Towards Quantum Data Structures for Enhanced Database Performance

Tim Littau
Technische Universiteit Delft
Delft, Netherlands
info@tmlittau.com

Ziyu Li
Technische Universiteit Delft
Delft, Netherlands
Z.Li-14@tudelft.nl

Rihan Hai
Technische Universiteit Delft
Delft, Netherlands
R.Hai@tudelft.nl

ABSTRACT

Quantum computing holds the promise of expanding the capabilities of data management by introducing fundamentally new ways to interact with data, going beyond simple enhancements in processing speed and efficiency. This paper proposes the development of innovative quantum data structures designed to optimise database search and manipulation operations by leveraging the unique capabilities of quantum mechanics, such as superposition and entanglement. We introduce the Quantum Partitioned Database (QPD) utilising a modified Grover's Algorithm for data retrieval of multiple elements in a database, and demonstrate the practical implementation of circuit-based quantum data structures. Building on foundational concepts like Quantum Random Access Memory (QRAM) and Quantum Random Access Gates (QRAG), our approach bridges the gap between theoretical advancements and real-world applications. This research aims to catalyse the adoption of quantum technologies in data management, providing a robust framework for future innovations, performance enhancements, and a new paradigm in database search and manipulation.

VLDB Workshop Reference Format:
Tim Littau, Ziyu Li, and Rihan Hai. Towards Quantum Data Structures for Enhanced Database Performance. VLDB 2024 Workshop: The Second International Workshop on Quantum Data Science and Management (QDSM'24).

1 INTRODUCTION

Quantum computing, representing a pivotal shift from traditional computing paradigms, holds the promise of transforming various scientific and technological fields through both its computational power and its novel approach to data interaction. Just as classical computing began as a theoretical concept in the 19th century [9] and only became practical around the 1960s [2], quantum computing started as a theoretical idea around the 1980s [17] and has only recently become a reality with advancements in quantum computers [6]. Despite significant advancements in quantum hardware [12, 16], the application of quantum computing in real-world scenarios remains challenging, necessitating a crucial effort to bridge the gap between theory and practice.

Over the past 25 years, foundational work such as Grover's Algorithm [19] has demonstrated the potential of quantum computing to exponentially speed up search problems, which are central to many data management applications. While existing quantum computers are still in their early stages, they provide an opportunity to explore the first practical applications of quantum computing in data management systems. Recent research has explored optimisation of quantum query processes [34, 35, 37], join ordering [32, 33], and data manipulation in superposed states [21, 40]. However, the development of more general quantum data structures for data management on a circuit level has been largely overlooked, leaving a significant gap in the practical implementation of quantum algorithms in data management.

Classical computing has heavily relied on data structures like B-trees [13] and hash maps [14] to enhance data retrieval and manipulation efficiency. These structures leverage the properties of the represented data to improve performance, though often at the cost of increased memory requirements. In contrast, quantum computing presents unique properties such as superposition and entanglement, which, if harnessed effectively, can lead to the development of quantum data structures that process information in fundamentally novel ways. Integrating quantum computing into data management pipelines not only promises speed-ups without the need for traditional indexing but also holds the potential to reduce memory demands.

This paper proposes the development of innovative quantum data structures designed to optimise database search and manipulation operations by leveraging the unique capabilities of quantum mechanics. We propose the Quantum Partitioned Database (QPD), utilising Grover's Algorithm for retrieving multiple elements in a database. This approach demonstrates the practical application of circuit-based quantum data structures, bridging the gap between theoretical advancements and real-world applications. While our research draws inspiration from foundational concepts such as Quantum Random Access Memory (QRAM) [18] and Quantum Random Access Gates (QRAG) [1], it does not directly implement these structures. Instead, we demonstrate a basic form of circuit-based quantum memory, laying the groundwork for the development of more sophisticated quantum data structures used in data management.

We foresee not only performance enhancements in database operations through the practical implementation of quantum data structures, but also a fundamental shift on how we are able to interact with data. By addressing existing bottlenecks in classical database systems and proposing concrete solutions for data translation and hardware limitations, our work sets the stage for leveraging quantum computing's full potential in data management scenarios.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

Quantum data structures like QPD will act as a link between classical and quantum data and present the first step towards the development of a quantum database. This ultimate goal does not only promise enhanced performance, but a whole new paradigm for data management, offering novel ways to interact with data such as dynamic resource allocation using transactions in superposed states [31].

Our contributions go as follows:

- **Opportunities and Challenges(Sec. 2)**: We discuss the potential benefits and challenges faced in applying quantum computing in data management.
- **Proposed Approach (Sec. 3)**: Design of a proposed quantum data structure to practically apply a modified Grover's Algorithm for retrieving multiple elements.
- **Our Vision (Sec. 4)**: Presents our overall vision and a possible outlook for future research.
- **Related Concepts (Sec. 5)**: Presents foundational work on QRAM and QRAG, providing a general framework for circuit-based quantum data structures.

2 CHALLENGES

In this section we address current challenges faced in data management and the paradigm shift offered by quantum computing promising to tackle them. Furthermore, we present how practical applications in quantum computing are currently implemented and finally the consequent idea of designing circuit-based quantum data structures for the practical application of quantum algorithms to combat classical limitations.

Classical vs Quantum Data Management. While classical data management heavily relies on index data structures, such as B-trees and hash maps, to enhance data retrieval and manipulation, they typically come with downsides, such as increased memory requirements and complex management overheads when the index needs to be updated. As databases grow, maintaining and optimising these indices becomes increasingly challenging and resource-intensive.

Quantum computing introduces a fundamentally different paradigm that can potentially eliminate the need for traditional indices. Quantum mechanical principles like superposition and entanglement allow for fundamentally new ways to interact with data. Furthermore, quantum algorithms already show a quadratic speedup for unstructured search problems [19]. This efficiency can reduce the complexity of tasks that are infeasible with classical algorithms, enabling faster and more scalable data processing [39].

Practical Implementation of Quantum Algorithms. Currently for practical applications, quantum algorithms are typically implemented using quantum circuits. Software development kits like Qiskit [24] allow for the design and execution of the circuits on current quantum hardware. The implementation of quantum algorithms in the form of quantum circuits is a crucial step in bridging the gap between theoretical advancements and real-world applications.

In classical computing, algorithms are designed to interact with and manipulate data structures efficiently. The choice and design of a data structure can significantly influence the performance and complexity of an algorithm. Data structures provide the necessary organisation and storage mechanisms for data, while algorithms define the procedures for data processing tasks such as searching, sorting, and updating.

In the context of quantum computing, this interdependence suggests that to implement quantum algorithms effectively, we should design corresponding quantum data structures. These structures should leverage quantum mechanical properties to enhance data processing capabilities. One example of such quantum algorithm would be the modified Grover's Algorithm for multiple item retrieval, which requires a quantum data structure that supports parallel searches on multiple datasets [20].

Practical Challenges

Designing effective and practical circuit-based quantum data structures comes with its own challenges. In the following, we address some of these challenges and offer an outlook on possible solutions.

Data Translation Overhead. Efficiently loading classical data into quantum systems presents a significant bottleneck in practical applications of quantum algorithms. To overcome this bottleneck, optimising the encoding of classical into quantum data is required when designing circuit-based quantum data structures [3]. Otherwise, the speed-up through quantum algorithms could, in the worst case, be nullified.

Hardware Limitations and Scalability. Current quantum hardware, while advancing rapidly, still faces limitations in terms of qubit count [12], coherence time [29], and error rates [25]. Developing robust and efficient error correction techniques and improving qubit coherence times are critical for overcoming these limitations. To mitigate these hardware limitations in the short term, hybrid quantum-classical approaches, where classical systems handle task scheduling and tasks requiring high reliability, offer a viable solution while quantum processors only tackle the problems benefiting from quantum speed-up. For these hybrid quantum-classical architectures, middleware layers and APIs that facilitate communication between classical and quantum components are crucial to guarantee a seamless integration [7].

3 QPD: A QUANTUM DATA STRUCTURE FOR SEARCH

In this section, we present the circuit design of what we refer to as the Quantum Partitioned Database (QPD) using Grover's Algorithm to retrieve multiple elements from a database [20]. This approach leverages the principles of quantum parallelism and is inspired by the capabilities of QRAG.

Given a large database D with N elements, our goal is to retrieve k marked elements efficiently. To achieve this as described in [20], the database is split into d subsets, such that the condition $k \leq \sqrt{d}$ is satisfied. Each subset's indices are stored in separate quantum registers, allowing us to perform parallel searches on each register using Grover's Algorithm.

Differences between QPD and Grover and Radhakrishnan's Work [20]. While the theoretical framework proposed by [20] provides foundational principles for parallel quantum search algorithms, our QPD extends this work by focusing on practical implementation.

QPD is designed with current quantum hardware limitations in mind, including considerations for qubit count, coherence times, and error rates, which are not addressed in [20] purely theoretical algorithm. Additionally, we integrate a middleware layer to facilitate seamless interaction between classical and quantum systems, ensuring efficient resource usage and practical applicability. This hybrid approach ensures efficient resource usage and practical applicability, bridging the gap between theoretical quantum algorithms and practical data management systems.

The additional qubits required to store the indices of different subsets in parallel are a trade-off for the speed-up we can achieve through parallelisation.

In Figure 1 we show the architecture of the proposed implementation of QPD. Assuming an incoming query to a classical database we require some sort of interface to the quantum system such that the splitted database can be encoded as shown in Figure 2. In contrast to a classical table of data pairs (i_j, x_{ij}), in the quantum version the qubits in the index register are effectively entangled with the qubits in the data register, such that they create the mapping described in Equation 1. Equally the QPD operation can be expressed as shown in Equation 2.

$$|i_j\rangle |0\rangle \mapsto |i_j\rangle |x_{ij}\rangle \tag{1}$$

$$U_{QPD} |i_j\rangle |0\rangle = |i_j\rangle |x_{ij}\rangle \tag{2}$$

To decrease qubit requirement there is no need to split the whole database, instead the queried properties or columns are extracted in the classical-quantum interface, followed by the actual split. The extracted columns are then provided to the quantum system to be encoded.
Once the indices of the searched items are retrieved by the quantum algorithm they are returned to the interface to properly translate them back to the unsplitted database.

In the following, the steps to process the translated classical data received by the quantum interface are described in more detail.

Steps of QPD Implementation.

(1) Database Initialization
   - Randomly splitting the database D of size N into d subsets, $D = \{D_1, D_2, ..., D_d\}$, each containing $\frac{N}{d}$ elements.
   - Storing the indices of each subset D_i in a separate quantum register $|i_j\rangle$.

(2) QPD Configuration
   - Each subset D_i is configured with quantum registers to store data pairs (i_j, x_{ij}), where i_j is the j-th index of subset register i and x_{ij} is the corresponding data element.
   - The QPD maps quantum state $|i_j\rangle |0\rangle$ to $|i\rangle |x_{ij}\rangle$, enabling data retrieval in superposition.
   - The circuit implementation is shown in Figure 2.

(3) Parallel Query Processing
   - Initialize the index registers in superposition and entangle these with the data register across all subsets as described in step 2.
   - Use Grover's Algorithm [19] to perform parallel searches. This involves:
     – Applying the oracle function to mark the elements that match the search criteria $U_\omega |x_{ij}\rangle$.
     – Applying the diffusion operator to amplify the probability amplitude of the indices of marked states $U_D |i_j\rangle$.

(4) Measurement
   - Measure the index registers to retrieve the indices of the marked elements from the corresponding subsets.

This implementation is inspired by unary encoding in conjunction with parallel QRAM. These are further explained in the Section 5. Even though we require more qubits to store each subset of indices into separate quantum registers, it enables the whole system to be easily adapted for parallel implementation like in a distributed quantum system [8].

The QPD is implemented using the Qiskit SDK [24]. In Figure 3 the circuit design of the algorithm as it is implemented in Qiskit is shown. The Oracle operation simply flips the probability amplitude of all searched states, effectively performing a rotation about $|0\rangle$. The Diffusion operation flips the indices of their corresponding marked states back by rotating them 90° about the original index state $|s\rangle$. This results in the operator for marking the states U_ω and the operator for amplifying their amplitudes U_D:

$$U_\omega = I - 2 \sum_l |\omega_l\rangle \langle\omega_l| \tag{3}$$

$$U_D = 2 |s\rangle \langle s| - I \tag{4}$$

Where I is the Identity Matrix and $|\omega_l\rangle$ is one of the searched states.

A benchmark comparison between QPD and classical search methods would involve evaluating the execution time for retrieving multiple elements from a database. For classical searches, we consider both unindexed linear searches with a time complexity of $O(N)$ and indexed searches with a complexity of $O(\log(N))$. In contrast, the QPD utilises Grover's Algorithm, providing a quadratic speedup in comparison to unindexed search with a time complexity of $O(\sqrt{(N)})$.

Addressing the Challenges

Following the challenges presented in Section 2, we discuss how the proposed implementation of QPD and quantum data structures designed and implemented in a similar way overcome these challenges in the short and long term.

Data Translation Overhead. To tackle the overhead of encoding classical data into quantum data, QPD offers a solution by leveraging its partitioned structure. Instead of encoding the entire database, QPD partitions it and encodes $\frac{N}{d}$ entries in parallel, effectively reducing the circuit depth through parallelisation. However, parallelisation is just one approach to optimise the encoding step. There are already

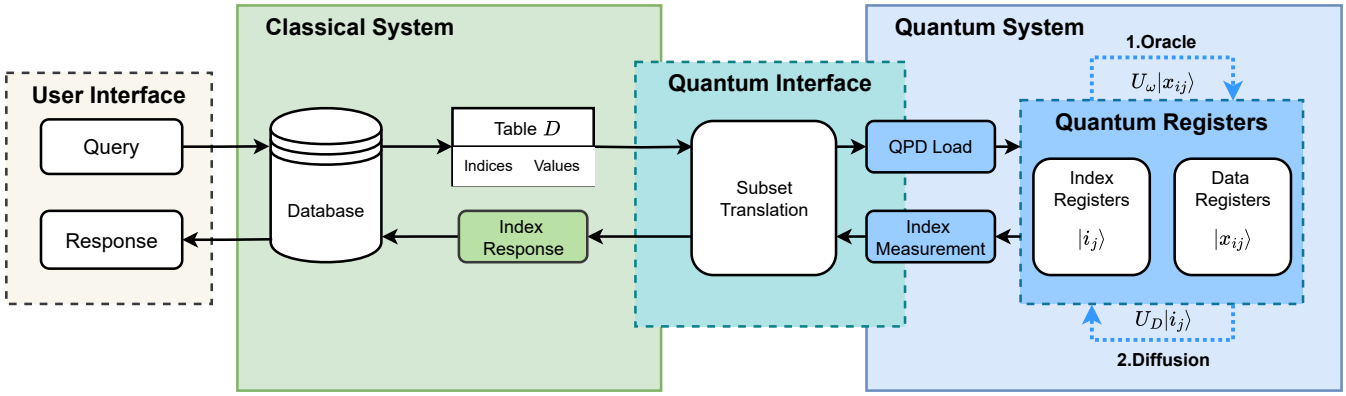

Figure 1: Architecture of the proposed implementation using QPD for Database Searches. The Quantum Interface and Quantum System can be adjusted to implement other operations, algorithms and data structures

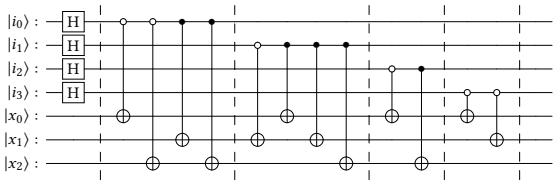

Figure 2: QPD circuit design of a split into four subsets, $\{(5,6), (2,7), (1,4), (3,0)\}$. The first column of Hadamard-gates superposes all index registers into $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$

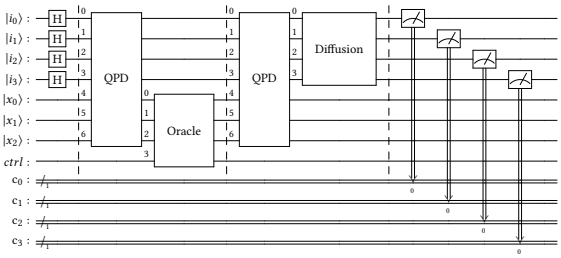

Figure 3: Algorithm circuit design as implemented in Qiskit. The columns with custom gates *QPD, Oracle* and *Diffusion* represent one iteration and are repeated $\sqrt{\frac{N}{d}}$ times. In this example $N = 8$ and $d = 4$, which results in just one iteration.

QRAM implementations that reduce linear gate-cost to encode data to logarithmic cost by [23]. Furthermore, if the encoded data is sparse, an even further reduction is possible [15]. Depending on the requirements, either solution can be chosen.

Hardware Limitations and Scalability. Addressing the limitations of current quantum hardware, our proposed architecture adopts a hybrid quantum-classical approach that optimises resource usage. The partitioning of the database reduces circuit depth which is crucial with respect to coherence times. By initially filtering the database, the qubit count is kept minimal, only providing the relevant data to the quantum system. Automated optimisation of quantum circuits using machine learning methods to reduce error seems promising [22]. Furthermore, limited coherence times can also be handled using temporal-planners to reduce the duration of quantum circuits [38]. Applying methods like this circumvents the current limitations of quantum hardware, while coherence times, error rates and qubit counts improve over the coming years.

Scalability Analysis of QPD. The scalabiliy of QPD hinges on balancing the theoretical speedup with practical limitations of current quantum hardware. QPD offers a quadratic speedup with its $O(\sqrt{\frac{N}{d}})$ scaling compared to classical $O(N)$ algorithms, but this advantage is tempered by increased qubit requirements and circuit depth. Specifically, $d \log(\frac{N}{d})$ qubits are needed for partitioned indices, possibly posing a challenge for near-term quantum hardware.

Additionally, the circuit depth, mainly determined by $O(\sqrt{\frac{N}{d}})$ Grover iterations [20], can strain systems with limited coherence times. While partitioning reduces iterations, it increases qubit count, necessitating a careful balance based on hardware capabilities. Scalability is further constrained by data loading times, error accumulation in deeper circuits, and computationally intensive classical pre-processing. Future advancements in quantum error correction and memory architectures will be essential to fully leverage QPD's potential for large-scale databases, but for now, its practical applicability is limited by current hardware constraints.

4 VISION FOR QUANTUM DATA MANAGEMENT

The vision we propose for quantum data management is to first develop robust, versatile quantum data structures that are connected to classical systems, transforming how data is stored, accessed and manipulated. Followed by the development of quantum data structures that can be applied in fully quantum-based systems, this, in the near future, could involve the application of quantum data structures in distributed systems using the quantum internet, enabling quantum computations with increased performance by using shared resources. Ultimately, these data structures will build the

foundation of a quantum database offering entirely new ways to store and handle data.

Hence, in the following are some of the key aspects of our vision.

Quantum Data Structures. We envision creating quantum data structures that are not only analogous to classical ones but leverage the unique advantages of quantum computing. These structures should be capable of handling a wide range of data management tasks, from simple retrievals to complex queries and optimisations. While there have already been efforts in developing quantum data structures for specific applications as shown in section 5. Future data structures should be versatile, scalable, and efficient, enabling quantum computing to be easier applied in data management applications.

Hybrid Quantum-Classical Systems. The near-term future of quantum computing lies in hybrid systems that combine the strengths of classical and quantum processors. The proposed data structures aim to bridge the translational gap between classical and quantum data, representing one step further into transforming data as we know it. To achieve practical and scalable quantum data management, integrating quantum data structures with classical systems is crucial. Our proposed vision includes middleware layers that bridge classical and quantum components, facilitating efficient data transfer and processing between these systems.

Middleware Layers and APIs: The middleware layer, or Quantum Interface, comprises several key APIs and protocols to ensure seamless communication and data transfer between classical and quantum components. These include:

(1) **Data Translation Layer:** This layer converts classical data structures and queries into quantum states suitable for processing by quantum algorithms, ensuring efficient data encoding and maintaining query integrity. This process involves mapping database indices to qubit states and preparing the quantum register accordingly.
(2) **Task Scheduling API:** This API manages the distribution of computational tasks between classical and quantum processors, scheduling queries based on system load and hardware status to ensure optimal resource utilization and minimal latency.
(3) **Result Aggregation Layer:** This layer collects and processes the results from quantum computations, translating them back into a format understandable by classical systems. This layer handles the post-processing of quantum data, including subsequent error correction and result verification. Besides subsequent error correction, a variety of integrations for quantum error correction (QEC) already exists, such as Boulder Opal by Q-CTRL [11] or Deltaflow by Riverlane [5, 36].

Similar to the architecture described in Section 3 a more general description for hybrid systems looks like this:

(1) **Query Reception:** The classical system receives the query and performs initial preprocessing to determine relevant data subsets. During this stage, the system analyzes the query to identify which parts of the database are relevant. It filters and preprocesses the data, ensuring that only the necessary information is prepared for quantum processing. This step helps to reduce the amount of data that needs to be encoded into quantum states, optimizing the overall process.
(2) **Data Encoding:** The Data Translation Layer encodes the identified data subsets into quantum states. This involves transforming classical data, such as database indices or values, into qubits. The encoding process uses efficient schemes to map the data accurately onto the quantum register, ensuring that the quantum algorithm can process it effectively. The goal is to represent the data in a way that leverages quantum parallelism and interference.
(3) **Quantum Execution:** The encoded data is sent to the Quantum System, where quantum algorithms (e.g., Grover's Algorithm) are executed. The quantum processor performs the necessary computations on the encoded data, utilising the inherent advantages of quantum mechanics, such as superposition and entanglement, to search or analyze the data more efficiently than classical algorithms could. This step is the core of the quantum enhancement, providing potential speedups and performance improvements.
(4) **Result Decoding:** The results from the quantum computation are decoded and processed by the Result Aggregation Layer. After the quantum computation is complete, the results are measured and translated back into classical information. The Result Aggregation Layer applies error correction protocols to mitigate any quantum errors that may have occurred during computation. It then verifies the accuracy of the results, ensuring they are valid before passing them back to the classical system.
(5) **Final Output:** The processed results are returned to the classical system, which formats and presents them to the user. The classical system takes the validated results from the Result Aggregation Layer and formats them into a user-friendly output, such as a report or visualization. This final step ensures that the end user receives accurate and actionable information derived from the quantum-enhanced data processing.

Scalable Quantum Data Management Solutions. As quantum hardware continues to evolve, scalable solutions will be essential for practical deployment. We aim to develop quantum data structures that can grow with advancements in quantum technology, ensuring they remain relevant and efficient as qubit counts and coherence times improve.

Real-World Applications. Our goal is to bridge the gap between theoretical research and real-world applications. By focusing on practical implementation, we aim to demonstrate the tangible benefits of quantum data structures in the field of data management.

Quantum Databases. Quantum data structures promise not only computational speed-ups but also a fundamental transformation in how data is interacted with. Current efforts in practically implementing quantum approaches are just the beginning and will pave the way for more sophisticated applications, such as quantum databases.

While classical computing is adept at managing our everyday tasks, the inherent uncertainty in human nature presents a unique opportunity for quantum computing to excel in handling complex

and probabilistic data more efficiently. Quantum data structures lay the groundwork for a fully functional quantum database, leveraging the principles of quantum mechanics. Existing efforts in researching quantum databases mainly focus on the integration of classical data into quantum systems [30]. However, foundational research for fully quantum databases is lacking, despite their potential advantages beyond classical paradigms. Rather than entirely replacing classical computing, quantum computing aims to complement and enhance it, particularly in areas where its unique properties offer significant advantages. Efficiently managing probabilistic data will be a crucial application of such a quantum database, demonstrating its potential to transform data management practices.

Future research should focus on refining these quantum data structures, and developing efficient quantum-classical interfaces, while quantum hardware improves. Real-world applications in areas such as dynamic resource allocation, probabilistic data management, and database optimisation will serve as testbeds for these advancements.

5 RELATED WORK

In this section, we review the existing work on circuit-based quantum data structures, namely the concept of QRAM and the more generalised QRAG.

While the QPD concept leverages principles from Grover's Algorithm, it distinguishes itself through practical implementation which necessitates robust data structures. This practical aspect is notably absent in Grover and Radhakrishnan's method [20]. Here, the ideas of QRAM and QRAG are particularly relevant. QRAM provides a foundational framework for storing and retrieving data efficiently in quantum states, enabling the simultaneous querying of multiple data entries, which is crucial for the parallelism inherent in QPD. QRAG extends this by incorporating in-memory quantum operations, facilitating data manipulation during retrieval, thus optimizing the efficiency of quantum algorithms. Although QPD is not a direct implementation of QRAM or QRAG, these concepts are fundamental inspirations for developing practical quantum data structures. They demonstrate how classical data can be effectively represented and encoded within quantum circuits, laying the groundwork for QPD's architecture and its ability to perform data retrieval operations.

5.1 Quantum Random Access Memory

QRAM is an analogous concept to classical RAM in the realm of quantum data structures, providing a bridge between classical and quantum information for quantum computing. QRAM is designed to enable efficient access of data stored in quantum states.

At its core, QRAM allows for the storage and retrieval of data where the pair of address and data (i, x_i) can exist in superposition. This means that QRAM can query and process multiple data entries simultaneously, which is the foundation for most quantum algorithms [18].

The relevance of QRAM to our vision lies in its role as a basic idea to develop more advanced circuit-based quantum data structures. By understanding QRAM, we can built upon its fundamental principles in acting as a translational layer between classical and quantum data. This translational layer is required to power the speed-up promised by quantum algorithms.

Circuit-based QRAM currently encompasses four different types of implementations as described in [23] and shown in Table 1.

Unary Encoding: The optimised version of this approach uses an efficient recursive structure for QRAM control [4]. In the base case of a one-element controlled look-up is just a CNOT gate, which simplifies the implementation. It is characterised by its low qubit count and simplicity, making it an attractive option to start with. However, it requires auxiliary qubits for address comparison, which adds complexity. While it is efficient in terms of qubit count and gate usage, the need for auxiliary qubits for each address comparison can become cumbersome as the memory size increases. The recursive structure ensures logarithmic depth, but the overall complexity increases linearly with the size of the memory.

Bucket-Brigade: This approach employs a binary tree structure for memory access, using controlled-SWAP gates along with CNOT and Toffoli gates [18]. It is known for its favorable error scaling with logarithmic depth and efficient memory access routing. However, it also involves significant complexity in uncomputation and managing ancillary qubits. Additionally, it requires a large number of ancilla qubits for the routing tree. Despite these challenges, the scalability of the Bucket-Brigade method is high. It scales well with logarithmic depth, making it efficient for large memory sizes. The logarithmic error scaling further enhances its suitability for large-scale QRAM implementations, although practical challenges arise from its complexity.

Select-Swap: This method reduces the LOAD-count using a paging analogy similar to classical memory paging. This involves first accessing a larger segment of memory and then performing a secondary, more detailed access within this segment. Specifically, this method uses a combination of CNOT, SWAP, and Toffoli gates. By reducing the number of LOAD-gates required, this approach makes the overall quantum circuit more efficient in terms of gate operations. However, it increases the number of ancillary qubits needed and adds complexity in managing these qubits and the paging process. Scalability is moderate to high, offering a balance between efficiency and complexity for medium to large implementations, though the ancillary qubit requirements and paging complexity grow significantly with larger memory sizes.

Parallel: Parallel QRAM implements parallel circuits to allow simultaneous memory access, using various quantum gates in parallel. This parallel access significantly reduces overall access time, potentially enhancing performance. However, it introduces complexity in managing parallel circuits and requires sophisticated control mechanisms to handle these operations. It is designed to handle large-scale implementations by efficiently managing very large memory sizes through parallel processing. Despite the challenges in managing parallel circuits and the need for sophisticated control mechanisms, its parallel nature makes it highly scalable.

Quantum Data Centers with QRAM. Recent research has introduced the concept of Quantum Data Centers (QDCs), which integrate QRAM with quantum networks [26, 27]. QDCs provide a versatile platform for applications ranging from multiparty private quantum communication to distributed sensing through data compression.

Implementation	Scalability	Advantages	Challenges
Unary Encoding [4]	$O(N)$	low Qbit count, simple	address comparison
Bucket-Brigade [18]	$O(\log(N))$	favorable error scaling log depth	complex uncomputation ancilla management
Select-Swap [28]	reduced LOAD-count	efficient	ancilla Qbit requirements
Parallel [8]	scales well in terms of access time	reduced access time	circuit management

Table 1: Comparison of circuit-based QRAM Implementations.

5.2 Quantum Random Access Gate

QRAGs extend the concept of QRAM by integrating quantum computational capabilities directly into the memory access process. While QRAM facilitates the storage and retrieval of classical or quantum data using addresses in superposition, QRAGs elevate this functionality by enabling in-memory quantum operations. This integration allows for quantum data manipulation and computation to occur during the memory access phase, optimizing the efficiency and scope of quantum algorithms.

QRAGs are designed to address the need for more sophisticated data structures that not only fetch quantum data efficiently but also perform computational tasks as part of the retrieval process. This is particularly beneficial in quantum algorithms that require tight integration between data access and processing, such as quantum search algorithms and quantum machine learning applications. By embedding quantum gate operations within the memory access architecture, QRAGs reduce the overhead associated with separate computation and data retrieval steps, thus enhancing the overall performance of the quantum system [10].

Despite the increased complexity, QRAGs offer significant advantages in terms of computational capabilities and operational efficiency, making them a foundational concept in the development of quantum data structures.

Definition and Operation. QRAG typically operates on three key inputs:

- **Index Register ($|i\rangle$):** Identifies the memory location or address.
- **Data Register ($|x\rangle$):** Holds the actual data or a portion of it.
- **Swap qubit ($|b\rangle$):** Acts as an intermediate qubit to manipulate an accessed data qubit $|x_i\rangle$.

The gate operation then executes the mapping:

$$|i, b, x\rangle \rightarrow |i, x_i, x_1 \cdots x_{i-1}, b, x_{i+1} \cdots x_m\rangle$$

In this mapping, the value at the indexed position (x_i) is accessed and potentially modified based on the swap qubit $|b\rangle$, while the rest of the data register remains unaffected except at position i.

Formally, by this definition the presented QRAM and also our proposed QPD are QRAG implementations for a specific $|b\rangle$.

Beyond Classical Data Structures. QRAGs present a more dynamic alternative to simple QRAM, with the possibility to integrate combined properties of classical data structures into quantum computing. This has already been shown for a hybrid implementation of a QRAG that acts as a hash table and a skip list in combination [1].

Probabilistic data structures like skip lists can benefit significantly from the inherent properties of quantum computing. Combining a skip list with a hash table in a quantum implementation does not necessarily offer advantages over a classical hash table by default. However, with larger datasets, classical hash tables tend to become less efficient due to increased collision handling and rehashing overheads. These inefficiencies can be significantly mitigated in a quantum implementation, where quantum parallelism and superposition can enhance data retrieval and management.

This is just one example of how QRAGs can offer the conceptual foundation for data structures that are tightly integrated with quantum algorithms.

6 CONCLUSION

In this paper, we have explored the promising frontier of quantum data structures for enhanced database performance, leveraging the unique properties of quantum mechanics, such as superposition and entanglement. By developing novel quantum data structures like the Quantum Partitioned Database (QPD), we demonstrate the practical application of quantum computing in database search.

Our work is inspired by foundational concepts such as QRAM and QRAG, providing a first starting point for practical implementation of quantum algorithms in data management. The proposed hybrid quantum-classical systems serve as an immediate step towards realising quantum computing's benefits, bridging the gap between current technological capabilities and future advancements.

In summary, bridging the gap between theoretical research and practical implementation will be essential for leveraging quantum computing technologies effectively. While quantum computing may not replace classical computing, it promises to revolutionise data interaction and expand our computational capabilities through innovative, not just performance-driven, approaches.

ACKNOWLEDGMENTS

We thank Georgios Christodoulou for contributing by providing valuable feedback with his extensive knowledge in the field of Data Indexing.

REFERENCES

[1] Andris Ambainis. 2014. Quantum walk algorithm for element distinctness. arXiv:quant-ph/0311001 [quant-ph]

[2] G. M. Amdahl, G. A. Blaauw, and F. P. Brooks. 1964. Architecture of the IBM System/360. *IBM Journal of Research and Development* 8, 2 (1964), 87–101. https://doi.org/10.1147/rd.82.0087

[3] Sahel Ashhab. 2022. Quantum state preparation protocol for encoding classical data into the amplitudes of a quantum information processing register's wave function. *Physical Review Research* 4, 1 (Feb. 2022). https://doi.org/10.1103/physrevresearch.4.013091

[4] Ryan Babbush, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. 2018. Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity. *Physical Review X* 8, 4 (Oct. 2018). https://doi.org/10.1103/physrevx.8.041015

[5] Ben Barber, Kenton M. Barnes, Tomasz Bialas, Okan Buğdaycı, Earl T. Campbell, Neil I. Gillespie, Kauser Johar, Ram Rajan, Adam W. Richardson, Luka Skoric, Canberk Topal, Mark L. Turner, and Abbas B. Ziad. 2023. A real-time, scalable, fast and highly resource efficient decoder for a quantum computer. arXiv:2309.05558 [quant-ph] https://arxiv.org/abs/2309.05558

[6] R. Barends, L. Lamata, J. Kelly, L. García-Álvarez, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Yu Chen, Z. Chen, B. Chiaro, A. Dunsworth, I.-C. Hoi, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, E. Solano, and John M. Martinis. 2015. Digital quantum simulation of fermionic models with a superconducting circuit. *Nature Communications* 6, 1 (08 Jul 2015), 7654. https://doi.org/10.1038/ncomms8654

[7] Kenton Barnes, Anton Buyskikh, Nicholas Chen, Gabriel Gallardo, Marco Ghibaudi, Matthew Ruszala, Daniel Underwood, Abhishek Agarwal, Deep Lall, Ivan Rungger, and Nikolaos Schoinas. 2023. Optimising the quantum/classical interface for efficiency and portability with a multi-level hardware abstraction layer for quantum computers. *EPJ Quantum Technology* 10 (09 2023). https://doi.org/10.1140/epjqt/s40507-023-00192-z

[8] Robert Beals, Stephen Brierley, Oliver Gray, Aram W. Harrow, Samuel Kutin, Noah Linden, Dan Shepherd, and Mark Stather. 2013. Efficient distributed quantum computing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 469, 2153 (May 2013), 20120686. https://doi.org/10.1098/rspa.2012.0686

[9] A.G. Bromley. 1998. Charles Babbage's Analytical Engine, 1838. *IEEE Annals of the History of Computing* 20, 4 (1998), 29–45. https://doi.org/10.1109/85.728228

[10] Harry Buhrman, Bruno Loff, Subhasree Patro, and Florian Speelman. 2022. Memory Compression with Quantum Random-Access Gates. In *Theory of Quantum Computation, Communication, and Cryptography*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPICS.TQC.2022.10

[11] Andre R. R. Carvalho, Harrison Ball, Michael J. Biercuk, Michael R. Hush, and Felix Thomsen. 2021. Error-Robust Quantum Logic Optimization Using a Cloud Quantum Computer Interface. *Phys. Rev. Appl.* 15 (Jun 2021), 064054. Issue 6. https://doi.org/10.1103/PhysRevApplied.15.064054

[12] Charles Q. Choi. 2023. IBM's Quantum Leap: The Company Will Take Quantum Tech Past the 1,000-Qubit Mark in 2023. *IEEE Spectrum* 60, 1 (2023), 46–47. https://doi.org/10.1109/MSPEC.2023.10006669

[13] Douglas Comer. 1979. Ubiquitous B-Tree. *ACM Comput. Surv.* 11, 2 (jun 1979), 121–137. https://doi.org/10.1145/356770.356776

[14] Thomas H. Cormen, Charles Eric Leiserson, Ronald Linn Rivest, and Clifford Seth Stein. 2009. *Introduction to Algorithms* (third ed.). MIT Press. 221–252 pages. https://mitpress.mit.edu/books/introduction-algorithms-third-edition

[15] Tiago M. L. de Veras, Leon D. da Silva, and Adenilton J. da Silva. 2022. Double sparse quantum state preparation. *Quantum Information Processing* 21, 6 (June 2022). https://doi.org/10.1007/s11128-022-03549-y

[16] Simon J. Devitt. 2016. Performing quantum computing experiments in the cloud. *Phys. Rev. A* 94 (Sep 2016), 032329. Issue 3. https://doi.org/10.1103/PhysRevA.94.032329

[17] Richard P Feynman. 1982. Simulating physics with computers. *International journal of theoretical physics* 21, 6/7 (1982), 467–488.

[18] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008. Quantum Random Access Memory. *Physical Review Letters* 100, 16 (April 2008). https://doi.org/10.1103/physrevlett.100.160501

[19] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. arXiv:quant-ph/9605043 [quant-ph]

[20] Lov K. Grover and Jaikumar Radhakrishnan. 2004. Quantum search for multiple items using parallel queries. arXiv:quant-ph/0407217 [quant-ph]

[21] A. Gueddana, R. Chatta, and M. Attia. 2014. CNOT-based design and query management in quantum relational databases. *International Journal of Quantum Information* 12 (2014), 1450023. https://doi.org/10.1142/S0219749914500233

[22] Gavin S. Hartnett, Aaron Barbosa, Pranav S. Mundada, Michael Hush, Michael J. Biercuk, and Yuval Baum. 2024. Learning to rank quantum circuits for hardware-optimized performance enhancement. arXiv:2404.06535 [quant-ph]

[23] Samuel Jaques and Arthur G. Rattew. 2023. QRAM: A Survey and Critique. arXiv:2305.10310 [quant-ph]

[24] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. 2024. Quantum computing with Qiskit. https://doi.org/10.48550/arXiv.2405.08810 arXiv:2405.08810 [quant-ph]

[25] Md Khalik khan and Dr. Sapna Jain. 2023. ERROR CORRECTION USING QUANTUM COMPUTING. *International Journal of Engineering Applied Sciences and Technology* (2023). https://doi.org/10.33564/ijeast.2023.v08i01.014

[26] Junyu Liu, Connor T. Hann, and Liang Jiang. 2023. Data centers with quantum random access memory and quantum networks. *Phys. Rev. A* 108 (Sep 2023), 032610. Issue 3. https://doi.org/10.1103/PhysRevA.108.032610

[27] Junyu Liu and Liang Jiang. 2024. Quantum Data Center: Perspectives. *IEEE Network* (2024), 1–1. https://doi.org/10.1109/mnet.2024.3397836

[28] Guang Hao Low, Vadym Kliuchnikov, and Luke Schaeffer. 2018. Trading T-gates for dirty qubits in state preparation and unitary synthesis. arXiv:1812.00954 [quant-ph]

[29] Ofir Milul, Barkay Guttel, Uri Goldblatt, Sergey Hazanov, Lalit M. Joshi, Daniel Chausovsky, Nitzan Kahn, Engin Çiftyürek, Fabien Lafont, and Serge Rosenblum. 2023. Superconducting Cavity Qubit with Tens of Milliseconds Single-Photon Coherence Time. *PRX Quantum* 4 (Sep 2023), 030336. Issue 3. https://doi.org/10.1103/PRXQuantum.4.030336

[30] Carla Rieger, Michele Grossi, Gian Giacomo Guerreschi, Sofia Vallecorsa, and Martin Werner. 2024. Operational Framework for a Quantum Database. arXiv:2405.14947 [quant-ph]

[31] Sudip Roy, Lucja Kot, and Christoph E. Koch. 2013. Quantum Databases. In *Conference on Innovative Data Systems Research*. https://api.semanticscholar.org/CorpusID:16242503

[32] Manuel Schönberger, Stefanie Scherzinger, and Wolfgang Mauerer. 2023. Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware. *Proc. ACM Manag. Data* 1, 1, Article 92 (may 2023), 27 pages. https://doi.org/10.1145/3588946

[33] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum-Inspired Digital Annealing for Join Ordering. *Proc. VLDB Endow.* 17, 3 (nov 2023), 511–524. https://doi.org/10.14778/3632093.3632112

[34] Xie Shi-man and Shang Xin-zhi. 2012. The Building and Optimization of Quantum Database. *Physics Procedia* 25 (2012), 1602–1609. https://doi.org/10.1016/j.phpro.2012.03.282 International Conference on Solid State Devices and Materials Science, April 1-2, 2012, Macao.

[35] Xie Shi-man and Shang Xin-zhi. 2012. The Building and Optimization of Quantum Database. *Physics Procedia* 25 (2012), 1602–1609.

[36] Luka Skoric, Dan E. Browne, Kenton M. Barnes, Neil I. Gillespie, and Earl T. Campbell. 2023. Parallel window decoding enables scalable fault tolerant quantum computation. *Nature Communications* 14, 1 (03 Nov 2023), 7040. https://doi.org/10.1038/s41467-023-42482-1

[37] Immanuel Trummer and Christoph Koch. 2015. Multiple Query Optimization on the D-Wave 2X Adiabatic Quantum Computer. arXiv:1510.06437 [cs.DB]

[38] Davide Venturelli, Minh Do, Eleanor Rieffel, and Jeremy Frank. 2018. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology* 3, 2 (Feb. 2018), 025004. https://doi.org/10.1088/2058-9565/aaa331

[39] Zhiyao Wang. 2023. Comparison of Quantum and Classical Algorithm in Searching a Number in a Database Case. *Highlights in Science, Engineering and Technology* 38 (03 2023), 370–376. https://doi.org/10.54097/hset.v38i.5831

[40] A. Younes. 2013. Database Manipulation Operations on Quantum Systems. 1 (2013), 9–17. https://doi.org/10.12785/QIR/010102