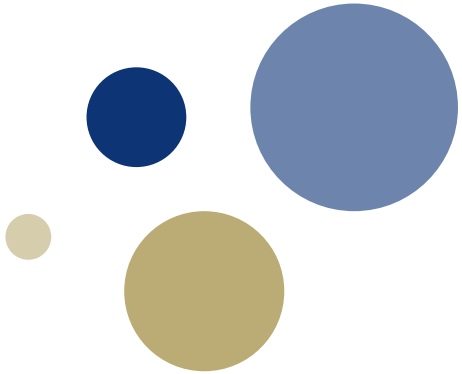
NTNU

Kunnskap for en bedre verden



What Goes Around Comes Around... And Around...

Michael Stonebraker and Andy Pavlo

Svein Erik Bratsberg, IDI/NTNU

Conclusions

- Efforts to replace SQL or the Relational Model (RM)
- Despite efforts to replace either of them.
- Instead, SQL absorbed the best ideas from these alternative approaches
- Done 20 years ago, the same goes around today.
- Mostly related to the “stickiness” of data
- Traditionally, SQL systems used for business data, but today used for any kind of data

Previous paper (Red book, ed 4, 2005)

- Hierarchical (e.g., IMS): late 1960s and 1970s
- Network (e.g., CODASYL): 1970s
- Relational: 1970s and early 1980s
- Entity-Relationship: 1970s
- Extended Relational: 1980s
- Semantic: late 1970s and 1980s
- Object-Oriented: late 1980s and early 1990s
- Object-Relational: late 1980s and early 1990s
- Semi-structured (e.g., XML): late 1990s and 2000s

New type of model for database systems

- MapReduce Systems
- Key-value Stores
- Document Databases
- Column Family / Wide-Column
- Text Search Engines
- Array Databases
- Vector Databases
- Graph Databases

New system architectures

- Columnar Systems
- Cloud Databases
- Data Lakes / Lake houses
- NewSQL Systems
- Hardware Accelerators
- Blockchain Databases

MapReduce Systems (1)

- Google created (2003) it to support internet crawl: SELECT map() FROM crawl_table GROUP BY reduce()
- map() and reduce() written in a programming language
- Yahoo created HADOOP (2005)
- GFS and HDFS file systems
- Many enterprises spent a lot of money on Hadoop clusters, only to find there was little interest in this functionality
- Google announced that they were moving their crawl processing from MR to BigTable. The reason was that Google needed to interactively update its crawl database in real-time but MR was a batch system (2014).

MapReduce Systems (2)

- There was considerable efforts to provide a SQL and RM interface on top of Hadoop, most notable was Meta's Hive, later replaced by Presto: SQL on Everything
- Presto: Connector API allows plugins to provide an I/O interface to dozens of data sources, including Hadoop data warehouses, RDBMSs, NoSQL systems, and stream processing systems.
- Hadoop died about a decade ago, Spark and Flink are new implementations with support for SQL

Key/Value Stores (1)

- (key, value)
- The value is typically an untyped array of bytes (i.e., a blob), and the DBMS is unaware of its contents.
- get/set/delete operations
- Caching and storing session data: Memcached and Redis.
- Amazon Dynamo KVstore (2007)
- Google LevelDB (2010) and Meta's RocksDB
- Some replaced the opaque value with a semi-structured value, such as a JSON document. Amazon's DynamoDB and Aerospike.

Key/Value Stores (2)

- MySQL was the first DBMS to expose an API that allowed developers to replace its default KV storage manager. This API enabled Meta to build RocksDB to replace InnoDB for its massive fleet of MySQL databases
- MongoDB discarded their ill-fated MMAP based storage manager in favor of WiredTiger's KV store (2014).
- MMAP:
  - Problems with transactional safety.
  - I/O stalls. Lack of async I/O.
  - Error handling is not modularized. SIGBUS signals.
  - Performance issues, e.g. no compression and TLB shootdowns
  - See Pavlo's Are You Sure You Want to Use MMAP in Your Database Management System? (CIDR 2022)

Document Databases (1)

- Each document contains a hierarchy of field/value pairs, where each field is identified by a name and a field's value can be either a scalar type, an array of values, or another document.
- { "name": "First Last", "orders": [
  { "id": 123, "items": [...] }, { "id": 456, "items": [...] },] }
- DBMS that stored records or documents as JSON, supported a lower-level API, and weak or non-existent transactions. Example: MongoDB
- Removes the impedance mismatch between how application OO code interacts with data
- Denormalization:

1. if the join is not one-to-many, then there will be duplicated data
2. prejoins are not necessarily faster than joins
3. there is no data independence.

Document Databases (2)

- Adding SQL and ACID to a NoSQL DBMS lowers their intellectual distance from RDBMSs
- Higher level languages are almost universally preferred to record-at-a-time notations as they require less code and provide greater data independence.
- But the optimizer remains the hardest part of building a DBMS.
- We suspect that this engineering burden was a contributing factor to why NoSQL systems originally chose to not support SQL

Column-Family Databases

- It is a reduction of the document data model that only supports one level of nesting instead of arbitrary nesting;
- Google's BigTable (2004), Cassandra and Hbase
- Lack of joins and secondary indexes.
- Cassandra replaced their Thrift-API with a SQL-like language called CQL
- HBase now recommends the Phoenix SQL-frontend

```
User1000 → { "name": "Alice",
             "accounts": [ 123, 456 ],
             "email": "xxx@xxx.edu" }
User1001 → { "name": "Bob",
             "email": [ "yyy@yyy.org", "zzz@zzz.com" ] }
```

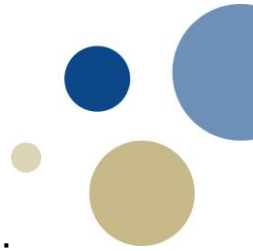
Text Search Engines

- The leading text search systems today include Elastic search and Solr, which both use Lucene (library)
- Good support for storing and indexing text data but offer none-to-limited transaction capabilities.
- Updates through merge of new index into existing indexes
- Often combined with a SQL database, but this creates some trouble.
- All the leading RDBMSs now support full-text search indexes
- They lack a standard interface in SQL

Array Databases

- “array” means all variants:
  - vectors (one dimension)
  - matrices (two dimensions)
  - tensors (three or more dimensions)
- E.g. geodata: (latitude, longitude, time, [vector-of-values])
- Genomic sequencing and computational fluid dynamics
- Arrays are also the core of most ML data sets (embeddings)
- Rasdaman and kdb+ are old systems.
- Newer array DBMSs include SciDB and TileDB.
- HDF5 and NetCDF are popular array file formats for scientific data
- Array DBMSs are a niche market that has only seen adoption in specific verticals
- SQL includes support for ordered arrays as first-class data types
- SQL:2023 allows SQL to represent arrays with arbitrary dimensions using integer-based coordinates

Vector Databases



- Interesting due to developers use them to store single-dimension embeddings generated from AI tools.
- Not to be confused with *vector query processing*, which is about the implementation of memory efficient processing by batching data.
- (title, date, author, [embedding-vector])
- The size of these embedding vectors range from 100s of dimensions to 1000s for highend models
- The key difference between vector and array DBMSs is their query patterns.
- The former are designed for similarity searches that find records whose vectors have the shortest distance to a given input vector in a highdimensional space
- Build indexes to accelerate approximate nearest neighbor (ANN) searches
- Pinecone, Milvus, and Weaviate
- Vector DBMSs are essentially document-oriented DBMSs with specialized ANN indexes.
- Oracle, SingleStore, Rockset, and Clickhouse added vectors often using open-source library (e.g., pgVector, DiskANN , FAISS)

Graph Databases

- Many applications use knowledge graphs to model semi-structured information.
- Social media applications inherently contain graph-oriented relationships (“likes”, “friend-of”)
- (1) resource description framework (RDF, triplestores)
- (2) property graphs
- Operational / OLTP workloads: Neo4j. Follow chains of links
- Analytics: Derive information from the graph. E.g finding which user has the most friends under 30 years old: Tigergraph and JanusGraph
- Distribution is difficult: Compress a graph into a space-efficient data structure that fits in memory on a single node and then run the query against this data structure.
- SQL:2023 introduced property graph queries (SQL/PGQ)

Summary of approaches

- Non-SQL, non-relational systems are either a niche market or are fast becoming SQL/RM systems
- MapReduceSystems: died years ago
- Key-value Stores: matured as RM systems
- Document Databases: moved towards SQL
- Column-Family Systems: Google's BigTable
- Text Search Engines: ElasticSearch
- Array Databases: Valuable because SQL DBMSes are bad at this.
- Vector Databases: SQL databases will include this soon
- Graph Databases: Small market

Architectures: Columnar Systems

- Data warehouse
  - Historical data (periodically loaded)
  - Organizations retain everything (terrabytes to petabytes)
  - Queries typically only access a small subset of attributes
- Compressing columnar data is more effective
- A Volcano-style engine executes operators once per row. In contrast, a column-oriented engine has an inner loop that processes a whole column using vectorized instructions
- All vendors active in the data warehouse market have converted their offerings from a row store to a column store.
- New Actors: Amazon's Redshift and Google's BigQuery along with Snowflake, DuckDB

Cloud Databases (1)

- Initial cloud DBMS offerings repackaged on prem systems into managed VMs with direct-attached storage.
- All major cloud vendors offer NAS via object stores (e.g., Amazon S3) with some DBMS functionality (e.g., replication, filtering).
- Compute nodes are disconnected from the storage nodes, a system can provide per-query elasticity; the DBMS can add new compute nodes dynamically without having to reshuffle data
- “Pushing the query to the data” versus “pulling the data to the query”. Both possible.
- “Serverless computing”, and was introduced for cloud-native DBMSs by Snowflake
- Hosted multi-node environment in which multiple DBMS customers are grouped onto the same node(s) with a multi-tenant execution scheme.

Cloud databases (2)

- Has impacted DBMSs, causing them to be completely re-architected.
- Vendors can track usage trends for all their customers: they can monitor unexpected behavior, performance degradations, and usage patterns.
- Open-source DBMSs face the danger of becoming too popular and being monetized by the major cloud providers. The public spats between Amazon and ISVs like MongoDB and Elasticsearch

Data Lakes / Lakehouses

- With a data lake architecture, applications upload files to a distributed object store, bypassing the traditional route through the DBMS
- Lake house systems provide a unified infrastructure supporting SQL and non-SQL workloads
- Python-based notebooks that use Panda's DataFrame API to access data instead of SQL.
- Dask, Polars, Modin, and Bodo
- Formats: Twitter/Cloudera's Parquet and Meta's ORC
- Data lakes are the successor to "Big Data" movement from the early 2010s
- New challenges to query optimization
- Independent systems: Databricks, Dremio, PrestoDB, and Trino

NewSQL Systems

- NoSQL provided scalability with sharding without support for transactions
- NewSQL was created to provide scalability together with (distributed) transactions
- In-memory DBMSs, including H-Store (commercialized as VoltDB), SingleStore, Microsoft Hekaton, and HyPer.
- Disk-oriented: NuoDB and Clustrix
- Distributed, transactional SQL RDBMSs: Google Spanner, TiDB, CockroachDB, PlanetScale (based on the Vitess sharding middleware), and YugabyteDB
- Restrictions in NewSQL DBMSs: only supporting a subset of standard SQL or bad performance on multi-node transactions

Hardware Accelerators

- All custom hardware DBMS acceleration during the 1980s failed (replaced with software)
- Recently, commodity hardware (FPGAs, GPUs) to accelerate queries
- DeepgreenDB is the only remaining FPGA-enhanced DBMS available
- If data does not fit in GPU memory, then query execution is bottlenecked on loading data into the device.
- Creating custom hardware just for a DBMS is not cost-effective for most companies
- The only place that custom hardware accelerators will succeed is for the large cloud vendors.

Blockchain databases

- Decentralized log-structured databases (i.e., ledger) that maintain incremental checksums using some variation of Merkle trees.
- Use these checksums to verify that previous database updates have not been altered.
- At the present time, cryptocurrencies (Bitcoin) are the only use case for blockchains.
- Fluree, BigChainDB, and ResilientDB
- The only applications without real-world trust are dark web interactions (e.g., money laundering).
- All the major cryptocurrency exchanges run their businesses off traditional RDBMSs and not blockchain systems.

Key takeaways

- **Columnar Systems**: The change to columnar storage revolutionized OLAP DBMS architectures.
- **Cloud Databases**: The cloud has upended the coventional wisdom on how to build scalable DBMSs.
- **DataLakes/Lakehouses**: Cloud-based object storage using open-source formats will be the OLAP DBMS - archetype for the next ten years.
- **NewSQL Systems**: Promising, but has not had impact yet.
- **Hardware Accelerators**: Perhaps coming soon?
- **Blockchain Databases**: An inefficient technology looking for an application.

Comments from Stonebraker/Pavlo

- Never underestimate the value of good marketing for bad products.
- Beware of DBMSs from large non-DBMS vendors.
- Do not ignore the out-of-box experience.
- Developers need to query their database directly.
- The impact of AI/ML on DBMSs will be significant.
  - using natural languages (NLs) to query databases due to advancements in LLMs
  - English and other NLs are rife with ambiguities and impreciseness
  - The output of an LLM is not explainable to a human (enterprise decision making)
  - Research on using AI/ML to optimize the DBMSs: Indexes and query optimization