Section 2: Week 6: Database Quantum Supremacy

Nate Bachmeier

TIM-7020: Database and Business Intelligence

January 12th, 2020

North Central University

# Section 1: Identify and Discuss Technology

## How did we get here

If we had asked database administrators twenty years ago to predict the landscape of data management today, they would have likely pictured faster vertically scaled relational stores that enforce strong consistency and ACID compliance. When we reask the question a decade later, their answer would have changed to include NoSQL technologies, like Mongo, as the emergence of ICBM (IoT, Cloud, Big Data, and Mobile) introduced specific challenges. However, for many practitioners in 2010, these issues occurred in specific scenarios and irrelevant to many daily workloads. Today, those same experts would agree that modern data architectures require combinations of technology that rely on various design trade-offs. For instance, in-memory NoSQL stores, such as ElasticSearch, allow for real-time data exploration versus Big Data platforms, such as Apache Hive, focus on longitudinal reporting through batch processing. A critical advantage of these NoSQL stores comes with their ability to horizontally scale-out capacity. However, several studies show that the volume of ICBM data is doubling every year (Mansouri, Nadjaran, & Buyya, 2017), so this linear solution will eventually fail to keep pace with the exponential data growth. Instead, a “fundamental paradigm shift ‘from 10x to 10x’ needs to happen (Brandao & Kessler, 2019), like the one expected from quantum computing.”

## What is Quantum Supremacy

Quantum Supremacy occurs when a quantum processing unit (QPU) can perform a task more efficiently than a central processing unit (CPU). According to Brandao and Kessler (2019), the task does not need to be interesting or useful, and any proof of concept is sufficient. An extension of this idea resides in quantum database supremacy, which occurs when a quantum database can exceed the capabilities of a classical database. There are specific aspects of data management that are more likely to demonstrate this trait than others. These aspects tend to exist within matrix transformations (Brandao & Kessler, 2019), optimization problems (Harrison, 2020), and unsorted datasets (Grover, 1996). There are other areas, such as fetching an indexed row, that will be difficult to exceed classically designed systems because the algorithms are already extremely efficient and due to known insights about the data’s structure. It would, therefore, stand to reason that hybrid CPU/QPU databases exist before native QPU technologies, as this improves economics for businesses that purchase these systems. Such a hybrid model would need to build on the advancements of cloud-native NoSQL solutions, and rely on the QPU for specific query acceleration scenarios.

Some scientific enterprises are already adopting General Purpose Graph Processing Units (GPGPU) to accelerate their databases (Roozmeh, Torino, & Lavagno, 2017) (Marcin & Csillaghy, 2017) and will need to consider the trade-off of GPGPU/QPU technologies as there will be overlap across the problem space. For instance, both accelerators can perform complex matrix arithmetic. While GPGPU has a lower price point and well-entrenched frameworks, e.g., OpenCL, QPU allows for wildcards semantics through massively parallel entanglement processing. These differences might lead the organization to use a heterogeneous solution, such as GPGPU cards on general processing nodes, and reserve QPU circuits for expert systems.

## An extremely brief intro to Quantum Computing

A traditional computer operates on bits that hold a discrete value of either on or off. In contrast, a quantum bit or Qubit holds a superposition that expresses two distinct probabilities of being on and off simultaneously. These probabilities can become *entangled* with other qubits such that the resolved state of (A) cascades into (B) and (C). Until that resolution occurs, these entangled values can act as wildcards that pipeline many potential futures calculations in a deferred execution state. Consider the analogy that Alice has a 90% chance (amplitude) of paying Bill, and Bill might go to a movie with Charlie; provided Alice pays him—thus whether Charlie sees the movie with Bill is dependent (entangled) with Alice. Quantum circuits can use a series of Hadamard Gates to model this relationship in a pattern analogous to logic gates on traditional silicon (Gueddana, Chatta, & Boudriga, 2010). There are multiple possible outcomes, though once Charlie calls Bill to confirm is going, the circuit executes exactly-once, and that becomes the reality of their movie night.

Transaction management on traditional database systems uses locking or stacked storage constructs (Mansouri, Nadjaran, & Buyya, 2017). These approaches introduce blocking and excessive I/O operations, as the layers are merged during the commit phase. However, quantum databases could handle this scenario fundamentally different by directly entangling the state of the transaction with the existing data. When the transaction completes, the entangled values are associated with the commit state, similar to Charlie calling Bill. Using strategies along these lines could eliminate specific blocking and rollback challenges, enabling more concurrent transactions over fewer resources. As strongly consistent transactional stores become more competitive with the performance characteristics of eventual consistent stores, it could slow down the migration away from these legacy systems. Some argue that systems that can maintain strong consistency guarantees are more reliable and encounter lower maintenance costs (Liu, Arden, George, & Myers, 2017).

## Grover’s Algorithm

Quantum database theories often build on Grover’s fast quantum mechanical algorithm for database search, a generic solution that finds a specific value in an unordered set in exactly sqrt(N) steps. Consider the scenario where a million IoT sensors emit thousands of data points every few seconds continuously, resulting in 100 trillion records in the data lake. If the analysis needs to filter on an unindexed attribute, then the query would take an average time of 50 trillion (N/2) steps to complete. However, the application of a Grover search reduces the search space to only ten million steps, a large but manageable feat.

To understand Grover’s search, imagine trying to find a ball contained within one of ten identical boxes. Initially, there’s a ten percent chance of randomly selecting the correct choice. First, each box passes through an oracle (e.g., shaken horizontally), causing boxes (3, 6, and 9) to rattle, so the amplitude (probability) increases for these boxes. Next, the boxes iterate through the oracle (e.g., shaken at various angles) two more times. As each iteration completes, more evidence accumulates in the form of increased probability toward one of the boxes, allowing it to be selected correctly. De la Guardia(2016) lectures more concretely on the mechanics of this transform and visually explains the geometry involved.

Similar problems exist in distributed storage scenarios, where user records are load-balanced across various cluster nodes. Efficiently retrieving those records requires a distributed hash algorithm or a secondary index (Mansouri, Nadjaran, & Buyya, 2017). However, it can be economically prohibitive to index every value, which leads to the need for discovery protocols in certain scenarios. Using a quantum-accelerated system could provide that ad-hoc discovery in a reasonable length of time. Alternative classical data management technologies, like Apache Lucene, construct reverse term indexes to address these challenges, but its deployment relies on an entirely separate cluster.

# Section II: Advantages and Disadvantages

## Advantages of QDB

The primary strength of quantum databases comes from its ability to encode sequences of parallel potential future values. Consider an aggregation pipeline that contains a series of qubits such that multiple permutations are solvable in parallel. This optimization reduces the I/O requirements on the query engine and provides a more responsive experience to the customer. Another scenario might involve fuzzy matching and approximation, similar to neural networks, for making recommendations and proposing classifications. Because these operations are native to the quantum database, there certain extract-transform-load operations that are no longer needed. Traditional systems decouple the storage and machine learning platform, forcing engineers to manage analytic pipelines that become unwieldy in practice (Harrison, 2020). Another benefit comes from the performance improvements through Grover searches, bringing natural joins back into the NoSQL toolset, further reducing the learning curve for traditional SQL users.

## Disadvantages of Quantum Database

A critical hindrance to quantum databases is they only exist in mathematical proofs, and despite Quantum as a Service (QaaS) solutions like Azure Quantum and Amazon Bracket, it will be several years before these concepts become mainstream. According to Brandao (2019), many non-trivial algorithms need between 100 to 500 qubits, assuming error-free storage and external mitigation against random bit flipping, similar to traditional random access memory (RAM). Redundant encoding schemes, such as using three bits and a majority-wins voting protocol, are currently the de facto solution and cause the algorithms to need 0.5 to 1.0 million qubits. There are also limitations in simulation, as the entanglement of long qubit strings requires enormous amounts of storage, e.g., 50 qubits requires up to 16 petabytes.

Aside from the technical challenges of building reliable QPU circuits, the technology also needs to reach a state of supremacy. Data scientists can perform approximations and machine learning offline at relatively low costs, and GPGPU acceleration cards are readily available for other matrix computations. Public cloud providers, such as Amazon Web Services (AWS), support a notion of Elastic Interference Interfaces (EII) that can dynamically add and remove GPGPUs to hosted databases as customer demands fluctuation. Engineering teams also lack the training to take advantage of complex physics algorithms and providing that education would detract focus from the organization’s core competencies. Database administrators are already squeezing comparable performance from their system by optimizing indexes and data partitions around known business questions. For instance, analyzing a trillion sensor data points is less of an issue if the storage layer partitioned those records into time-ordered blocks—allowing for exclusion of entire ranges during the pre-query phase.

# Section III: Evolution

## Getting Ready for Quantum

Before many organizations can consider their quantum-accelerated implementation, they first need to migrate from traditional relational stores, such as SQL Server and Postgres, toward more cloud-native solutions. Network administrators have some freedom to the degree of their cloud adoption investment size. For instance, SQL Azure and Amazon Aurora present a NewSql interface that is familiar to engineering teams and offers capabilities such as automated fail-over and increased availability. Other organizations might choose to make a larger investment and select a NoSQL technology, such as Apache Casandra, Azure CosmosDb, or Amazon DynamoDB. These technologies make specific trade-offs in terms of throughput over functionality, like natural joins and built-in aggregations.

As the organization’s data management strategy matures, it discovers that different workloads require different technologies to gain specific optimizations. For example, the business uses a graph database, like Apache Tinkerpop, to store and query relationship information from a known entity (vertex). However, the graph data structures would not be efficient for holding time-series information, and those values would need a separate time-series store like Influx or OpenTSDB. Handling unknown starting vertices requires graph metadata to be placed in term stores like Apache Solr or Elastic Search. Answer extraction to common business questions into OLAP stores provides a consistent interface for visualization tooling, but many OLAP technologies, such as Amazon Redshift, target high bandwidth/high latency use cases, driving the need for Redis cache clusters to hold temporal data, such as results for the website’s homepage.

At this point, the organization takes a step back at the vast collection of built-for-purpose tooling they need to support and asks, what’s one more? Now they are ready for quantum. Jokes aside, the vision of quantum is that it reduces the need for these decoupled tooling scenarios. Instead, there is one universal technology stack that has ample power to deliver the needs of these various systems. For instance, Grover searches remove the need to maintain separate term stores, and offline processing is eliminated as online processing is sufficient. As performance challenges decrease, the decision to use ACID over BASE can focus on schema semantics and not solely transactional locks.

# Section IV: Business Adoption

## What challenges does this address

When academia first proposed neural networks, the business community did not understand the practical application of these systems. Since then, neural networks have touched every aspect of our online lives. Quantum databases will have a similar impact on society as they address optimization problems, machine learning, and fuzzy logic. As quantum technologies mature, they will grow as cloud-native extensions of the enterprise environment, and unlock new insights through massively parallel processing that powers rich business intelligence platforms. For instance, Brandano (2019) describes the challenges with modeling nitrogen processes as they contain over seventy states that High-Performance Computing (HPC) clusters can only approximate. As that estimations improve those manufacturing processes, gain huge efficiencies reducing global energy waste. All of these benefits happen on systems faster and in a more interactive manner. Extensions to the SQL query language will follow to take advantage of the entanglement constructs to naturally explore enormous data lakes of unstructured data through parallel processing.