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 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 were impractical to consider for 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), which means that a linear solution is attempting to solve an exponential problem. 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 classical databases. 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 can rely on known insights into the data 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 Graph Processing Units (GPU) to accelerate their databases (Roozmeh, Torino, & Lavagno, 2017) (Marcin & Csillaghy, 2017), which introduces another dimension to the complexity of the procurement.

## An extremely brief intro to Quantum Computing

A traditional computer operates on bits, which can hold a discrete value of either on or off. In contrast, a quantum bit or Qubit can hold a superposition that expresses two distinct probabilities of being on and off simultaneously. These probabilities can become *entangled* with other qubits where the resolved state of (A) cascades into (B) and (C). Until that resolution occurs, these entangled values can act as wildcards that connect many potential futures. Consider the analogy that Alice has a 90% chance (amplitude) of paying Bill, and Bill might go to a movie will Charlie if Alice pays him—thus whether Charlie sees the movie with Bill is dependent (entangled) with Alice. This quantum circuit can are modeled with Hadamard Gates, which are analogous to logic gates on traditional hardware (Gueddana, Chatta, & Boudriga, 2010).

Similar problems occur in various data management scenarios, such as transactions, where the committed result becomes entangled with processing results. That is fundamentally different than traditional systems that rely on more acts such as locking and stacked storage constructs (Mansouri, Nadjaran, & Buyya, 2017). Through the elimination of blocking and I/O intensive actions, the database could potentially scale to higher numbers of transactions. Mansouri et al. highlight that a key driver of workloads into eventual consistency solutions is to lessen the influence of transaction locking. However, fundamentally reducing this characteristic could push adoption back into strongly consistent technologies.

## Grover’s Algorithm

Quantum database theory often builds on Grover’s *fast quantum mechanical algorithm for database search* (1996). His algorithm can find a specific value from 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 could take a very long time to complete. However, the application of a Grover search would reduce 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. Then each box is shaken horizontally, causing boxes 3, 6, and 9 to rattle, so the amplitude (probability) increases for these boxes. Next, the boxes are shaken vertically and at various angles. As each iteration completes, more evidence accumulates that the probability that, e.g., box 7, is the correct instance.

Similar problems exist in distributed storage scenarios, where user records reside on arbitrary nodes. Today, retrieving those values requires a distributed hash algorithm or a secondary index (Mansouri, Nadjaran, & Buyya, 2017). However, it can be economically prohibitive to index every value, which further leads to the need for discovery after creation. Specific technologies, like Apache Lucene, expose term indexing to improve the costs and performance of these scenarios today, though the deployment relies on an entirely separate second data store. In a quantum-accelerated system, a redundant copy and the synchronization overhead does not need to exist.

# Section II: Advantages and Disadvantages

## Advantages of QDB

The primary strength of quantum databases comes in their ability to encode sequences of potential future values. Consider an aggregation pipeline that sequences a series of qubits such that various permutations are solvable in parallel. This optimization reduces the I/O requirements on the query engine and provides a richer experience for the customer. Another scenario might involve fuzzy matching and approximation, similar to neural networks today, to make recommendations or propose classifications. In contrast, typical data machine learning workflows are complex due to the decoupling of storage and machine learning platform. This separation creates the need for extract-transform-load systems that can become unwieldy (Harrison, 2020). Another benefit comes from the performance improvements through Grover searches, which could allow natural joins could become part of 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 it will be several years before these concepts become mainstream, despite Quantum as a Service (QaaS) solutions like Azure Quantum and Amazon Bracket. According to Brandao (2019), many non-trivial algorithms need between 100 to 500 qubits, assuming error-free storage and mitigate random bit flipping similar to traditional random access memory (RAM). In reality, redundant encoding schemes, such as using three bits and a majority-wins voting protocol, 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 a reliable QPU, there is also the need for this technology to reach supremacy. Data scientists can perform approximations and machine learning offline at relatively low costs, and GPU acceleration cards are readily available today. Public cloud providers, such as Amazon Web Services (AWS), support a notion of Elastic Interference Interfaces (EII) that can dynamically add and remove GPUs to hosted databases to meet demand fluctuations. Engineering teams also lack the training to take advantage of complex physics algorithms and providing that education would remove focus from the organization’s core competencies. Database administrators are also squeezing comparable performance from their system by optimizing indexes and data partitions around business questions. For instance, analyzing a trillion sensor data points is less of an issue if the storage grouped those values into time-ordered blocks—allowing for exclusion of entire ranges during the pre-query phase.

# Section III: Evolution

## Transitioning from Relational

Many organizations there are