: 10.1006/jvlc.2001.0219.

E. Wu, S. Madden, and M. Stonebraker. 2012. A demonstration of dbwipes: Clean as you query. *Proc. VLDB Endowment*, 5(12): 1894–1897. DOI: 10.14778/2367502.2367531.

E. Wu, S. Madden, and M. Stonebraker. 2013. Subzero: A fine-grained lineage system for scientific databases. In *Proc. 29th International Conference on Data Engineering*, pp. 865–876. DOI: 10.1109/ICDE.2013.6544881.

X. Yu, G. Bezerra, A. Pavlo, S. Devadas, and M. Stonebraker. 2014. Staring into the abyss: An evaluation of concurrency control with one thousand cores. *Proc. VLDB Endowment*, 8(3): 209–220. http://www.vldb.org/pvldb/vol8/p209-yu.pdf.

K. Yu, V. Gadepally, and M. Stonebraker. 2017. Database engine integration and performance analysis of the BigDAWG polystore system. *High Performance Extreme Computing Conference (HPEC)*. IEEE, 2017. DOI: 10.1109/HPEC.2017.8091081. 376

S. B. Zdonik, M. Stonebraker, M. Cherniack, U. Çetintemel, M. Balazinska, and H. Balakrishnan. 2003. The aurora and medusa projects. *Quarterly Bulletin IEEE Technical Committee on Data Engineering*, 26(1): 3–10. http://sites.computer.org/debull/A03mar/zdonik.ps. 228, 324

References

D. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik. 2005. The design of the Borealis stream processing engine. *Proc. of the 2nd Biennial Conference on Innovative Data Systems Research* (*CIDR*'05), Asilomar, CA, January. 228

Z. Abedjan, L. Golab, and F. Naumann. August 2015. Profiling relational data: a survey. *The VLDB Journal,* 24(4): 557–581. DOI: DOI: 10.1007/s00778-015-0389-y. 297

ACM. 2015a. Announcement: Michael Stonebraker, Pioneer in Database Systems Architecture, Receives 2014 ACM Turing Award. http://amturing.acm.org/award_winners/stonebraker_1172121.cfm. Accessed February 5, 2018.

ACM. March 2015b. Press Release: MIT's Stonebraker Brought Relational Database Systems from Concept to Commercial Success, Set the Research Agenda for the Multibillion-Dollar Database Field for Decades. http://sigmodrecord.org/publications/sigmodRecord/1503/pdfs/04_announcements_Stonebraker.pdf. Accessed February 5, 2018.

ACM. 2016. A.M. Turing Award Citation and Biography. http://amturing.acm.org/award_winners/stonebraker_1172121.cfm. Accessed September 24, 2018. xxxi

Y. Ahmad, B. Berg, U. Çetintemel, M. Humphrey, J. Hwang, A. Jhingran, A. Maskey, O. Papaemmanouil, A. Rasin, N. Tatbul, W. Xing, Y. Xing, and S. Zdonik. June 2005. Distributed operation in the Borealis Stream Processing Engine. Demonstration, *ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*. Baltimore, MD. Best Demonstration Award. 230, 325

M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson. 1976. System R: relational approach to database management. *ACM Transactions on Database Systems*, 1(2): 97–137. DOI: 10.1145/320455.320457. 397

P. Bailis, E. Gan, S. Madden, D. Narayanan, K. Rong, and S. Suri. 2017. Macrobase: Prioritizing attention in fast data. *Proc. of the 2017 ACM International Conference on Management of Data*. ACM. DOI: 10.1145/3035918.3035928. 374

Berkeley Software Distribution. n.d. In Wikipedia. http://en.wikipedia.org/wiki/Berkeley_Software_Distribution. Last accessed March 1, 2018. 109

G. Beskales, I.F. Ilyas, L. Golab, and A. Galiullin. 2013. On the relative trust between inconsistent data and inaccurate constraints. *Proc. of the IEEE International Conference on Data Engineering*, *ICDE 2013*, pp. 541–552. Australia. DOI: 10.1109/ICDE.2013.6544854. 270

L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R. C. Whaley. 2017. *ScaLAPACK Users' Guide*. Society for Industrial and Applied Mathematics http://netlib.org/scalapack/slug/index.html. Last accessed December 31, 2017. 258

D. Bitton, D. J. DeWitt, and C. Turbyfill. 1983. Benchmarking database systems—a systematic approach. Computer Sciences Technical Report #526, University of Wisconsin. http://minds.wisconsin.edu/handle/1793/58490, 111

P. A. Boncz, M. L. Kersten, and S. Manegold. December 2008. Breaking the memory wall in MonetDB. *Communications of the ACM*, 51(12): 77–85. DOI: 10.1145/1409360.1409380. 151

M. L. Brodie. June 2015. Understanding data science: an emerging discipline for data-intensive discovery. In S. Cutt, editor, *Getting Data Right: Tackling the Challenges of Big Data Volume and Variety*. O'Reilly Media, Sebastopol, CA. 291

Brown University, Department of Computer Science. Fall 2002. Next generation stream-based applications. *Conduit Magazine*, 11(2). https://cs.brown.edu/about/conduit/conduit_v11n2.pdf. Last accessed May 14, 2018. 322

BSD licenses. n.d. In Wikipedia. http://en.wikipedia.org/wiki/BSD_licenses. Last accessed March 1, 2018. 109

M. Cafarella and C. Ré. April 2018. The last decade of database research and its blindingly bright future. or Database Research: A love song. DAWN Project, Stanford University. http://dawn.cs.stanford.edu/2018/04/11/db-community/. 6

M. J. Carey, D. J. DeWitt, M. J. Franklin, N. E Hall, M. L. McAuliffe, J. F. Naughton, D. T. Schuh, M. H. Solomon, C. K. Tan, O. G. Tsatalos, S. J. White, and M. J. Zwilling. 1994. Shoring up persistent applications. *Proc. of the 1994 ACM SIGMOD international conference on Management of data (SIGMOD '94)*, 383–394. DOI: 10.1145/191839.191915. 152

M. J. Carey, D. J. Dewitt, M. J. Franklin, N.E. Hall, M. L. McAuliffe, J. F. Naughton, D. T. Schuh, M. H. Solomon, C. K. Tan, O. G. Tsatalos, S. J. White, and M. J. Zwilling. 1994. Shoring up persistent applications. In *Proc. of the 1994 ACM SIGMOD International Conference on Management of Data (SIGMOD '94)*, pp. 383–394. DOI: 10.1145/191839.191915. 336

M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. E. Cody, R. Fagin, M. Flickner, A. W. Luniewski, W. Niblack, and D. Petkovic. 1995. Towards heterogeneous multimedia information systems: The garlic approach. In *Research Issues in Data Engineering, 1995: Distributed Object Management, Proceedings*, pp. 124–131. IEEE. DOI: 10.1109/RIDE.1995.378736. 284

CERN. http://home.cern/about/computing. Last accessed December 31, 2017.

D. D. Chamberlin and R. F. Boyce. 1974. SEQUEL: A structured English query language. In *Proc. of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description,*

Access and Control (SIGFIDET '74), pp. 249–264. ACM, New York. DOI: 10.1145/800296.811515. 404, 407

D. D. Chamberlin, M. M. Astrahan, K. P. Eswaran, P. P. Griffiths, R. A. Lorie, J. W. Mehl, P. Reisner, and B. W. Wade. 1976. SEQUEL 2: a unified approach to data definition, manipulation, and control. *IBM Journal of Research and Development*, 20(6): 560–575. DOI: 10.1147/rd.206.0560. 398

S. Chandrasekaran, O, Cooper, A. Deshpande, M.J. Franklin, J.M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah. 2003. TelegraphCQ: Continuous dataflow processing for an uncertain world. *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD '03)*, pp. 668–668. ACM, New York. DOI :10.1145/872757.872857. 231

J. Chen, D.J. DeWitt, F. Tian, and Y. Wang. 2000. NiagaraCQ: A scalable continuous query system for Internet databases. *Proc. of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD '00)*, pp. 379–390. ACM, New York. DOI 10.1145/342009.335432. 231

M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Çetintemel, Y. Xing, and S. Zdonik. 2003. Scalable distributed stream processing. *Proc. of the First Biennial Conference on Innovative Database Systems* (*CIDR*'03), Asilomar, CA, January. 228

C. M. Christensen. 1997. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Harvard Business School Press, Boston, MA. 100

X. Chu, I. F. Ilyas, and P. Papotti. 2013a. Holistic data cleaning: Putting violations into context. *Proc. of the IEEE International Conference on Data Engineering*, *ICDE 2013*, pp. 458–469. Australia. DOI: 10.1109/ICDE.2013.6544847 270, 297

X. Chu, I. F. Ilyas, and P. Papotti. 2013b. Discovering denial constraints. *Proc. of the VLDB Endowment*, *PVLDB* 6(13): 1498–1509. DOI: 10.14778/2536258.2536262. 270

X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proc. of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15)*, pp. 1247–1261. ACM, New York. DOI: 10.1145/2723372.2749431. 297

P. J. A. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice. 2009. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research* 38.6: 1767–1771. DOI: 10.1093/nar/gkp1137. 374

E. F. Codd. June 1970. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6): 377–387. DOI: 10.1145/362384.362685. 42, 98, 166, 397, 404, 405, 407

M. Collins. 2016. Thomson Reuters uses Tamr to deliver better connected content at a fraction of the time and cost of legacy approaches. Tamr blog, July 28. https://www.tamr.com/video/thomson-reuters-uses-tamr-deliver-better-connected-content-fraction-time-cost-legacy-approaches/. Last accessed January 24, 2018. 275

G. Copeland and D. Maier. 1984. Making smalltalk a database system. *Proc. of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD '84)*, pp. 316–325. ACM, New York. DOI: 10.1145/602259.602300. 111

C. Cranor, T. Johnson, V. Shkapenyuk, and O. Spatscheck. 2003. Gigascope: A stream database for network applications. *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD '03)*, pp. 647–651. ACM, New York. DOI :10.1145/872757.872838. 231

A. Crotty, A. Galakatos, K. Dursun, T. Kraska, U. Cetintemel, and S. Zdonik. 2015. Tupleware: "Big Data, Big Analytics, Small Clusters." *CIDR*. DOI: 10.1.1.696.32. 374

M. Dallachiesa, A. Ebaid, A. Eldawi, A. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. 2013. NADEEF, a commodity data cleaning system. *Proc. of the 2013 ACM SIGMOD Conference on Management of Data*, pp. 541–552. New York. http://dx.doi.org/10.1145/2463676.2465327. 270, 297

T. Dasu and J. M. Loh. 2012. Statistical distortion: Consequences of data cleaning. *PVLDB*, 5(11): 1674–1683. DOI: 10.14778/2350229.2350279. 297

C. J. Date and E. F. Codd. 1975. The relational and network approaches: Comparison of the application programming interfaces. In *Proc. of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control: Data Models: Data-Structure-Set Versus Relational (SIGFIDET '74)*, pp. 83–113. ACM, New York. DOI: 10.1145/800297.811534. 405

D. J. DeWitt. 1979a. Direct a multiprocessor organization for supporting relational database management systems. *IEEE Transactions of Computers,* 28(6), 395–406. DOI: 10.1109/TC.1979.1675379. 109

D. J. DeWitt. 1979b. Query execution in DIRECT. In *Proc. of the 1979 ACM SIGMOD International Conference on Management of Data (SIGMOD '79)*, pp. 13–22. ACM, New York. DOI: 10.1145/582095.582098. 109

D. J. DeWitt, R. H. Gerber, G. Graefe, M. L. Heytens, K. B. Kumar, and M. Muralikrishna. 1986. GAMMA—a high performance dataflow database machine. *Proc. of the 12th International Conference on Very Large Data Bases (VLDB '86)*, W. W. Chu, G. Gardarin, S. Ohsuga, and Y. Kambayashi, editors, pp. 228–237. Morgan Kaufmann Publishers Inc., San Francisco, CA. 111

D. J. DeWitt, S. Ghandeharizadeh, D. A. Schneider, A. Bricker, H.-I. Hsiao, and R. Rasmussen. March 1990. The Gamma database machine project. *IEEE Transactions on Knowledge and Data Engineering,* 2(1): 44–62. DOI: 10.1109/69.50905. 151, 400

D. DeWitt and J. Gray. June 1992. Parallel database systems: the future of high performance database systems. *Communications of the ACM,* 35(6): 85–98. DOI: 10.1145/129888.129894. 199

D. J. DeWitt, A. Halverson, R. Nehme, S. Shankar, J. Aguilar-Saborit, A. Avanes, M. Flasza, and J. Gramling. 2013. Split query processing in polybase. *Proc. of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13)*, pp. 1255–1266. ACM, New York. 284

C. Diaconu, C. Freedman, E. Ismert, P-A. Larson, P. Mittal, R. Stonecipher, N. Verma, and M. Zwilling. 2013. Hekaton: SQL server's memory-optimized OLTP engine. In *Proc. of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13)*, pp. 1243–1254. ACM, New York. DOI: 10.1145/2463676.2463710.

K. P. Eswaran, J. N. Gray, R. A. Lorie, and I. L. Traiger. November 1976. The notions of consistency and predicate locks in a database system. *Communications of the ACM,* 19(11): 624–633. DOI: 10.1145/360363.360369. 114

W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. April 2012. Towards certain fixes with editing rules and master data. *The VLDB Journal*, 21(2): 213–238. DOI: 10.1007/s00778-011-0253-7. 297

D. Fogg. September 1982. Implementation of domain abstraction in the relational database system INGRES. Master of Science Report, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA. 201

T. Flory, A. Robbin, and M. David. May 1988. Creating SIPP longitudinal analysis files using a relational database management system. CDE Working Paper No. 88-32, Institute for Research on Poverty, University of Wisconsin-Madison, Madison, WI. 197

V. Gadepally, J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, L. Edwards, M. Hubbell, P. Michaleas, J. Mullen, A. Prout, A. Rosa, C. Yee, and A. Reuther. 2015. D4M: Bringing associative arrays to database engines. *High Performance Extreme Computing Conference (HPEC)*. IEEE, 2015. DOI: 10.1109/HPEC.2015.7322472. 370

V. Gadepally, K. O'Brien, A. Dziedzic, A. Elmore, J. Kepner, S. Madden, T. Mattson, J. Rogers, Z. She, and M. Stonebraker. September 2017. BigDAWG Version 0.1. *IEEE High Performance Extreme*. DOI: 10.1109/HPEC.2017.8091077. 288, 369

J. Gantz and D. Reinsel. 2013. The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East—United States, IDC, February. 5

L. Gerhardt, C. H. Faham, and Y. Yao. 2015. Accelerating scientific analysis with SciDB. *Journal of Physics: Conference Series*, 664(7). 268

B. Grad. 2007. Oral history of Michael Stonebraker, Transcription. Recorded: August 23, 2007. Computer History Museum, Moultonborough, NH. http://archive.computerhistory.org/resources/access/text/2012/12/102635858-05-01-acc.pdf. Last accessed April 8, 2018. 42, 43, 44, 98

A. Guttman. 1984. R-trees: a dynamic index structure for spatial searching. In *Proc. of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD '84)*, pp. 47–57. ACM, New York. DOI: 10.1145/602259.602266. 205

L. M. Haas, J. C. Freytag, G. M. Lohman, and H. Pirahesh. 1989. Extensible query processing in starburst. In *Proc. of the 1989 ACM SIGMOD International Conference on Management of Data (SIGMOD '89)*, pp. 377–388. ACM, New York. DOI: 10.1145/67544.66962. 399

D. Halperin, V. Teixeira de Almeida, L. L. Choo, S. Chu, P. Koutris, D. Moritz, J. Ortiz, V. Ruamviboonsuk, J. Wang, A. Whitaker. 2014. Demonstration of the Myria big data management service. *Proc. of the 2014 ACM SIGMOD International Conference*

on Management of Data (SIGMOD '14), p. 881–884. ACM, New York. DOI: 10.1145/2588555.2594530. 284, 370

B. Haynes, A. Cheung, and M. Balazinska. 2016. PipeGen: Data pipe generator for hybrid analytics. *Proc. of the Seventh ACM Symposium on Cloud Computing (SoCC '16)*, M. K. Aguilera, B. Cooper, and Y. Diao, editors, pp. 470–483. ACM, New York. DOI: 10.1145/2987550.2987567. 287

M. A. Hearst. 2009. *Search user interfaces*. Cambridge University Press, New York. 394

J. M. Hellerstein, J. F. Naughton, and A. Pfeffer. 1995. Generalized search trees for database systems. In *Proc. of the 21th International Conference on Very Large Data Bases (VLDB '95)*, pp. 562–573. Morgan Kaufmann Publishers Inc., San Francisco, CA. http://dl.acm.org/citation.cfm?id=645921.673145. 210

J. M. Hellerstein, E. Koutsoupias, D. P. Miranker, C. H. Papadimitriou, V. Samoladas. 2002. On a model of indexability and its bounds for range queries, *Journal of the ACM (JACM)*, 49.1: 35–55. DOI: 10.1145/505241.505244. 210

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

S. Idreos, F. Groffen, N. Nes, S. Manegold, S. K. Mullender, and M. L. Kersten. 2012. MonetDB: two decades of research in column-oriented database architectures. *IEEE Data Engineering Bulletin*, 35(1): 40–45. 258

N. Jain, S. Mishra, A. Srinivasan, J. Gehrke, J. Widom, H. Balakrishnan, U. Çetintemel, M. Cherniack, R. Tibbetts, and S. Zdonik. 2008. Towards a streaming SQL standard. *Proc. VLDB Endowment*, pp. 1379–1390. August 1–2. DOI: 10.14778/1454159.1454179. 229

A. E. W. Johnson, T. J. Pollard, L. Shen, L. H. Lehman, M. Feng, M. Ghassemi, B. E. Moody, P. Szolovits, L. A. G. Celi, and R. G. Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data* 3: 160035 DOI: 10.1038/sdata.2016.35. 370

V. Josifovski, P. Schwarz, L. Haas, and E. Lin. 2002. Garlic: a new flavor of federated query processing for DB2. In *Proc. of the 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD '02)*, pp. 524–532. ACM, New York. DOI: 10.1145/564691.564751. 401

J. W. Josten, C. Mohan, I. Narang, and J. Z. Teng. 1997. DB2's use of the coupling facility for data sharing. *IBM Systems Journal*, 36(2): 327–351. DOI: 10.1147/sj.362.0327. 400

S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*, pp. 3363–3372. ACM, New York. DOI: 10.1145/1978942.1979444. 297

R. Katz. editor. June 1982. Special issue on design data management. *IEEE Database Engineering Newsletter*, 5(2). 200

J. Kepner, V. Gadepally, D. Hutchison, H. Jensen, T. Mattson, S. Samsi, and A. Reuther. 2016. Associative array model of SQL, NoSQL, and NewSQL Databases. *IEEE High*

Performance Extreme Computing Conference (HPEC) 2016, Waltham, MA, September 13–15. DOI: 10.1109/HPEC.2016.7761647. 289

V. Kevin and M. Whitney. 1974. Relational data management implementation techniques. In *Proc. of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control (SIGFIDET '74)*, pp. 321–350. ACM, New York. DOI: 10.1145/800296.811519 404

Z. Khayyat, I.F. Ilyas, A. Jindal, S. Madden, M. Ouzzani, P. Papotti, J.-A. Quiané-Ruiz, N. Tang, and S. Yin. 2015. Bigdansing: A system for big data cleansing. In *Proc. of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15),* pp. 1215–1230. ACM, New York. DOI: 10.1145/2723372.2747646. 297

R. Kimball and M. Ross. 2013. *The Data Warehouse Toolkit*. John Wiley & Sons, Inc. https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/books/. Last accessed March 2, 2018. 337

M. Kornacker, C. Mohan, and J.M. Hellerstein. 1997. Concurrency and recovery in generalized search trees. In *Proc. of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD '97)*, pp. 62–72. ACM, New York. DOI: 10.1145/253260.253272. 210

A. Lamb, M. Fuller, R. Varadarajan, N. Tran, B. Vandiver, L. Doshi, and C. Bear. August 2012. The Vertica Analytic Database: C-Store 7 years later. *Proc. VLDB Endowment*, 5(12): 1790–1801. DOI: 10.14778/2367502.2367518. 333, 336

L. Lamport. 2001. Paxos Made Simple. http://lamport.azurewebsites.net/pubs/paxos-simple.pdf. Last accessed December 31, 2017. 258

D. Laney. 2001. 3D data management: controlling data volume, variety and velocity. META Group Research, February 6. https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf. Last accessed April 22, 2018. 357

P-A. Larson, C. Clinciu, E.N. Hanson, A. Oks, S.L. Price, S. Rangarajan, A. Surna, and Q. Zhou. 2011. SQL server column store indexes. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOD '11)*, pp. 1177–1184. ACM, New York. DOI: 10.1145/1989323.1989448.

J. LeFevre, J. Sankaranarayanan, H. Hacigumus, J. Tatemura, N. Polyzotis, and M. J. Carey. 2014. MISO: Souping up big data query processing with a multistore system. *Proc. of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*, pp. 1591–1602. ACM, New York. DOI: 10.1145/2588555.2588568. 284

B. G. Lindsay. 1987. A retrospective of R*: a distributed database management system. In *Proc. of the IEEE*,75(5): 668–673. DOI: 10.1109/PROC.1987.13780. 400

B. Liskov and S.N. Zilles. 1974. Programming with abstract data types. *SIGPLAN Notices,* 9(4): 50–59. DOI: 10.1145/942572.807045. 88

S. Marcin and A. Csillaghy. 2016. Running scientific algorithms as array database operators: Bringing the processing power to the data. *2016 IEEE International Conference on Big Data*. pp. 3187–3193. DOI: 10.1109/BigData.2016.7840974. 350

T. Mattson, V. Gadepally, Z. She, A. Dziedzic, and J. Parkhurst. 2017. Demonstrating the BigDAWG polystore system for ocean metagenomic analysis. *CIDR'17* Chaminade, CA. http://cidrdb.org/cidr2017/papers/p120-mattson-cidr17.pdf. 288, 374

J. Meehan, C. Aslantas, S. Zdonik, N. Tatbul, and J. Du. 2017. Data ingestion for the connected world. *Conference on Innovative Data Systems Research (CIDR'17)*, Chaminade, CA, January. 376

A. Metaxides, W. B. Helgeson, R. E. Seth, G. C. Bryson, M. A. Coane, D. G. Dodd, C. P. Earnest, R. W. Engles, L. N. Harper, P. A. Hartley, D. J. Hopkin, J. D. Joyce, S. C. Knapp, J. R. Lucking, J. M. Muro, M. P. Persily, M. A. Ramm, J. F. Russell, R. F. Schubert, J. R. Sidlo, M. M. Smith, and G. T. Werner. April 1971. *Data Base Task Group Report to the CODASYL Programming Language Committee*. ACM, New York. 43

C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz. 1992. ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Transactions on Database Systems*, 17(1), 94–162. DOI: 10.1145/128765.128770. 402

R. Motwani, J. Widom, A. Arasu B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. 2003. Query processing, approximation, and resource management in a data stream management system. *Proc. of the First Biennial Conference on Innovative Data Systems Research (CIDR)*, January. 229, 231

A. Oloso, K-S Kuo, T. Clune, P. Brown, A. Poliakov, H. Yu. 2016. Implementing connected component labeling as a user defined operator for SciDB. *Proc. of 2016 IEEE International Conference on Big Data (Big Data)*. Washington, DC. DOI: 10.1109/BigData.2016.7840945. 263, 350

M. A. Olson. 1993. The design and implementation of the inversion file system. *USENIX* Winter. http://www.usenix.org/conference/usenix-winter-1993-conference/presentation/design-and-implementation-inversion-file-syste. Last accessed January 22, 2018. 215

J. C. Ong. 1982. Implementation of abstract data types in the relational database system INGRES, Master of Science Report, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, September 1982. 201

A. Palmer. 2013. Culture matters: Facebook CIO talks about how well Vertica, Facebook people mesh. *Koa Labs Blog*, December 20. http://koablog.wordpress.com/2013/12/20/culture-matters-facebook-cio-talks-about-how-well-vertica-facebook-people-mesh. Last accessed March 14, 2018. 132, 133

A. Palmer. 2015a. The simple truth: happy people, healthy company. *Tamr Blog*, March 23. http://www.tamr.com/the-simple-truth-happy-people-healthy-company/. Last accessed March 14, 2018. 138

A. Palmer. 2015b. Where the red book meets the unicorn, *Xconomy*, June 22. http://www.xconomy.com/boston/2015/06/22/where-the-red-book-meets-the-unicorn/ Last accessed March 14, 2018. 130

A. Pavlo and M. Aslett. September 2016. What's really new with NewSQL? *ACM SIGMOD Record*, 45(2): 45–55. DOI: DOI: 10.1145/3003665.3003674. 246

G. Press. 2016. Cleaning big data: most time-consuming, least enjoyable data science task, survey says. *Forbes*, May 23. https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#79e14e326f63. 357

N. Prokoshyna, J. Szlichta, F. Chiang, R. J. Miller, and D. Srivastava. 2015. Combining quantitative and logical data cleaning. *PVLDB*, 9(4): 300–311. DOI: 10.14778/2856318.2856325. 297

E. Ryvkina, A. S. Maskey, M. Cherniack, and S. Zdonik. 2006. Revision processing in a stream processing engine: a high-level design. *Proc. of the 22nd International Conference on Data Engineering* (*ICDE*'06), pp. 141–. Atlanta, GA, April. IEEE Computer Society, Washington, DC. DOI: 10.1109/ICDE.2006.130. 228

C. Saracco and D. Haderle. 2013. The history and growth of IBM's DB2. *IEEE Annals of the History of Computing*, 35(2): 54–66. DOI: 10.1109/MAHC.2012.55. 398

N. Savage. May 2015. Forging relationships. *Communications of the ACM*, 58(6): 22–23. DOI: 10.1145/2754956.

M. C. Schatz and B. Langmead. 2013. The DNA data deluge. *IEEE Spectrum Magazine*. https://spectrum.ieee.org/biomedical/devices/the-dna-data-deluge. 354

Z. She, S. Ravishankar, and J. Duggan. 2016. BigDAWG polystore query optimization through semantic equivalences. *High Performance Extreme Computing Conference (HPEC)*. IEEE, 2016. DOI: :10.1109/HPEC.2016.7761584. 373

SIGFIDET panel discussion. 1974. In *Proc. of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control: Data Models: Data-Structure-Set Versus Relational (SIGFIDET '74)*, pp. 121–144. ACM, New York. DOI: 10.1145/800297.811534. 404

R. Snodgrass. December 1982. Monitoring distributed systems: a relational approach. Ph.D. Dissertation, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 197

A. Szalay. June 2008. The Sloan digital sky survey and beyond. *ACM SIGMOD Record*, 37(2): 61–66. 255

Tamr. 2017. Tamr awarded patent for enterprise-scale data unification system. Tamr blog. February 9 2017. https://www.tamr.com/tamr-awarded-patent-enterprise-scale-data-unification-system-2/. Last accessed January 24, 2018. 275

R. Tan, R. Chirkova, V. Gadepally, and T. Mattson. 2017. Enabling query processing across heterogeneous data models: A survey. *IEEE Big Data Workshop: Methods to Manage Heterogeneous Big Data and Polystore Databases*, Boston, MA. DOI: 10.1109/BigData.2017.8258302. 284, 376

N. Tatbul and S. Zdonik. 2006. Window-aware Load Shedding for Aggregation Queries over Data Streams. In *Proc. of the 32nd International Conference on Very Large Databases (VLDB'06)*, Seoul, Korea. 228, 229

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

R. P. van de Riet. 1986. Expert database systems. In *Future Generation Computer Systems*, 2(3): 191–199, DOI: 10.1016/0167-739X(86)90015-4. 407

M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis. September 2015. Seedb: Efficient data-driven visualization recommendations to support visual analytics. *PVLDB*, 8(13): 2182–2193. DOI: 10.14778/2831360.2831371. 297

B. Wallace. June 9, 1986. Data base tool links to remote sites. *Network World*. http://books.google.com/books?id=aBwEAAAAMBAJ&pg=PA49&lpg=PA49&dq=ingres+star&source=bl&ots=FSMIR4thMj&sig=S1fzaaOT5CHRq4cwbLFEQp4UYCs&hl=en&sa=X&ved=0ahUKEwjJ1J_NttvZAhUG82MKHco2CfAQ6AEIYzAP#v=onepage&q=ingres%20star&f=false. Last accessed March 14, 2018.305

J. Wang and N. J. Tang. 2014. Towards dependable data repairing with fixing rules. In *Proc. of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD ’14*), pp. 457–468. ACM, New York. DOI: 10.1145/2588555.2610494. 297

E. Wong and K. Youssefi. September 1976. Decomposition—a strategy for query processing. *ACM Transactions on Database Systems*, 1(3): 223–241. DOI: 10.1145/320473.320479. 196

E. Wu and S. Madden. 2013. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8): 553–564. DOI: 10.14778/2536354.2536356. 297

Y. Xing, S. Zdonik, and J.-H. Hwang. April 2005. Dynamic load distribution in the Borealis Stream Processor. *Proc. of the 21st International Conference on Data Engineering* (*ICDE*’05), Tokyo, Japan. DOI: 10.1109/ICDE.2005.53. 228, 230, 325

Index

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

Biographies

Editor

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

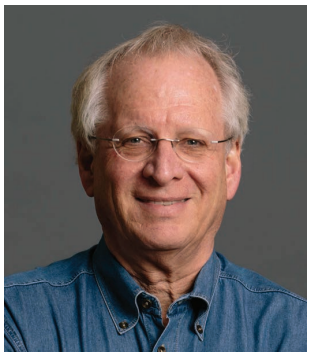
Magdalena Balazinska is a professor in the Paul G. Allen School of Computer Science and Engineering at the University of Washington and is the director of the University's eScience Institute. She's also director of the IGERT PhD Program in Big Data and Data Science and the associated Advanced Data Science 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.

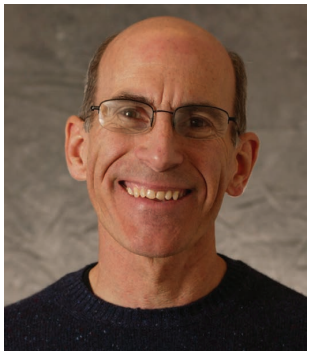
Janice L. Brown

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 coauthors 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.

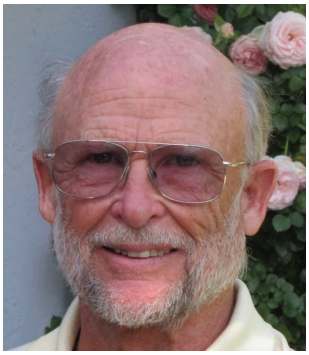
Vijay Gadepally

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

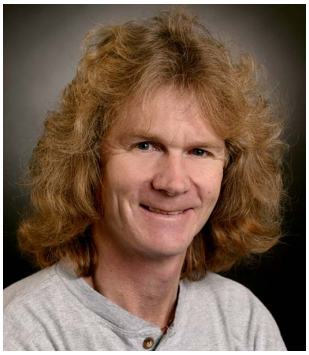
Nabil Hachem is currently Vice President, Head of Data Architecture, Technology, and Standards at MassMutual. He was formerly Global Head of Data Engineering at Novartis Institute for Biomedical Research, Inc. He also held senior data engineering posts at Vertica Systems, Inc., Infinity Pharmaceuticals, 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. inComputer Science from the University of Victoria and a Master's in Math, Computer Science from the University of Waterloo.

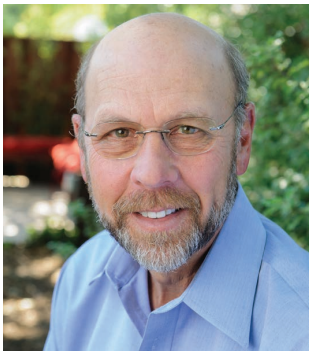
Stavros Harizopoulos

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 petabyt-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 hos 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.

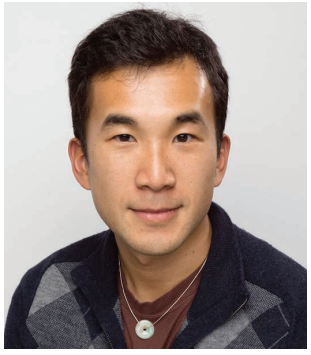
Jason Kinchen

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

life sciences. 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 Erdös number is 3.

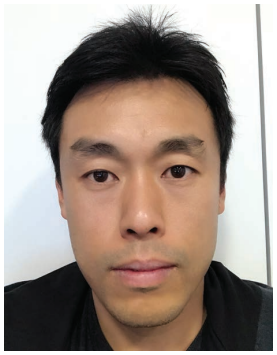
Lawrence A. Rowe

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 jtangoVC.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."

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.