Data Management in the Noisy Intermediate-Scale Quantum Era

Rihan Hai
Delft University of Technology
r.hai@tudelft.nl

Shih-Han Hung
National Taiwan University
shihhanh@ntu.edu.tw

Tim Coopmans
Leiden University
t.j.coopmans@tudelft.nl

Floris Geerts
University of Antwerp
floris.geerts@uantwerp.be

ABSTRACT

Quantum computing has emerged as a promising tool for transforming the landscape of computing technology. Recent efforts have applied quantum techniques to classical database challenges, such as query optimization, data integration, index selection, and transaction management. In this paper, we shift focus to a critical yet underexplored area: data management *for* quantum computing. We are currently in the Noisy Intermediate-Scale Quantum (NISQ) era, where qubits, while promising, are fragile and still limited in scale. After differentiating quantum data from classical data, we outline current and future data management paradigms in the NISQ era and beyond. We address the data management challenges arising from the emerging demands of near-term quantum computing. Our goal is to chart a clear course for future quantum-oriented data management research, establishing it as a cornerstone for the advancement of quantum computing in the NISQ era.

1 INTRODUCTION

Data management is crucial in our increasingly data-driven world, exemplified by the widespread use of database systems and the rapid advancement of big data systems [6, 157]. In recent years, the field of computer science has been energized by the transformative potential of quantum technologies. Quantum computing promises computational capacities far beyond what traditional computers can achieve [124]. However, quantum computing is still in a nascent stage, i.e., the *Noisy Intermediate-Scale Quantum (NISQ)* era, characterized by quantum computers that are constrained by noise and limited numbers of qubits [138].

With the ongoing development of quantum computing, new data management challenges naturally emerge. The fundamental differences between quantum and classical computing call for novel data representations to effectively preserve quantum information [124, 174]. Moreover, many quantum computing tasks are inherently *data- and computation-intensive* due to the need to handle large-scale quantum states [25], multidimensional quantum data structures [18, 186], and error correction codes [59, 60, 69, 159]. For example, simulating large-scale quantum computation on classical computers involves processing vast amounts of quantum information, leading to significant scalability and optimization challenges [25, 102, 184, 186]. By addressing data management challenges in the NISQ era, the database community has a unique opportunity to significantly enhance the scalability and reliability of quantum technologies, which will advance both research and real-world quantum applications.

These new data management challenges are not yet clearly defined, despite the recent efforts by the database community. Existing work has explored leveraging quantum computers as new hardware to address classical database challenges, such as query optimization [55, 56, 120, 145–148, 163, 179], data integration [61], index selection [71, 89], and transaction management [19, 70, 161]. However, fundamental questions about data management in the NISQ era remain unanswered. Specifically, how should we define and manage data in the context of near-term and future quantum advancements? Given the unique features of quantum computing, such as superposition and entanglement, what are the new considerations to be addressed for effective data management? What data structures and data management systems will best support the development of quantum technologies, particularly given a limited number of qubits, noise, and other challenges of the NISQ era?

While recent vision papers, tutorials, and surveys [31, 72, 180, 187, 188] have begun discussing the intersection of data management and quantum computing, they primarily focus on how quantum technologies can accelerate classical database operations. In contrast, our work delves into an equally important yet underexplored area: *data management for quantum computing*. At the moment, the fundamental concepts, research directions, and problem definitions in this area remain obscure within the database community. This paper initiates the exploration of these critical aspects and lays the foundation for further deeper integration of data management and quantum computing. We will not dive into the basics of quantum computing and information, as ample resources are available [18, 140, 184, 186]. Instead, we focus on outlining the vision for data management paradigms in the NISQ era and identifying potential research challenges that are particularly relevant to the database community, which could have a significant impact on the advancement of today's quantum technologies.

Contributions. Our contributions are summarized as follows:

- **Fundamentals**: We explain quantum data, differentiate it from classical data (Sec. 2).
- **Roadmap**: We present our vision for data management research for quantum computing in three paradigms (Sec. 3).
- **Research problems**: We elaborate on near-term research questions in data management for quantum computing (Sec. 4).

2 DATA IN THE QUANTUM ERA

Since the 1960s, database management systems (DBMSs) have been developed and operated on classical computers, which rely on bits as fundamental units of data. Classical bits represent binary values

of 0 or 1. In the development of quantum computers and their associated software systems, a key challenge is the implementation of qubits [136], the quantum counterparts of classical bits. Following the principles of quantum mechanics, qubits differ from bits in properties such as superposition and entanglement. These properties fundamentally differentiate quantum data from classical data, which we will explore in this section.

2.1 Classical data vs. quantum data

Classical data is the information that is collected, processed, and stored with traditional computing methods. Today, most of the classical data is stored and queried using database systems such as relational databases, document stores, graph databases, and vector databases [154]. We refer to *quantum data* as information collected and processed using *quantum computing devices*, i.e., computing devices that follow the rules of quantum mechanics to their advantage [124]. Quantum data is represented by qubits. Next, we list key differences between quantum and classical data below to help understand the unique features of quantum data.

1. Quantum data is probabilistic. Unlike a classical bit, which is 0 or 1, a quantum bit can be in *superposition*. Mathematically, a zero state and a one state may be represented by a unit vector in the *standard basis*. That is, the zero state $|0\rangle$ is represented by the vector $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ and the one state $|1\rangle$ is represented by the vector $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$. A single qubit state, denoted $|\psi\rangle$, may be represented by a superposition, i.e., a linear combination of $|0\rangle$ and $|1\rangle$: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, for some pair of complex numbers α, β called *amplitudes*, which satisfy $|\alpha|^2 + |\beta|^2 = 1$. The probabilistic nature arises when a *measurement* is performed. When a measurement is performed on the state $|\psi\rangle$, an outcome 0 is obtained with probability $|\alpha|^2$ and 1 obtained with probability $|\beta|^2$, and the state permanently changes to the obtained outcome.

Example 2.1 (Superposition). Consider the qubit in the following superposition:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

When measuring this qubit, there is an equal probability of 50% ($|\alpha|^2 = |\beta|^2 = \frac{1}{2}$) to get a 0 or 1 as a result.

In classical computing, individual bits can be concatenated to form a bit string, e.g., the three-bit string "010." Similarly, we may concatenate qubits into a multi-qubit state. In this case, an n-qubit state can be represented as a superposition of $|x\rangle$ for $x \in \{0,1\}^n$, or equivalently a vector of 2^n components. Later in Sec. 4, we will discuss a three-qubit state in Fig. 2a.

2. Quantum data is fragile. Quantum computers are anticipated to outperform classical computers in solving certain problems. However, with current quantum technology in the NISQ era, quantum resources remain scarce, and quantum data is prone to noise. *Decoherence* [137], is the process where quantum states lose their coherence due to environmental interactions, resulting in the gradual loss of quantum information. Quantum noise, resulting from unintended couplings with the environment, can significantly degrade the performance of quantum computers [165]. Commonly used noise models, such as amplitude damping, phase damping, bit-flip, and phase-flip, mathematically describe how various types of quantum noise lead to decoherence [124].

3. Quantum data can be entangled. Another difference between qubits and bits is *entanglement* [52], leading to *spooky action at a distance*, a phrase coined by Einstein. Imagine two qubits with qubit A being in Amsterdam and qubit B in San Francisco. When qubit A and qubit B are entangled this means their states are correlated. The *spooky* part is that if we measure qubit A in Amsterdam and find it in a state of 0 (or 1), we instantaneously know the state of qubit B is 0 (or 1), even though it is far away in San Francisco. This correlation is maintained independently of the physical distance between them. The next example introduces *Bell's state*, a well-known entangled state.

Example 2.2. (Bell's state) We extend the one-qubit state in Example 2.1 to a two-qubit state as follows:

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\0\\1\end{bmatrix}$$

2.2 Quantum-classical data transformation

The transformations that map quantum data to classical data, and vice versa, are essential ingredients of quantum computing. These transformations can be described as quantum algorithms.

Quantum algorithms, circuits, and gates [124]. A *quantum algorithm* is a carefully designed quantum process that maps input data, classical or quantum, to the desired output data. The most commonly used model to describe quantum algorithms is the *quantum circuit* model. A quantum circuit is a sequence of operations, *quantum gates*, which manipulate qubits to perform computations. Quantum gates are *unitary operators*. An operator U is unitary if its conjugate transpose $U^\dagger$ multiplied by U (and vice versa) equals the identity matrix I, denoted as $U^\dagger U = UU^\dagger = I$. A unitary operator may be composed as a sequence of basic unitary operators. A set of quantum gates is universal if it can approximate any unitary operator to desirable precision. A widely known universal gate set consists of Pauli gates (including X, Y, Z gates defined by Pauli matrices), the Hadamard gates (see Example 2.3), the T gate, and the controlled-NOT (CNOT) gate.

Example 2.3. The Hadamard gate (H) is a unitary operator, which maps $|0\rangle$ to $(|0\rangle + |1\rangle)/\sqrt{2}$, and $|1\rangle$ to $(|0\rangle - |1\rangle)/\sqrt{2}$. In matrix form, the operator is represented as

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

3 OUR VISION: DATA MANAGEMENT FOR QUANTUM COMPUTING

To facilitate future research on data management for quantum computing, we first sketch the whole landscape in Fig. 1. We categorize this landscape into three distinct paradigm based on how quantum and classical data are transformed and utilized, and the type of hardware involved—whether a quantum or classical computer is employed. Each paradigm not only guides the exploration of quantum research topics leveraging database technologies but also paves the way for new opportunities in database systems, ultimately driving the development of quantum computing. The goal of this section is to introduce data management researchers to quantum use cases, and by making a distinction based on the nature and role that data in it plays, focus future data management research for

quantum computing. In Sec. 4, we will provide concrete research directions in one of the paradigms we distinguished.

***I Classical simulation of quantum computing paradigm*: *classical data represents quantum states and operations*.** This paradigm presents significant potential for new database challenges. It focuses on using *classical data to represent and simulate quantum data*. For example, Bell's state from Example 2.2 is represented by a classical vector. Research in this paradigm is more accessible because it does not require a quantum computer; instead, it involves classical computers.

The representative task in this paradigm is *simulation*. Simulation is the process of emulating quantum computation, enabling researchers to model and analyze quantum processes as if they were operating on actual quantum hardware [184, 186].[1] Simulations are of paramount importance in the NISQ era [138]. Consider, for instance, the concept of quantum supremacy [25], which seeks to demonstrate the superior capabilities of quantum computers over classical computers. Given that large-scale quantum computers are not yet available in the NISQ era, simulation is essential for comparing the scalability of quantum computers with classical ones. Additionally, simulations play a crucial role in the development of new quantum algorithms, allowing researchers to design, debug, and validate the correctness of these algorithms before deploying them on expensive quantum devices. Moreover, simulations aid in the development of quantum hardware by evaluating error mitigation schemes, predicting algorithm runtimes, and more [13, 85, 171, 191, 193]. Simulation serves as a foundational tool across key areas of quantum computing, including quantum supremacy, quantum algorithms, quantum hardware, error correction, and the exploration of potential quantum applications [184].

The potential for database research in this paradigm is immense. First, imagine innovative database systems specifically designed to manage classical data that supports simulation. Such systems could handle complex queries, such as updating quantum-state representations as gates are applied, returning classical measurement outcomes, or calculating the closeness of the current quantum state to a desired target state [174].

Second, efficient caching strategies for expensive operations, such as repeated simulations of quantum circuits with varying error parameters, present another important research direction. These strategies require the development of efficient data management systems that optimize the computation of such operations.

The third research direction in this paradigm involves the representation of quantum states and operations. Not only can a quantum state be represented as classical data (e.g., vectors), but quantum gates can also be represented as classical data, such as matrices [82, 167]. This is useful for quantum design questions such as optimizing the number of quantum gates or gate depth [92, 192] and compiling textbook quantum circuits to only include the low-level operations that real quantum devices support [153]. As the representation of a quantum state (a vector) or quantum gate (a matrix) generally grows exponentially with the number of qubits, developing efficient methods to store and process these data structures is key to making simulations of quantum computation feasible, either in vector form or through more compressed structures like tensor networks [18, 127] or other advanced representations [27, 87, 167, 193].

***II Joint Quantum-Classical Computing paradigm*: *classical preprocessing & postprocessing for quantum computing*.** This paradigm focuses on the situation involved in which most quantum-technology applications: classical computers handle the preprocessing and postprocessing of data for quantum devices, such as quantum chips, quantum sensors, and quantum computers. In this paradigm, the focus is on data which is primarily stored and processed as *classical data*. For instance, a classical computer may send instructions to a quantum chip. These instructions are classical data and, for example, describe the quantum circuit that should be run. The quantum device runs its computation and, after some time, performs measurements onto one or more of its qubits. The measurement outcomes, which are bits and hence classical data, are then returned to the classical computer. Thus, the quantum device receives classical input (the instructions) and returns the classical output (the measurement outcomes).

Database research can enhance this process by developing efficient systems for managing, storing, and querying the classical data involved in preprocessing, postprocessing, and iterative feedback loops between classical computers and quantum devices, ensuring seamless integration and optimization of data workflows. For instance, the development of quantum error correction codes—a critical area in quantum computing—can benefit significantly from efficient graph data analytics techniques, as further discussed in Sec 4.3.

We demonstrate the importance of this paradigm through three key categories of quantum applications: first, applications where the quantum computation is completed immediately upon returning a result. For example, two separated quantum computers can generate a secure, shared key (password) by encoding bits into qubits, then sending and measuring the qubits [16, 135]. This is followed by classical postprocessing on the bitstring (i.e. classical data) that the measurement returns [53]. The resulting bitstring is a secret key that can later be used for secure communication. In the second category, the quantum computation is stalled temporarily, and the quantum chip still holds quantum data on which the computation will continue later. An example is the detection of errors during a quantum computation, where at fixed timesteps during the computation check measurements are performed [69, 159], whose outcomes are then decoded to find out which error has most likely occurred [14, 110]. Third, efficient preprocessing is critical for quantum algorithms, which requires loading classical data into the quantum domain, typically by mapping classical bits onto quantum bits. This process often requires encoding classical data into a quantum circuit, either as part of the algorithm [158] or to output a quantum state with amplitudes representing the classical data [41]. For example, the Harrow–Hassidim–Lloyd (HHL) algorithm, a well-known quantum approach for solving a system of linear equations [74], achieves exponential speedup over classical methods. However, this requires encoding the problem's vector elements into the amplitudes of the input quantum state. Developing a general and efficient method to encode arbitrary vectors into quantum states for the HHL algorithm remains an open research question.

[1] In this work, by simulation we refer to *classical simulation*. Another related term is *quantum simulation* [30, 66], which pertains to simulating quantum mechanics on a quantum computer, a subject studied in physics.

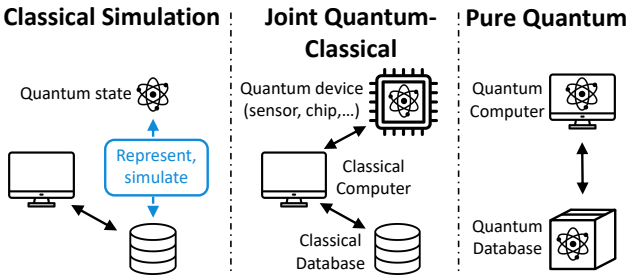


Figure 1: Landscape: data management for quantum computing

III Pure quantum computing paradigm: quantum-native data storage and processing. Finally, we envision a future research paradigm beyond NISQ, when we have large-scale, fault-tolerant quantum computers. We will deal with pure *quantum data*, i.e., qubits. Here, the quantum hardware supports storing qubits for long enough to perform other tasks in the meantime. This enables for example a cloud quantum computer [29] who rotates its resources among multiple clients, various types of quantum machine learning [32], and Quantum Random Access Memory [67], where data can be queried in superposition. A recent vision [80] shows a possible future: quantum data is collected from quantum sensing systems, e.g., for discovering a black hole; then stored and processed via the quantum memory of a quantum computer.

The data management research in this paradigm will be centered around handling quantum data, possibly developing quantum-native algorithms. Moreover, storing quantum data will remain challenging, as quantum data storage will still be costly in various ways (the number of qubits is not unbounded, robust storage of quantum data requires large-scale quantum error correction, which is computationally intensive, etc.). Another research topic is to efficiently allocate the available qubits. This includes, for example, various scheduling and allocation tasks [34, 57, 64, 77, 114, 151] and efficient use of different quantum hardware types each of which has speed-decoherence trade-offs [49]. Given the amount of quantum data in this paradigm, most research here will call for novel quantum data processing algorithms and systems beyond NISQ era.

4 NEAR-TERM RESEARCH PROBLEMS

We will now explore data management research questions across the three paradigms. In Sec. 4.1 and 4.2, we focus on a key challenge within the I Classical simulation of quantum computing paradigm. In Sec. 4.3, we broaden the scope to include research opportunities across all three paradigms.

4.1 The challenge of simulating quantum computation

The input for a simulation is a quantum algorithm described as a quantum circuit,[2] consisting of input qubits upon which quantum gates are applied, ultimately resulting in a probability vector of the output states, i.e., measurement outcomes (see Sec. 2.2). Then, a simulation is a classical computation that computes that output

[2]A uniform family of circuits, to be precise, one for each input size.

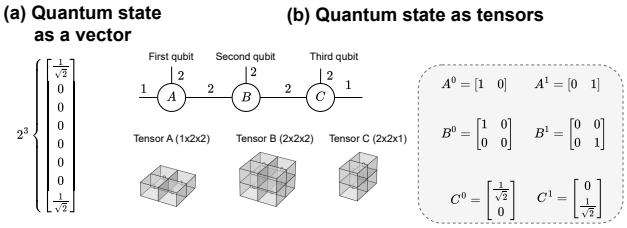


Figure 2: (a) The 3-qubit GHZ state $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ [124] represented in state vector, whose size is $2^3 = 8$. (b) GHZ state represented as a tensor network [126, 134]. For better visualization, in the grey box, we write the tensors in vector/matrix form by slicing the last dimension of A, B, and C.

distribution (so-called *strong* simulation), or samples from it (*weak* simulation) [169]. We primarily focus on strong simulation.

The major practical bottleneck of (strong) simulation is *scalability* [25, 102, 184, 186]. When simulating a quantum state of n qubits, the state is typically represented as a vector with a size of 2^n. The size of the state vector grows exponentially as n increases. For instance, an experiment demonstrating quantum supremacy required 2.25 petabytes for 48 qubits, reaching the memory limits of today's supercomputers [25]. To overcome the memory restriction, existing simulation tools have explored approximation [86, 112, 170], data compression [182], parallelization [79], and distributed computing [152]. In Fig. 2 we illustrate concepts mentioned in this section.

4.2 Databases to the rescue

Simulation offers significant opportunities for database research, and conversely, database expertise can greatly enhance simulation.

> We envision a *classical-quantum simulation system* (CQSS) with the following capabilities: (i) automatically providing the most efficient simulation of the input circuit by selecting optimal data structures and operations based on available resources and circuit properties; (ii) operating inherently out-of-core to support the simulation of large circuits that exceed main memory capacity; (iii) ensuring consistency to prevent data corruption and enabling recovery in the event of large-scale simulation crashes; and (iv) improving the entire simulation workflow, including parameter tuning, data collection and querying, exploration and visualization.

At its core, a CQSS must, at a minimum, be capable of evaluating quantum circuits. This primarily involves performing linear algebraic operations, often described in terms of *tensor networks* [17].[3] A *tensor* is a multidimensional array, and the dimensions along which a tensor extends are its *indices*. A tensor network consists of vertices representing tensors, and edges the indices. Free indices are depicted as legs, i.e., edges that only connect to one vertex in the graph. Connecting two vertices by joining a leg corresponds to the contraction with the corresponding indices. Fig. 2b shows an example of the tensor network representation of GHZ state. Tensor

[3]For simplicity of exposition, we do not consider non-tensor-based simulation methods which represent quantum states by their symmetries, rather than by complex-valued vectors [68].

contraction is the summation of shared indices between tensors and is a generalization of matrix multiplication [18], as shown in Fig. 2c. The first question that comes to mind is as follows.

Q1. *Should we push the simulation workload to existing DBMSs?*

In other words, should we map the simulation workload to SQL queries and store the results in a DBMS? The advantage is that we can benefit from the efficiency and portability of modern database systems. And indeed, initial efforts from the database community [20, 162] have begun to address the out-of-core simulation challenge by mapping quantum states and gates to SQL queries. For instance, in [20], qubit states and gates are represented as tensor networks, with sparse tensors in COO format[4] and Einstein summation operations mapped to SQL queries. Additionally, the recent abstract [162] introduces the idea of supporting dense vectors using SQL UNION ALL and CASE statements. However, despite these early efforts that support basic quantum states and gates up to 18 qubits and a circuit depth of 18 in separate experiments, the core challenges of representing general quantum states and operations in DBMSs and achieving scalability remain unresolved. Furthermore, preliminary findings question whether query optimizers in existing RDBMSs can effectively optimize SQL queries for simulation workloads [20]. Therefore, designing intermediate representations [75, 88] and seeking more effective query optimization, potentially beyond existing techniques [121], could be crucial and require further exploration. A forward-thinking extension of Q1 is to choose most suitable databases for simulation. NoSQL databases such as TileDB, an array database, offers native storage and retrieval of multi-dimensional arrays (tensors), leveraging parallel processing, distributed computing, and easy integration with machine learning and data science tools. The downside is that much effort is needed to support in-database computation and query optimization for simulation workload and expand the ecosystem to match the versatility of RDBMSs, highlighting critical areas for future research in adapting array databases for simulation workloads.

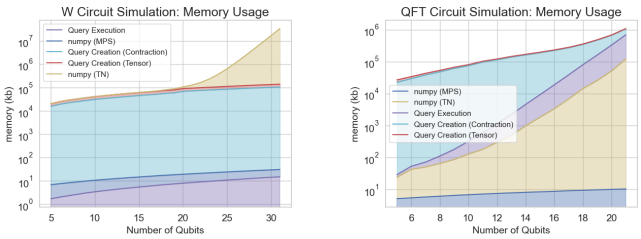
Q1 summary. Leveraging existing DBMSs for simulation workloads offers potential efficiency gains; however, critical challenges in scaling and optimizing SQL queries for complex quantum simulations remain unaddressed.

On the other hand, the connection to tensor computations suggests the following question.

Q2. *Should we instead leverage tensor-based database technologies for simulation?*

Indeed, the database community has a substantial body of work on tensor computation, especially with the recent synergies between databases and machine learning (ML) such as *learning ML models over relational data* [22, 91, 107, 130, 190]. These studies range from high-level representations to the runtime optimization of ML workflows. At the representation level, numerous studies represent data as tensors [95] and aim to integrate relational and tensor operations [81, 90, 142]. Meanwhile, system-building endeavors from the database community [21, 21, 63, 75, 105, 117, 125, 133, 143, 144] focus on performance efficiency and scalability, employing optimization

[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.html



(a) W state prep. (sparse circuit) **(b) QFT on $|0..0\rangle$ (dense circuit)**

Figure 3: Memory requirement comparison of RDBMS-based vs. numy-based solutions

techniques during compilation (e.g., operator fusion [22, 23]) and runtime (e.g., parallelization [24]). Similar endeavours exist in the areas of compilers and High Performance Computing [36, 84, 93, 141].

Our envisioned classical-quantum simulation system holds the potential to achieve optimal performance. By integrating relevant optimization techniques from the aforementioned existing systems, we can facilitate the automatic optimization of the underlying tensor network. The actual runtime performance will depend on several factors: the operations within the tensor network, the sparsity of the tensors involved, the memory layouts (i.e., how tensors are stored in memory, such as row-major or column-major formats), and the available hardware [22, 144].

Q2 summary. Tensor-based database technologies are promising for boosting simulation performance, with efficiency depending on tensor operations, sparsity, memory layouts, and hardware.

While both machine learning and quantum applications share fundamental building blocks like matrices, tensors, and tables, we must address challenges unique to quantum computing. We highlight two key gaps that need to be overcome:

4.2.1 Quantum-specific optimizations. Existing simulators for quantum computing are mostly in Python [44, 172], C++ [8, 172] or Julia [106]. There are also dedicated quantum programming languages and domain-specific compilation techniques. See the survey [76]. In addition, tensor network rewriting techniques have been considered in the ZX-calculus [38]. ZX-calculus is a graphical language for representing and reasoning about quantum computations, represented as tensor networks with specialized tensors called *spiders* [168]. Its graphical nature enables simplifications by locally merging connected vertices while preserving the original tensor, useful in various contexts including formal verification [48, 100, 168].

Quantum-specific optimizations share some conceptual similarities with classical database optimization, yet they diverge significantly due to the unique characteristics of quantum data and operations. First, classical databases rely on relational or NoSQL data models (relational, graph, document-based, etc.) with indexing mechanisms like B-trees or hash tables to facilitate efficient access and retrieval. In simulations, however, quantum states are represented as state vectors (Fig. 2a) or using more advanced data structures such as tensor networks (Fig. 2b). Second, in classical databases, query optimization involves techniques like join ordering to identify the optimal ordering of join operations between relations for an efficient query plan [101, 156, 164]. For tensor-based simulators, a defining feature is the optimization of *contraction order*, which determines the most efficient sequence to contract

tensors representing quantum states [44, 186]. Contraction order optimization parallels the role of join order optimization in traditional databases but is more complex due to the exponential growth of quantum state dimensions and the unique features of quantum data, such as entanglement. Third, classical database systems rely heavily on cost estimation to predict and minimize resource use for query execution. Quantum computation simulators, however, lack cost estimation frameworks and instead use optimizations like parallelization, SIMD (Single-Instruction, Multiple-Data) processing, and matrix decomposition techniques (e.g., SVD) to improve performance [186]. Quantum-specific issues, such as noise models (Sec. 2.1), complicate simulation further. These differences lead to the need for fundamentally new optimization frameworks to manage and simulate quantum computation effectively, beyond classical structures and processes." The following question now arises.

Q3. *Can we build an optimizer for simulations?*

Such an optimizer should, given a circuit, determine the most efficient way to simulate it. For example, when a circuit allows for a tractable simulation (e.g., circuits of bounded treewidth), the optimizer should compile it into an efficient simulation. Similarly, based on the sparsity of (intermediate) quantum states, specialized sparse gate computations may be employed instead of the standard dense computations.

An open question is when it is advantageous to leverage RDBMSs for simulation workloads. Fig. 14a presents preliminary experimental results for sparse circuits, GHZ state preparation circuits involving sparse tensor computations, where we observe that pushing simulation workloads to RDBMSs reduces memory consumption compared to a numpy-based baseline as qubit count increases. Interestingly, this benefit does not extend to dense circuits, such as quantum fourier transform (QFT) circuits, shown in Fig. 14b, where dense tensor computations negate these gains. We are currently developing a cost estimator in CQSS that considers sparsity, and our preliminary experiments reveal additional factors that impact optimization design, including qubit and gate counts, entanglement, circuit structure (e.g., sparse/dense, network topology), and hardware configuration (CPU/GPU availability). Details on the experimental setup, implementation, and full results are available on GitHub: .

Particular to the quantum setting is the noisy character of the quantum computation, which is known to impact the efficiency of simulation [7, 78, 79, 132, 139]. Hence, an optimizer needs to take noise levels into account. Moreover, the optimizer should consider the downstream task – whether it's computing precise probabilities for strong simulations or sampling measurement outcomes for weak simulations – and select the appropriate simulation strategy (algorithm) accordingly.

As part of the optimizer, we may also want to check whether two quantum states are exactly or approximately equivalent, analogous to checking equivalent conjunctive queries [35, 39, 94]. Having algorithms in place for testing equivalence may help to avoid redundant computations during simulation.

Q3 summary. Developing an optimizer could enhance efficiency by selecting optimal strategies based on circuit structure, state sparsity, noise levels, and task requirements, while also reducing redundancy through state equivalence checks.

4.2.2 Quantum-specific data representations. Simulating quantum circuits requires precise and efficient representations of quantum states and operations, accounting for the complex nature of quantum mechanics. Ensuring accuracy and scalability as the system grows in complexity is crucial. We mentioned tensor networks as a way to represent the simulation, but various other data representations exist [174], e.g., based on algebraic decision diagrams, matrix product states, just to name a few.

Q4. *What are good data representations – possibly beyond tensors – for supporting simulations?*

An operation on a data structure is *tractable* if it executes in polynomial time with respect to the input size [47]. The choice of data representation can significantly impact whether certain quantum operations (e.g., gates, measurement explained in Sec. 2.2) are tractable [174]. We emphasize three main requirements for data representation: *expressiveness*, *closure*, and *succinctness/tractability*.

For expressiveness, the data representation must be sufficiently rich to represent quantum states and measurements. Closure refers to the property that the result of quantum operations on the represented states can also be represented within the same data representation. This property ensures compositionality. Finally, it is crucial to minimize the space required to represent a quantum state, as the number of qubits and the level of entanglement can result in exponentially large states. Therefore, we seek data representations that are succinct, minimizing the space needed to store data while still allowing for efficient query processing. It may also be interesting to consider these questions beyond tensor network representations as well [27, 87, 167, 193]. A concrete idea for a novel data representation is to combine matrix product states (MPS) and the recently-proposed *local-invertible map decision diagram* (LIMDD) [173]. LIMDD compresses a state vector by lumping together parts of the vector that are equivalent modulo simple quantum gates. In polynomial time and space in the number of qubits, LIMDDs can simulate circuits that MPS cannot and vice versa [174]. One approach to combine these strengths in a single data structure is applying the lumping to the MPS matrices instead of quantum state vectors, by building upon existing work on equivalence characterizations of MPS [98].

Q4 summary. Effective data representations for quantum simulations must balance expressiveness, closure, and succinctness.

Q5. *Are there other benefits from relying on database technology?*

By using database technology we aim for the simulation of complex circuits that require significant memory, without relying on HPC infrastructures or the operating system's memory management. By controlling what data is pushed to secondary storage during simulations, we can achieve more efficient I/O behavior. Additionally, transaction management and recovery become important—where a transaction could, for example, represent a sequence of quantum gates—ensuring that computations can restart from saved checkpoints while maintaining the correctness of partially saved results. Parallel evaluation strategies in quantum circuit simulations also highlight the need for effective transaction scheduling,

leading to more reliable computational processes. Undoubtedly, many challenges research questions are in this context when considering quantum simulation. We prioritize Q1–Q4, however, as first steps.

4.3 More data management opportunities

Beyond simulation for quantum computing, quantum-related research is broad, e.g., error correction [69, 150, 155], quantum networks [45, 177]. Next, we expand our discussion to uncover more opportunities spanning all three paradigms shown in Fig. 1.

Quantum error correction & Graph analytics. Quantum error correction (QEC) enables reliable execution of a quantum computation on noisy quantum processors and is a crucial part of the roadmap to fault-tolerant quantum computation [159]. QEC process consists of two main aspects: coding and decoding. The input quantum information as well as the operations to be performed are coded using a quantum error-correcting code such as the surface code [26, 59, 60]. Intermittently, decoding is applied: errors are detected and corrected. Decoders [50, 58, 183] often leverage graph theory, solving tasks like a minimum-weight perfect matching (MWPM) problem on a graph where each node represents a check measurement. For MWPM, the number of nodes is of the same order as the number of qubits. Many interesting research problems arise how to model such graphs and design the data formats. Moreover, QEC has to be performed at high speeds to avoid decaying quantum states over time (see Sec. 2), e.g. at most several microseconds for some types of quantum hardware for decoding [14]. The scalability to many qubits, and speed required for QEC might benefit from advanced data management tools and techniques [15, 108, 111, 176, 185], offering a promising direction for further exploration.

Quantum experiments & Scientific data management. Quantum experiments, like any other scientific experiments, generate data and require data management. For instance, quantum mechanics experiments on supercomputers often apply HDF5 files to store data [43]. Another compelling direction is to explore how to build specialized data lakes [73, 118] or lakehouses [11, 12, 83] to support scientific data management for quantum computing.

ML & Quantum data management. i) Simulation: as explained in Sec. 4.2.1, a key challenge in optimizing tensor network based simulation is to find optimal tensor contraction order. A recent research direction is to apply machine learning (ML), specifically reinforcement learning (RL) and graph neural networks (GNN), to optimize tensor contraction order, addressing the computationally intensive nature of this challenge [104, 113]. ii) Error correction: decoding methods for quantum error correction (QEC), like MWPM, face scalability challenges in large quantum systems, where rapid error detection within strict time limits is essential [175]. An interesting new direction is *data-driven QEC*, which employs ML techniques to quantum error correction, such as RL [10, 119, 189], multilayer perceptrons (MLP) [37], convolutional neural networks [33], and GNN [99]. iii) Improving quantum algorithm efficiency: Quantum algorithms are represented and implemented as quantum circuits, where efficiency can be improved by reducing costly gates like SWAP gate or by minimizing circuit depth and gate count. ML methods, specifically RL [54, 62] and MLP [131], have shown potential in optimizing circuit design to enhance the efficiency of quantum circuits on real devices.

5 CONCLUSION

We are currently at a privileged time in the evolution of data management, closely aligned with the rise of quantum computing. This convergence calls for innovative approaches to data representation, processing, and querying that are compatible with quantum computing. In this work, we pave the path for future quantum data management by the clarification of the unique features of quantum data compared to classical data, and the exploration of three data management paradigms, which reveal a rich field of complex and significant challenges. As we continue to explore these paradigms, it becomes clear that the challenges we face, such as enhancing simulations with database technologies, are just the beginning. There exists a broader spectrum of challenges that remain unexplored, which require sustained and focused efforts from both the data management and quantum computing communities.

REFERENCES

[1] 2019. Amazon Braket: API Reference. https://docs.aws.amazon.com/pdfs/braket/latest/APIReference/amazon-braket-api.pdf.
[2] 2024. Cirq import/export circuits. https://quantumai.google/cirq/build/interop.
[3] 2024. MongoDB. https://www.mongodb.com/.
[4] 2024. Qiskit documentation: QuantumCircuit class. https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.QuantumCircuit.
[5] 2024. TileDB data ingestion. https://cloud.tiledb.com/academy/structure/life-sciences/single-cell/tutorials/data-ingestion/index.html.
[6] Daniel Abadi, Rakesh Agrawal, Anastasia Ailamaki, Magdalena Balazinska, Philip A Bernstein, Michael J Carey, Surajit Chaudhuri, Jeffrey Dean, AnHai Doan, Michael J Franklin, et al. 2016. The Beckman report on database research. *Commun. ACM* 59, 2 (2016), 92–99.
[7] Dorit Aharonov, Xun Gao, Zeph Landau, Yunchao Liu, and Umesh Vazirani. 2023. A Polynomial-Time Classical Algorithm for Noisy Random Circuit Sampling. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*. 945–957.
[8] Quantum AI. 2021. qsim and qsimh. https://quantumai.google/qsim/overview. Accessed: 2024-08-05.
[9] Einstein Albert. 1916. The foundation of the general theory of relativity. *Annalen der Physik* 354, 7 (1916), 769.
[10] Philip Andreasson, Joel Johansson, Simon Liljestrand, and Mats Granath. 2019. Quantum error correction for the toric code using deep reinforcement learning. *Quantum* 3 (2019), 183.
[11] Michael Armbrust, Tathagata Das, Liwen Sun, Burak Yavuz, Shixiong Zhu, Mukul Murthy, Joseph Torres, Herman van Hovell, Adrian Ionescu, Alicja Łuszczak, et al. 2020. Delta lake: High-performance ACID table storage over cloud object stores. *Proceedings of the VLDB Endowment (PVLDB)* 13, 12 (2020), 3411–3424.
[12] Michael Armbrust, Ali Ghodsi, Reynold Xin, and Matei Zaharia. 2021. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*.
[13] Koji Azuma, Stefan Bäuml, Tim Coopmans, David Elkouss, and Boxi Li. 2021. Tools for quantum network design. *AVS Quantum Science* 3, 1 (2021), 014101.
[14] Francesco Battistel, Christopher Chamberland, Kauser Johar, Ramon WJ Overwater, Fabio Sebastiano, Luka Skoric, Yosuke Ueno, and Muhammad Usman. 2023. Real-time decoding for fault-tolerant quantum computing: Progress, challenges and outlook. *Nano Futures* 7, 3 (2023), 032003.
[15] Soheil Behnezhad, Mohammadtaghi Hajiaghayi, and David G Harris. 2023. Exponentially faster massively parallel maximal matching. *Journal of the ACM (JACM)* 70, 5 (2023), 1–18.
[16] Charles H Bennett and Gilles Brassard. 1984. Quantum cryptography: Public key distribution and coin tossing. *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing (ICCCSP)*, 175–179.
[17] Jacob Biamonte. 2019. Lectures on quantum tensor networks. *arXiv preprint arXiv:1912.10049* (2019).
[18] Jacob Biamonte. 2020. Lectures on Quantum Tensor Networks. (2020). arXiv:1912.10049 [quant-ph]
[19] Tim Bittner and Sven Groppe. 2020. Avoiding blocking by scheduling transactions using quantum annealing. In *Proceedings of the 24th Symposium on*

International Database Engineering & Applications (IDEAS). Article 21, 10 pages.

[20] Mark Blacher, Julien Klaus, Christoph Staudt, Sören Laue, Viktor Leis, and Joachim Giesen. 2023. Efficient and Portable Einstein Summation in SQL. *Proceedings of the ACM on Management of Data (PACMMOD)* 1, 2, Article 121 (jun 2023), 19 pages.

[21] Matthias Boehm, Iulian Antonov, Sebastian Baunsgaard, Mark Dokter, Robert Ginthör, Kevin Innerebner, Florijan Klezin, Stefanie N. Lindstaedt, Arnab Phani, Benjamin Rath, Berthold Reinwald, Shafaq Siddiqui, and Sebastian Benjamin Wrede. 2020. SystemDS: A Declarative Machine Learning System for the End-to-End Data Science Lifecycle. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*.

[22] Matthias Boehm, Matteo Interlandi, and Chris Jermaine. 2023. Optimizing Tensor Computations: From Applications to Compilation and Runtime Techniques. In *Companion of the 2023 International Conference on Management of Data (SIGMOD)*. 53–59.

[23] Matthias Boehm, Berthold Reinwald, Dylan Hutchison, Prithviraj Sen, Alexandre V Evfimievski, and Niketan Pansare. 2018. On Optimizing Operator Fusion Plans for Large-Scale Machine Learning in SystemML. *Proceedings of the VLDB Endowment (PVLDB)* 11, 12 (2018).

[24] Matthias Boehm, Shirish Tatikonda, Berthold Reinwald, Prithviraj Sen, Yuanyuan Tian, Douglas Burdick, and Shivakumar Vaithyanathan. 2014. Hybrid Parallelization Strategies for Large-Scale Machine Learning in SystemML. *Proceedings of the VLDB Endowment (PVLDB)* 7, 7 (2014), 553–564.

[25] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J Bremner, John M Martinis, and Hartmut Neven. 2018. Characterizing quantum supremacy in near-term devices. *Nature Physics* 14, 6 (2018), 595–600.

[26] Sergey Bravyi, Matthias Englbrecht, Robert König, and Nolan Peard. 2018. Correcting coherent errors with surface codes. *npj Quantum Information* 4, 1 (2018), 55.

[27] Sergey Bravyi and David Gosset. 2016. Improved Classical Simulation of Quantum Circuits Dominated by Clifford Gates. *Physical Review Letters* 116 (Jun 2016), 250501. Issue 25.

[28] Robert Brijder, Floris Geerts, Jan Van den Bussche, and Timmy Weerwag. 2019. MATLANG: Matrix operations and their expressive power. *ACM SIGMOD Record* 48, 1 (2019), 60–67.

[29] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. 2009. Universal Blind Quantum Computation. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 517–526.

[30] Iulia Buluta and Franco Nori. 2009. Quantum Simulators. *Science* 326, 5949 (2009), 108–111.

[31] Umut Çalikyilmaz, Sven Groppe, Jinghua Groppe, Tobias Winker, Stefan Prestel, Farida Shagieva, Daanish Arya, Florian Preis, and Le Gruenwald. 2023. Opportunities for Quantum Acceleration of Databases: Optimization of Queries and Transaction Schedules. *Proceedings of the VLDB Endowment (PVLDB)* 16, 9 (2023), 2344–2353.

[32] Marco Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Lukasz Cincio, and Patrick J Coles. 2022. Challenges and opportunities in quantum machine learning. *Nature Computational Science* 2, 9 (2022), 567–576.

[33] Christopher Chamberland, Luis Goncalves, Prasahnt Sivarajah, Eric Peterson, and Sebastian Grimberg. 2023. Techniques for combining fast local decoders with global decoders under circuit-level noise. *Quantum Science and Technology* 8, 4 (jul 2023), 045011. https://doi.org/10.1088/2058-9565/ace64d

[34] Aparimit Chandra, Wenhan Dai, and Don Towsley. 2022. Scheduling quantum teleportation with noisy memories. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 437–446.

[35] Ashok K. Chandra and Philip M. Merlin. 1977. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing (STOC)*. 77–90.

[36] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. 2018. TVM: An automated End-to-End optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 578–594.

[37] Cheng Chu, Nai-Hui Chia, Lei Jiang, and Fan Chen. 2022. Qmlp: An error-tolerant nonlinear quantum mlp architecture using parameterized two-qubit gates. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*. 1–6.

[38] Bob Coecke and Ross Duncan. 2011. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics* 13, 4 (2011).

[39] Sara Cohen, Werner Nutt, and Yehoshua Sagiv. 2007. Deciding equivalences among conjunctive aggregate queries. *Journal of the ACM (JACM)* 54, 2 (2007), 50 pages.

[40] Tim Coopmans, Robert Knegjens, Axel Dahlberg, David Maier, Loek Nijsten, Julio de Oliveira Filho, Martijn Papendrecht, Julian Rabbie, Filip Rozpędek, Matthew Skrzypczyk, et al. 2021. Netsquid, a network simulator for quantum information using discrete events. *Communications Physics* 4, 1 (2021), 164.

[41] John A Cortese and Timothy M Braje. 2018. Loading classical data into a quantum computer. *arXiv:1803.01958* (2018).

[42] Diogo Cruz, Romain Fournier, Fabien Gremion, Alix Jeannerot, Kenichi Komagata, Tara Tosic, Jarla Thiesbrummel, Chun Lam Chan, Nicolas Macris, Marc-André Dupertuis, et al. 2019. Efficient quantum algorithms for GHZ and W states, and implementation on the IBM quantum computer. *Advanced Quantum Technologies* 2, 5-6 (2019), 1900015.

[43] Raúl de la Cruz, Hadrien Calmet, and Guillaume Houzeaux. 2012. Implementing a XDMF/HDF5 Parallel File System in Alya. (Dec 2012). https://doi.org/10.5281/zenodo.6241986

[44] NVIDIA cuQuantum team. 2024. NVIDIA cuQuantum. https://docs.nvidia.com/cuda/cuquantum/22.07.1/cutensornet/overview.html. Accessed: 2024-08-14.

[45] Axel Dahlberg, Matthew Skrzypczyk, Tim Coopmans, Leon Wubben, Filip Rozpędek, Matteo Pompili, Arian Stolk, Przemysław Pawełczak, Robert Knegjens, Julio de Oliveira Filho, et al. 2019. A link layer protocol for quantum networks. In *Proceedings of the ACM special interest group on data communication (SIGCOMM)*. 159–173.

[46] Axel Dahlberg and Stephanie Wehner. 2018. SimulaQron—a simulator for developing quantum internet software. *Quantum Science and Technology* 4, 1 (2018), 015001.

[47] Adnan Darwiche and Pierre Marquis. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research (JAIR)* 17 (2002), 229–264.

[48] Niel de Beaudrap and Dominic Horsman. 2020. The ZX calculus is a language for surface code lattice surgery. *Quantum* 4 (2020), 218.

[49] Nathalie P De Leon, Kohei M Itoh, Dohun Kim, Karan K Mehta, Tracy E Northup, Hanhee Paik, BS Palmer, Nitin Samarth, Sorawis Sangtawesin, and David W Steuerman. 2021. Materials challenges and opportunities for quantum computing hardware. *Science* 372, 6539 (2021), eabb2823.

[50] Antonio deMarti iOlius, Patricio Fuentes, Román Orús, Pedro M. Crespo, and Josu Etxezarreta Martinez. 2024. Decoding algorithms for surface codes. (2024). arXiv:2307.14989 [quant-ph]

[51] Haowen Dong, Chao Zhang, Guoliang Li, and Huanchen Zhang. 2024. Cloud-Native Databases: A Survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2024), 1–20.

[52] Albert Einstein, Boris Podolsky, and Nathan Rosen. 1935. Can quantum-mechanical description of physical reality be considered complete? *Physical review* 47, 10 (1935), 777.

[53] David Elkouss, Jesus Martinez-mateo, and Vicente Martin. 2011. Information reconciliation for quantum key distribution. *Quantum Information & Computation* 11, 3 (2011), 226–238.

[54] Hongxiang Fan, Ce Guo, and Wayne Luk. 2022. Optimizing quantum circuit placement via machine learning. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 19–24.

[55] Tobias Fankhauser, Marc E Solèr, Rudolf M Füchslin, and Kurt Stockinger. 2021. Multiple query optimization using a hybrid approach of classical and quantum computing. (2021). arXiv:2107.10508 [cs.DB]

[56] Tobias Fankhauser, Marc E Soler, Rudolf M Füchslin, and Kurt Stockinger. 2023. Multiple Query Optimization using a Gate-Based Quantum Computer. *IEEE Access* (2023).

[57] Paolo Fittipaldi, Anastasios Giovanidis, and Frédéric Grosshans. 2023. A linear algebraic framework for dynamic scheduling over memory-equipped quantum networks. *IEEE Transactions on Quantum Engineering (TQE)* (2023).

[58] Austin G Fowler. 2015. Minimum weight perfect matching of fault-tolerant topological quantum error correction in average O (1) parallel time. *Quantum Information & Computation* 15, 1-2 (2015), 145–158.

[59] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. 2012. Surface codes: Towards practical large-scale quantum computation. *Physical Review A* 86, 3 (2012), 032324.

[60] Austin G Fowler, Ashley M Stephens, and Peter Groszkowski. 2009. High-threshold universal quantum computation on the surface code. *Physical Review A—Atomic, Molecular, and Optical Physics* 80, 5 (2009), 052312.

[61] Kristin Fritsch and Stefanie Scherzinger. 2023. Solving Hard Variants of Database Schema Matching on Quantum Computers. *Proceedings of the VLDB Endowment (PVLDB)* 16, 12 (2023), 3990–3993.

[62] Thomas Fösel, Murphy Yuezhen Niu, Florian Marquardt, and Li Li. 2021. Quantum circuit optimization with deep reinforcement learning. arXiv:2103.07585 [quant-ph] https://arxiv.org/abs/2103.07585

[63] Zekai J. Gao, Shangyu Luo, Luis Leopoldo Perez, and Chris Jermaine. 2017. The BUDS Language for Distributed Bayesian Machine Learning. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. ACM, 961–976.

[64] Scarlett Gauthier, Gayane Vardoyan, and Stephanie Wehner. 2023. A Control Architecture for Entanglement Generation Switches in Quantum Networks. In *Proceedings of the 1st Workshop on Quantum Networks and Distributed Quantum Computing*. 38–44.

[65] Floris Geerts, Thomas Muñoz, Cristian Riveros, Jan Van den Bussche, and Domagoj Vrgoč. 2021. Matrix query languages. *ACM SIGMOD Record* 50, 3 (2021), 6–19.

[66] Iulia M Georgescu, Sahel Ashhab, and Franco Nori. 2014. Quantum simulation. *Reviews of Modern Physics* 86, 1 (2014), 153–185.

[67] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008. Quantum random access memory. *Physical Review Letters* 100, 16 (2008), 160501.

[68] Daniel Gottesman. 1998. The Heisenberg Representation of Quantum Computers. (1998). arXiv:9807006 [quant-ph]

[69] Daniel Eric Gottesman. 1997. *Stabilizer Codes and Quantum Error Correction*. Ph.D. Dissertation. California Institute of Technology.

[70] Sven Groppe and Jinghua Groppe. 2021. Optimizing Transaction Schedules on Universal Quantum Computers via Code Generation for Grover's Search Algorithm. In *Proceedings of the 25th International Database Engineering & Applications Symposium (IDEAS)*. 149–156.

[71] Le Gruenwald, Tobias Winker, Umut Çalikyilmaz, Jinghua Groppe, and Sven Groppe. 2023. Index Tuning with Machine Learning on Quantum Computers for Large-Scale Database Applications.. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB) (CEUR Workshop Proceedings)*, Vol. 3462.

[72] Rihan Hai, Shih-Han Hung, and Sebastian Feld. 2024. Quantum Data Management: From Theory to Opportunities. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 5376–5381.

[73] Rihan Hai, Christos Koutras, Christoph Quix, and Matthias Jarke. 2023. Data lakes: A survey of functions and systems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 35, 12 (2023), 12571–12590.

[74] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. 2009. Quantum algorithm for linear systems of equations. *Physical Review Letters* 103, 15 (2009), 150502.

[75] Dong He, Supun Chathuranga Nakandala, Dalitso Banda, Rathijit Sen, Karla Saur, Kwanghyun Park, Carlo Curino, Jesús Camacho-Rodríguez, Konstantinos Karanasos, and Matteo Interlandi. 2022. Query Processing on Tensor Computation Runtimes. *Proceedings of the VLDB Endowment (PVLDB)* 15, 11 (2022), 2811–2825.

[76] Bettina Heim, Mathias Soeken, Sarah Marshall, Chris Granade, Martin Roetteler, Alan Geller, Matthias Troyer, and Krysta Svore. 2020. Quantum programming languages. *Nature Reviews Physics* 2, 12 (2020), 709–722.

[77] Oscar Higgott and Nikolas P. Breuckmann. 2021. Subsystem Codes with High Thresholds by Gauge Fixing and Reduced Qubit Overhead. *Physical Review X* 11 (Aug 2021), 031039. Issue 3.

[78] Cupjin Huang, Fang Zhang, Michael Newman, Junjie Cai, Xun Gao, Zhengxiong Tian, Junyin Wu, Haihong Xu, Huanjun Yu, Bo Yuan, et al. 2020. Classical simulation of quantum supremacy circuits. (2020). arXiv:2005.06787 [quant-ph]

[79] Cupjin Huang, Fang Zhang, Michael Newman, Xiaotong Ni, Dawei Ding, Junjie Cai, Xun Gao, Tenghui Wang, Feng Wu, Gengyan Zhang, et al. 2021. Efficient parallelization of tensor network contraction for simulating quantum computation. *Nature Computational Science* 1, 9 (2021), 578–587.

[80] Hsin-Yuan Huang, Michael Broughton, Jordan Cotler, Sitan Chen, Jerry Li, Masoud Mohseni, Hartmut Neven, Ryan Babbush, Richard Kueng, John Preskill, et al. 2022. Quantum advantage in learning from experiments. *Science* 376, 6598 (2022), 1182–1186.

[81] Zezhou Huang, Rathijit Sen, Jiaxiang Liu, and Eugene Wu. 2023. JoinBoost: Grow Trees Over Normalized Data Using Only SQL. *Proceedings of the VLDB Endowment (PVLDB)* 16, 11 (2023), 3071–3084.

[82] Claudius Hubig, Ian P. McCulloch, and Ulrich Schollwöck. 2017. Generic construction of efficient matrix product operators. *Physical Review B* 95 (Jan 2017), 035129. Issue 3.

[83] Paras Jain, Peter Kraft, Conor Power, Tathagata Das, Ion Stoica, and Matei Zaharia. 2023. Analyzing and Comparing Lakehouse Storage Systems. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*.

[84] Zhihao Jia, Oded Padon, James Thomas, Todd Warszawski, Matei Zaharia, and Alex Aiken. 2019. TASO: optimizing deep learning computation with automatic generation of graph substitutions. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP)*. 47–62.

[85] Tyson Jones, Anna Brown, Ian Bush, and Simon C Benjamin. 2019. QuEST and high performance simulation of quantum computers. *Scientific reports* 9, 1 (2019), 10736.

[86] Bjarni Jónsson, Bela Bauer, and Giuseppe Carleo. 2018. Neural-network states for the classical simulation of quantum computing. (2018). arXiv:1808.05232 [quant-ph]

[87] Richard Jozsa and Akimasa Miyake. 2008. Matchgates and classical simulation of quantum circuits. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 464, 2100 (2008), 3089–3106.

[88] Konstantinos Karanasos, Matteo Interlandi, Doris Xin, Fotis Psallidas, Rathijit Sen, Kwanghyun Park, Ivan Popivanov, Supun Nakandal, Subru Krishnan, Markus Weimer, et al. 2020. Extending Relational Query Processing with ML Inference. *Proceedings of the Conference on Innovative Data Systems Research (CIDR)* (2020).

[89] Manish Kesarwani and Jayant R. Haritsa. 2024. Index Advisors on Quantum Platforms. *Proceedings of the VLDB Endowment (PVLDB)* 17, 11 (2024), 3615–3628.

[90] Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2018. In-Database Learning with Sparse Tensors. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. 325–340.

[91] Mahmoud Abo Khamis, Hung Q. Ngo, Xuanlong Nguyen, Dan Olteanu, and Maximilian Schleich. 2020. Learning models over relational data using sparse tensors and functional dependencies. *ACM Transactions on Database Systems (TODS)* 45, 2 (2020).

[92] Aleks Kissinger and John van de Wetering. 2020. Reducing the number of non-Clifford gates in quantum circuits. *Physical Review A* 102, 2 (2020), 022406.

[93] Fredrik Kjolstad, Shoaib Kamil, Stephen Chou, David Lugato, and Saman Amarasinghe. 2017. The tensor algebra compiler. *Proceedings of the ACM on Programming Languages (PACMPL)* 1, OOPSLA (2017), 1–29.

[94] Phokion G. Kolaitis and Moshe Y. Vardi. 1998. Conjunctive-query containment and constraint satisfaction. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. 205–213.

[95] Dimitrios Koutsoukos, Supun Nakandala, Konstantinos Karanasos, Karla Saur, Gustavo Alonso, and Matteo Interlandi. 2021. Tensors: An abstraction for general data processing. *Proceedings of the VLDB Endowment (PVLDB)* 14, 10 (2021), 1797–1804.

[96] Laurens Kuiper, Peter Boncz, and Hannes Mühleisen. 2024. Robust External Hash Aggregation in the Solid State Age. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 3753–3766. https://doi.org/10.1109/ICDE60146.2024.00288

[97] Laurens Kuiper and Hannes Mühleisen. 2023. These Rows Are Made for Sorting and That's Just What We'll Do. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. 2050–2062.

[98] Guglielmo Lami and Mario Collura. 2024. Unveiling the Stabilizer Group of a Matrix Product State. *Physical Review Letters* 133 (Jul 2024), 010602. Issue 1. https://doi.org/10.1103/PhysRevLett.133.010602

[99] Moritz Lange, Pontus Havström, Basudha Srivastava, Valdemar Bergentall, Karl Hammar, Olivia Heuts, Evert van Nieuwenburg, and Mats Granath. 2023. Data-driven decoding of quantum error correcting codes using graph neural networks. arXiv:2307.01241 [quant-ph] https://arxiv.org/abs/2307.01241

[100] Adrian Lehmann, Ben Caldwell, and Robert Rand. 2022. VyZX: a vision for verifying the ZX calculus. *arXiv preprint arXiv:2205.05781* (2022).

[101] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proceedings of the VLDB Endowment (PVLDB)* 9, 3 (2015), 204–215.

[102] Riling Li, Bujiao Wu, Mingsheng Ying, Xiaoming Sun, and Guangwen Yang. 2020. Quantum Supremacy Circuit Simulation on Sunway TaihuLight. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 31, 4 (2020), 805–816.

[103] Victoria Lipinska, Gláucia Murta, and Stephanie Wehner. 2018. Anonymous transmission in a noisy quantum network using the W state. *Physical Review A* 98, 5 (2018), 052320.

[104] Xiao-Yang Liu and Zeliang Zhang. 2023. Classical simulation of quantum circuits using reinforcement learning: Parallel environments and benchmark. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS)*. 67082–67102.

[105] Shangyu Luo, Zekai J. Gao, Michael N. Gubanov, Luis Leopoldo Perez, and Christopher M. Jermaine. 2017. Scalable Linear Algebra on a Relational Database System. In *33rd IEEE International Conference on Data Engineering (ICDE)*. 523–534.

[106] Xiu-Zhe Luo, Jin-Guo Liu, Pan Zhang, and Lei Wang. 2020. Yao. jl: Extensible, efficient framework for quantum algorithm design. *Quantum* 4 (2020), 341.

[107] Nantia Makrynioti and Vasilis Vassalos. 2019. Declarative data analytics: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 33, 6 (2019), 2392–2411.

[108] Ioana Manolescu and Madhulika Mohanty. 2023. Full-Power Graph Querying: State of the Art and Challenges. *Proceedings of the VLDB Endowment (PVLDB)* 16, 12 (2023), 3886–3889.

[109] Rui Mao, Guojing Tian, and Xiaoming Sun. 2024. Toward optimal circuit size for sparse quantum state preparation. *Phys. Rev. A* 110 (Sep 2024), 032439. Issue 3. https://doi.org/10.1103/PhysRevA.110.032439

[110] Satvik Maurya and Swamit Tannu. 2024. Managing Classical Processing Requirements for Quantum Error Correction. *arXiv:2406.17995* (2024).

[111] Andrew McGregor. 2014. Graph stream algorithms: a survey. *ACM SIGMOD Record* 43, 1 (may 2014), 9–20.

[112] Matija Medvidović and Giuseppe Carleo. 2021. Classical variational simulation of the quantum approximate optimization algorithm. *npj Quantum Information* 7, 1 (2021), 101.

[113] Eli Meirom, Haggai Maron, Shie Mannor, and Gal Chechik. 2022. Optimizing Tensor Network Contraction Using Reinforcement Learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research)*, Vol. 162. PMLR, 15278–15292. https://proceedings.mlr.press/v162/meirom22a.html

[114] R. Van Meter, T. D. Ladd, W. J. Munro, and K. Nemoto. 2009. System Design for a Long-Line Quantum Repeater. *IEEE/ACM Transactions on Networking (TON)* 17, 3 (2009), 1002–1013.

[115] Jorge Miguel-Ramiro and Wolfgang Dür. 2020. Delocalized information in quantum networks. *New Journal of Physics* 22, 4 (2020), 043011.

[116] Jorge Miguel-Ramiro, Ferran Riera-Sàbat, and Wolfgang Dür. 2023. Quantum repeater for W states. *PRX Quantum* 4, 4 (2023), 040323.

[117] Supun Nakandala, Karla Saur, Gyeong-In Yu, Konstantinos Karanasos, Carlo Curino, Markus Weimer, and Matteo Interlandi. 2020. A Tensor Compiler for Unified Machine Learning Prediction Serving. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 899–917.

[118] Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. 2019. Data lake management: challenges and opportunities. *Proceedings of the VLDB Endowment (PVLDB)* 12, 12 (2019), 1986–1989.

[119] Hendrik Poulsen Nautrup, Nicolas Delfosse, Vedran Dunjko, Hans J Briegel, and Nicolai Friis. 2019. Optimizing quantum error correction codes with reinforcement learning. *Quantum* 3 (2019), 215.

[120] Nitin Nayak, Jan Rehfeld, Tobias Winker, Benjamin Warnke, Umut Çalikyilmaz, and Sven Groppe. 2023. Constructing Optimal Bushy Join Trees by Solving QUBO Problems on Quantum Hardware and Simulators. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments (BiDEDE '23)*. 1–7.

[121] Thomas Neumann. 2024. Closing the Gap between Theory and Practice in Query Optimization. In *Companion of the 43rd Symposium on Principles of Database Systems (PODS)*. ACM, 4.

[122] HT Ng and K Kim. 2014. Quantum estimation of magnetic-field gradient using W-state. *Optics Communications* 331 (2014), 353–358.

[123] Naomi H Nickerson, Joseph F Fitzsimons, and Simon C Benjamin. 2014. Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links. *Physical Review X* 4, 4 (2014), 041041.

[124] Michael A Nielsen and Isaac L Chuang. 2010. *Quantum computation and quantum information*. Cambridge university press.

[125] Milos Nikolic and Dan Olteanu. 2018. Incremental View Maintenance with Triple Lock Factorization Benefits. In *Proceedings of the 2018 ACM International Conference on Management of Data (SIGMOD)*. ACM, 365–380.

[126] Román Orús. 2014. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of physics* 349 (2014), 117–158.

[127] Román Orús. 2019. Tensor networks for complex quantum systems. *Nature Reviews Physics* 1, 9 (2019), 538–550.

[128] Román Orús. 2014. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics* 349 (2014), 117–158. https://doi.org/10.1016/j.aop.2014.06.013

[129] M Tamer Özsu, Patrick Valduriez, et al. 1999. *Principles of distributed database systems*. Vol. 2. Springer.

[130] Matteo Paganelli, Paolo Sottovia, Kwanghyun Park, Matteo Interlandi, and Francesco Guerra. 2023. Pushing ML Predictions Into DBMSs. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 35, 10 (2023), 10295–10308.

[131] Alexandru Paler, Lucian Sasu, Adrian-Cătălin Florea, and Răzvan Andonie. 2023. Machine learning optimization of quantum circuit layouts. *ACM Transactions on Quantum Computing* 4, 2 (2023), 1–25.

[132] Feng Pan and Pan Zhang. 2021. Simulating the Sycamore quantum supremacy circuits. (2021). arXiv:2103.03074 [quant-ph]

[133] Kwanghyun Park, Karla Saur, Dalitso Banda, Rathijit Sen, Matteo Interlandi, and Konstantinos Karanasos. 2022. End-to-end Optimization of Machine Learning Prediction Queries. In *Proceedings of the 2022 ACM International Conference on Management of Data (SIGMOD)*. 587–601.

[134] David Pérez-Garcia, Frank Verstraete, Michael M. Wolf, and J. Ignacio Cirac. 2007. Matrix product state representations. *Quantum Information & Computation* 7, 5 (2007), 401–430.

[135] Stefano Pirandola, Ulrik L Andersen, Leonardo Banchi, Mario Berta, Darius Bunandar, Roger Colbeck, Dirk Englund, Tobias Gehring, Cosmo Lupo, Carlo Ottaviani, et al. 2020. Advances in quantum cryptography. *Advances in Optics and Photonics* 12, 4 (2020), 1012–1236.

[136] Gabriel Popkin. 2016. Quest for qubits. *Science* 354, 6316 (2016), 1090–1093.

[137] John Preskill. 1999. Lecture notes for Physics 219: Quantum computation. *Caltech Lecture Notes* 7 (1999), 1.

[138] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (Aug. 2018), 79.

[139] Haoyu Qi, Daniel J. Brod, Nicolás Quesada, and Raúl García-Patrón. 2020. Regimes of Classical Simulability for Noisy Gaussian Boson Sampling. *Physical Review Letters* 124 (2020), 100502. Issue 10.

[140] Arend-Jan Quist, Jingyi Mei, Tim Coopmans, and Alfons Laarman. 2024. Advancing Quantum Computing with Formal Methods. (2024). arXiv:2407.11675 [quant-ph]

[141] Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, Frédo Durand, and Saman Amarasinghe. 2013. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. *ACM SIGPLAN Notices* 48, 6 (2013), 519–530.

[142] Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. 2016. Learning Linear Regression Models over Factorized Joins. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. ACM, 3–18.

[143] Maximilian Schleich, Dan Olteanu, Mahmoud Abo Khamis, Hung Q. Ngo, and XuanLong Nguyen. 2019. A Layered Aggregate Engine for Analytics Workloads. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. ACM, 1642–1659.

[144] Maximilian Schleich, Amir Shaikhha, and Dan Suciu. 2023. Optimizing tensor programs on flexible storage. *Proceedings of the ACM on Management of Data (PACMMOD)* 1, 1 (2023), 1–27.

[145] Manuel Schönberger. 2022. Applicability of Quantum Computing on Database Query Optimization. In *Proceedings of the 2022 ACM International Conference on Management of Data (SIGMOD)*. 2512–2514.

[146] Manuel Schönberger, Stefanie Scherzinger, and Wolfgang Mauerer. 2023. Ready to leap (by co-design)? join order optimisation on quantum hardware. *Proceedings of the ACM on Management of Data (PACMMOD)* 1, 1 (2023), 1–27.

[147] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum-inspired digital annealing for join ordering. *Proceedings of the VLDB Endowment (PVLDB)* 17, 3 (2023), 511–524.

[148] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum Optimisation of General Join Trees. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB) (CEUR Workshop Proceedings)*, Vol. 3462.

[149] Bao-Sen Shi and Akihisa Tomita. 2002. Teleportation of an unknown state by W state. *Physics Letters A* 296, 4-5 (2002), 161–164.

[150] Peter W Shor. 1995. Scheme for reducing decoherence in quantum computer memory. *Physical review A* 52, 4 (1995), R2493.

[151] Matthew Skrzypczyk and Stephanie Wehner. 2021. An architecture for meeting quality-of-service requirements in multi-user quantum networks. (2021). arXiv:2111.13124 [quant-ph]

[152] Mikhail Smelyanskiy, Nicolas P. D. Sawaya, and Alán Aspuru-Guzik. 2016. qHiPSTER: The Quantum High Performance Software Testing Environment. (2016). arXiv:1601.07195 [quant-ph]

[153] Robert S Smith, Eric C Peterson, Mark G Skilbeck, and Erik J Davis. 2020. An open-source, industrial-strength optimizing compiler for quantum programs. *Quantum Science and Technology* 5, 4 (2020), 044001.

[154] Redgate Software. 2024. DB-Engines Ranking. https://db-engines.com/en/ranking Accessed: 2024-06-30.

[155] Andrew M. Steane. 1996. Error Correcting Codes in Quantum Theory. *Physical Review Letters* 77 (Jul 1996), 793–797. Issue 5.

[156] Michael Steinbrunn, Guido Moerkotte, and Alfons Kemper. 1997. Heuristic and randomized optimization for the join ordering problem. *The VLDB journal* 6 (1997), 191–208.

[157] Michael Stonebraker and Andrew Pavlo. 2024. What Goes Around Comes Around... And Around... *ACM SIGMOD Record* 53, 2 (2024), 21–37.

[158] Christoph Sünderhauf, Earl Campbell, and Joan Camps. 2024. Block-encoding structured matrices for data input in quantum computing. *Quantum* 8 (Jan. 2024), 1226. https://doi.org/10.22331/q-2024-01-11-1226

[159] Barbara M. Terhal. 2015. Quantum error correction for quantum memories. *Reviews of Modern Physics* 87 (Apr 2015), 307–346. Issue 2.

[160] Yuanyuan Tian. 2023. The world of graph databases from an industry perspective. *ACM SIGMOD Record* 51, 4 (2023), 60–67.

[161] Tim Bittner and Sven Groppe. 2020. Hardware Accelerating the Optimization of Transaction Schedules via Quantum Annealing by Avoiding Blocking. *Open Journal of Cloud Computing (OJCC)* 7, 1 (2020), 1–21.

[162] Immanuel Trummer. 2024. Towards Out-of-Core Simulators for Quantum Computing. In *Proceedings of the 1st Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications (Q-Data '24)*.

[163] Immanuel Trummer and Christoph Koch. 2016. Multiple Query Optimization on the D-Wave 2X Adiabatic Quantum Computer. *Proceedings of the VLDB Endowment (PVLDB)* 9, 9 (2016).

[164] Immanuel Trummer and Christoph Koch. 2017. Solving the join ordering problem via mixed integer linear programming. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1025–1040.

[165] William G Unruh. 1995. Maintaining coherence in quantum computers. *Physical Review A* 51, 2 (1995), 992.

[166] Marzio Vallero, Flavio Vella, and Paolo Rech. 2024. State of practice: evaluating GPU performance of state vector and tensor network methods. arXiv:2401.06188 [quant-ph] https://arxiv.org/abs/2401.06188

[167] John van de Wetering. 2020. ZX-calculus for the working quantum computer scientist. (2020). arXiv:2012.13966 [quant-ph]

[168] John van de Wetering. 2020. ZX-calculus for the working quantum computer scientist. *arXiv preprint arXiv:2012.13966* (2020).

[169] Maarten Van Den Nes. 2010. Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond. *Quantum Information & Computation* 10, 3 (2010), 258–271.

[170] Guifré Vidal. 2003. Efficient classical simulation of slightly entangled quantum computations. *Physical Review Letters* 91, 14 (2003), 147902.

[171] Benjamin Villalonga, Sergio Boixo, Bron Nelson, Christopher Henze, Eleanor Rieffel, Rupak Biswas, and Salvatore Mandrà. 2019. A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware. *npj Quantum Information* 5, 1 (2019), 86.

[172] Trevor Vincent, Lee J O'Riordan, Mikhail Andrenkov, Jack Brown, Nathan Killoran, Haoyu Qi, and Ish Dhand. 2022. Jet: Fast quantum circuit simulations with parallel task-based tensor-network contraction. *Quantum* 6 (2022), 709.

[173] Lieuwe Vinkhuijzen, Tim Coopmans, David Elkouss, Vedran Dunjko, and Alfons Laarman. 2023. LIMDD: A decision diagram for simulation of quantum computing including stabilizer states. *Quantum* 7 (2023), 1108.

[174] Lieuwe Vinkhuijzen, Tim Coopmans, and Alfons Laarman. 2024. A Knowledge Compilation Map for Quantum Information. arXiv:2401.01322 [quant-ph] https://arxiv.org/abs/2401.01322

[175] Suhas Vittal, Poulami Das, and Moinuddin Qureshi. 2023. Astrea: Accurate Quantum Error-Decoding via Practical Minimum-Weight Perfect-Matching. In *Proceedings of the 50th Annual International Symposium on Computer Architecture* (Orlando, FL, USA) *(ISCA '23)*. Association for Computing Machinery, New York, NY, USA, Article 2, 16 pages. https://doi.org/10.1145/3579371.3589037

[176] Ye Wang, Qing Wang, Henning Koehler, and Yu Lin. 2021. Query-by-Sketch: Scaling Shortest Path Graph Queries on Very Large Networks. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD)*. 1946–1958.

[177] Stephanie Wehner, David Elkouss, and Ronald Hanson. 2018. Quantum internet: A vision for the road ahead. *Science* 362, 6412 (2018), eaam9288.

[178] Sam Westrick, Pengyu Liu, Byeongjee Kang, Colin McDonald, Mike Rainey, Mingkuan Xu, Jatin Arora, Yongshan Ding, and Umut A. Acar. 2024. GraFeyn: Efficient Parallel Sparse Simulation of Quantum Circuits. In *Proceedings of the IEEE International Conference on Quantum Computing and Engineering*.

[179] Tobias Winker, Umut Çalikyilmaz, Le Gruenwald, and Sven Groppe. 2023. Quantum machine learning for join order optimization using variational quantum circuits. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments (BiDEDE '23)*.

[180] Tobias Winker, Sven Groppe, Valter Uotila, Zhengtong Yan, Jiaheng Lu, Maja Franz, and Wolfgang Mauerer. 2023. Quantum machine learning: Foundation, new techniques, and opportunities for database research. In *Companion of the 2023 International Conference on Management of Data (SIGMOD)*. 45–52.

[181] Xiaoliang Wu, Alexander Kolar, Joaquin Chung, Dong Jin, Tian Zhong, Rajkumar Kettimuthu, and Martin Suchara. 2021. SeQUeNCe: a customizable discrete-event simulator of quantum networks. *Quantum Science and Technology* 6, 4 (2021), 045027.

[182] Xin-Chuan Wu, Sheng Di, Emma Maitreyee Dasgupta, Franck Cappello, Hal Finkel, Yuri Alexeev, and Frederic T Chong. 2019. Full-state quantum circuit simulation by using data compression. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 1–24.

[183] Yue Wu, Namitha Liyanage, and Lin Zhong. 2022. An interpretation of Union-Find Decoder on Weighted Graphs. (2022). arXiv:2211.03288 [quant-ph]

[184] Xiaosi Xu, Simon Benjamin, Jinzhao Sun, Xiao Yuan, and Pan Zhang. 2023. A Herculean task: Classical simulation of quantum computers. (2023). arXiv:2302.08880 [quant-ph]

[185] Xifeng Yan, Philip S Yu, and Jiawei Han. 2005. Substructure similarity search in graph databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. 766–777.

[186] Kieran Young, Marcus Scese, and Ali Ebnenasir. 2023. Simulating Quantum Computations on Classical Machines: A Survey. (2023). arXiv:2311.16505 [quant-ph]

[187] Gongsheng Yuan, Yuxing Chen, Jiaheng Lu, Sai Wu, Zhiwei Ye, Ling Qian, and Gang Chen. 2024. Quantum Computing for Databases: Overview and Challenges. (2024). arXiv:2405.12511 [cs.DB]

[188] Gongsheng Yuan, Jiaheng Lu, Yuxing Chen, Sai Wu, Chang Yao, Zhengtong Yan, Tuodu Li, and Gang Chen. 2023. Quantum Computing for Databases: A Short Survey and Vision. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB) (CEUR Workshop Proceedings)*, Vol. 3462.

[189] Yexiong Zeng, Zheng-Yang Zhou, Enrico Rinaldi, Clemens Gneiting, and Franco Nori. 2023. Approximate Autonomous Quantum Error Correction with Reinforcement Learning. *Physical Review Letters* 131 (Jul 2023), 050601. Issue 5. https://doi.org/10.1103/PhysRevLett.131.050601

[190] Xuanhe Zhou, Chengliang Chai, Guoliang Li, and Ji Sun. 2020. Database meets artificial intelligence: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2020).

[191] Yiqing Zhou, E Miles Stoudenmire, and Xavier Waintal. 2020. What limits the simulation of quantum computers? *Physical Review X* 10, 4 (2020), 041038.

[192] Alwin Zulehner, Alexandru Paler, and Robert Wille. 2018. An efficient methodology for mapping quantum circuits to the IBM QX architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 38, 7 (2018), 1226–1236.

[193] Alwin Zulehner and Robert Wille. 2018. Advanced simulation of quantum computations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 38, 5 (2018), 848–859.

[Shih-Han#1: There are quite a few compilation errors. I tried to fix but didn't succeed. Is it possible to remove them?]

Appendix A DATA REPRESENTATION FOR SIMULATION

[Rihan#1: Final check: appendix A includes necessary references]

We here describe several ways to represent a quantum state. We also explain how to simulate a quantum circuit which outputs some desired quantum state; the circuit thus consists of a canonical input state (defined below), followed by the application of gates (unitary matrices). See Fig. 4 for examples for the main ones mentioned in this work.

A.1 State vector

The canonical representation of an n-qubit quantum state is the *state vector*: a length-2^n vector $\vec{v}$ of complex entries v_j satisfying the normalisation constraint $\sum_{j=0}^{2^n-1} |v_j|^2 = 1$, where $|.|$ denotes the norm or modulus of a complex number [124]. See Fig. 4(a). The number v_j is called the amplitude corresponding to the index $j \in \{0, 1, \ldots, 2^n - 1\}$. The index j can be written as length-n bitstring, and we say that the quantum state $\vec{v}$ denotes an amplitude distribution over the length-n bitstrings. We can also interpret a bitstring as an address, where the k-th bit 0/1 recursively indicates chosing the top/bottom half of a length-2^k vector for some $k \in \{0, 1, \ldots, 2^n - 1\}$ (see Fig. 4(a1) for examples). Each qubit added thus adds a bit to the address, consequently doubling the size of the state vector.

A quantum circuit on n-qubits starts with as canonical input state the length-2^n vector whose topmost entry is 1, and the remainder entries are 0. Simulating a sequence of gates is then done by iteratively updating the quantum-state vector by multiplying it with the next gate (unitary matrix) in line. The computational complexity of a single gate update is thus the same as multiplying an $N \times N$ matrix with a length-N vector, where $N = 2^n$. This is generally $O(N^2)$**[Tim#1: can be improved using the matrix-multiplication exponent $\omega \approx 2.37\ldots$?]**.

A.2 Order-n tensor

An alternative, equivalent representation of the quantum-state vector is found by reordering the 2^n entries in an n-dimensional tensor, where every dimension is 2. See Fig. 4(b) for an example.

Evaluating a quantum gate on the order-n tensor is done by performing *tensor contraction* between the corresponding unitary matrix (an order-2 tensor). Tensor contraction, a generalization of matrix-matrix multiplication, is most easily explained by graphically visualizing a tensor as a node, with an edge attached for each order labelled with the corresponding dimension, see Fig. 4(a2, b2).

Two edges with the same dimension can be connected; contracting the edges is then defined as follows. Suppose we are given a 3-tensor A of dimensions $2 \times 3 \times 4$ and a 4-tensor B of dimensions $3 \times 6 \times 7 \times 8$ and an edge between these which indicates the second index of A and the first index of B. Then contracting the edge is an operation which replaces the two nodes by a single one, which represents a 5-tensor C of dimensions $2 \times 4 \times 6 \times 7 \times 8$, defined as $C_{ijklm} = \sum_x A_{ixj} \cdot B_{xklm}$. The naive computational complexity of tensor contraction is $O(X \cdot d)$ where X is the product of the dimensions of the output tensor and d the dimension over which the contraction is performed. **[Tim#2: but should be able to improve**

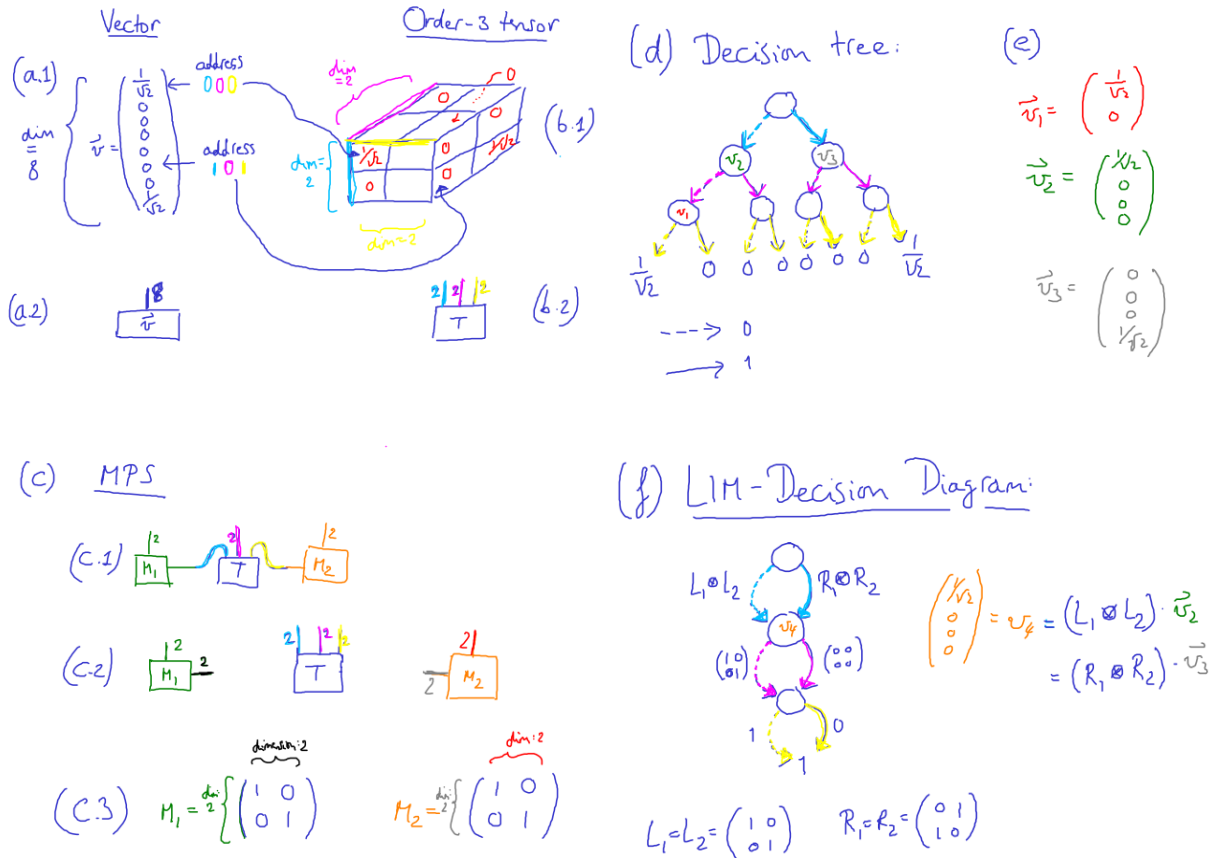


Figure 4: Various data structures representing the 3-qubit GHZ state. (a1) As vector of $2^3 = 8$ entries. Each entry is labelled a 3-bit address. The 8 entries can also be stacked differently as is done in the order-3 tensor (b1). The order-3 tensor has dimensions $2 \times 2 \times 2$, yielding precisely the $2^3 = 8$ different addresses. (b2) graphically depicts T as a node labelled with the T, while three outgoing edges (called 'legs') indicate the dimensions of the tensor. The length-8 vector from (a1) is a single-dimensional tensor, hence its depiction in (a2). Alternatively to vectors/order-n-tensors, (c) the Matrix Product Formalism writes the $n = 3$-qubit GHZ states as a collection of $n = 3$ tensors M_1, M_2 (c3) and T (b1) (depicted graphically in c2, b2), some of whose legs are connected (c1). Converting back to the vector or order-n-tensor representation is done by *contracting* the connected legs (a generalization of matrix multiplication; see main text for definition). Tim: TODO specify where in main text (d) depicts the vector entries from (a) as the leaf nodes of a decision tree where the decisions are indicated by the n address bits (coloring consistent with a,b). Each node represents a vector (e), with the root node depicting the original 3-qubit GHZ-state vector from (a). A decision diagram compresses the decision tree by realising that not all of the 2^n nodes in the decision tree need to be stored separately. The Local-Invertible Map (LIM) Decision Diagram in particular (f) merges nodes since the vectors they represent can be mapped to each other using Kronecker products (denoted $\otimes$) of 2×2 invertible matrices. TODO: draw figure in proper software

this using Strassen; reference?] For multiplying an $a \times b$ matrix with a $b \times c$ matrix, this complexity reduces to the well-known $O((a \cdot c) \cdot b)$.

For the order-n tensor, the update after a gate using tensor contraction is equivalent to the matrix-vector gate update for the state vector representation in the sense that one performs exactly the same multiplication and addition of the individual amplitudes. The gate update also has the same computational complexity.

A.3 MPS

In the Matrix-Product State (MPS) formalism, the order-n tensor (or, equivalently, the length-2^n vector), is decomposed as product of matrices. Formally, an MPS representing an n-qubit state is equivalently defined as a collection of $2n$ matrices, i.e. order-2-tensors, $M_1^0, M_1^1, M_2^0, \ldots, M_n^0, M_n^1$, such that the entry of the quantum state vector at binary address $x_1 x_2 \ldots x_n \in \{0,1\}^n$ is found as the matrix product $\prod_{j=1}^{n} M_j^{x_j}$. The largest matrix dimension among the matrices M is called *bond dimension*.

Equivalently, we can write MPS as a sequence of order-2 and order-3 tensors, whose edge connection structure forms a line. See Fig. 4(c) for an example.

Due to the simple graph structure of MPS, specialized simulation algorithms have been found which are fixed-parameter tractable in

the bond dimension. MPS can be used for both exact simulation (i.e. where the represented output state vector is precisely or approximate simulation. Approximate simulation, i.e. where the resulting output state vector is close to the real output state vector, is achieved by approximating the individual matrices using the singular value decomposition.**[Tim#3: add what is gained duringa pproximating matrices: space, smaller bond dimension] [Tim#4: should still add citations] [Rihan#2: can we here explictly say MPS only has order-2 and order-3 tensors; also, how does contraction works for MPS? I worry now the time complexity for W/GHZ in MPS/MPO are not very easy to understand]**

A Matrix Product State is one of the simplest tensor networks: the graph is a line.

[inline]Tim should continue writing/polishing the writing from here on

A.4 Tensor network

A Matrix Product State is one of the simplest tensor networks: the graph is a line. **[Rihan#3: @Tim, first, can we address here Floris#3 and #4? To be consistent with B.1&2, we need here peole understand the number of legs is the same as the order of the tensor. We can move Fig.7 here. Also, from the texts here, people can more or less understand how contraction works, so they understand the time complexity analysis]**

Tensor network **[Tim#5: copy/rephrase shih-han text at start of 4.2 with definition of tensor network]** [18] is a graph where each node represents a tensor (multidimensional array), and where the edges represent contractions (summations) over shared indices. By mapping qubits and gates to tensors, general quantum circuits can be simulated using tensor networks. *Einstein summation* [9] is a compact notation for summing over repeated indices in tensor operations. For example, for matrices A and B, $A_{ik}B_{kj}$ represents the matrix multiplication $\sum_k A_{ik}B_{kj}$. **[Tim#6: I presume what we actually want to say here is 'tensor contraction'. Einstein notation is merely notation. I added the following text, let's take another pass]** Another example: suppose we are given a 3-tensor A of dimensions $2 \times 3 \times 4$ and a 4-tensor B of dimensions $3 \times 6 \times 7 \times 8$ and an edge between these which indicates the second index of A and the first index of B. Then contracting the edge is an operation which replaces the two nodes by a single one, which represents a 5-tensor C of dimensions $2 \times 4 \times 6 \times 7 \times 8$, defined as $C_{ijklm} = \sum_x A_{ixj} \cdot B_{xklm}$. By contracting all edges of a tensor network, the result is a single node. A tensor network represents an n-qubit quantum state if, after contracting all edges, the resulting node represents a quantum state vector, i.e. a 1-tensor D of dimension 2^n for some n satisfying $\sum_{x=1}^{2^n} |D_x|^2 = 1$. **[Tim#7: use name 'order-n tensor']**Alternatively, the 2^n entries of the vector are stacked differently as an n-tensor of dimensions $2 \times 2 \times \cdots \times 2$. **[Tim#8: Tim Littau used the second definition. Needed to go into this much detail? Probably needs picture for clarification]**

Simulating a quantum circuit using a tensor network can be done in two steps. First, the input state to the circuit is written as tensor network. The n-qubit input state is typically the 2^n-length unit vector $(1, 0, 0, \ldots, 0)$, which is known to be representable by an n-node tensor network. Next, as each gate is represented by a matrix, i.e. a 2-tensor, a node is added for each gate, followed by adding edges to represents the application of the gate onto the specific qubits. The resulting larger tensor network represents the quantum state outputted by the circuit. To obtain the quantum state as a vector, the tensor network is contracted. Computationally, this is the expensive operation. Finding an order of the edge to contract them, the *contraction order*, is an NP-hard problem for which various heuristics have been investigated. **[Tim#9: should still add citations]**

A.5 LIM-decision diagrams

Binary decsision diagrams constitute a well-developed technique for succinct representation and manipulation of functions mapping bits to bits. Various generalizations have been found which map bits to real or complex numbers, such as Algebraic Decision Diagrams (ADD), Edge-Valued Decision Diagrams or Quantum Multiple-Valued Decision Diagrams (QMDD), and Local-Invertible-Map Decision Diagram (LIM-Decision Diagram or LIMDD). These specifically apply to representing a 2^n-sized quantum state vector v by interpreting it as a function f_v which maps the bitstring $x_1x_2 \ldots x_n \in \{0,1\}^n$ to the vector's entry at index $x_1x_2 \ldots x_n$, i.e. an integer between 1 and 2^n written in base 2.

To briefly explain how ADDs, QMDDs and LIMDDs achieve a succinct representation of the state vector, let us first write the state vector function as a binary tree on n levels, where the 2^n leaf nodes contain precisely the 2^n vector entries**[Tim#10: todo add figure]**. In the binary tree, a node v at level $\ell \in \{1, 2, \ldots, n\}$ (where level 1 denotes the level just above the leaf nodes) represents a function $f_v : \{0,1\}^\ell \to \mathbb{C}$. We denote this as a *subfunction* of f_v. **[Tim#11: I would say function, but in app B you seem to call this a relational representation...]**

An ADD is obtained from the binary tree by merging leaf nodes with the same value and rerouting edges accordingly. Each node still corresponds to a single function. Next, a QMDD is obtained from an ADD by merging nodes v, v' if their subfunctions are identical modulo a constant factor, i.e. there exists $\lambda \in \mathbb{C}$ such that $f_v = \lambda \cdot f_{v'}$. The factor λ is used as edge label. This process is continued until no more nodes can be merged. A LIMDD is obtained by using a stronger merge rule: nodes are now merged if they represent functions which can be mapped to each other using *local invertible maps*, e.g. a quantum gate on each qubit.

LIMDDs are provably exponentially more succinct than QMDDs in the worst case, and never need more than polynomially more space (and so are the relations QMDD→ADD and ADD→ binary tree). The same holds for asymptotic runtime of the existing simulation algorithms. **[Tim#12: TODO add citations and figure]**

A.6 ZX-calculus

The ZX-calculus is a graphical language, inspired by tensor networks. In the ZX-calculus, a quantum circuit containing gates from some finite-sized elementary gate set is represented by a graph with a node for each gate; each node is labeled with the type of gate it represents. The ZX-calculus also contains a set of rewrite rules, which for example enable one to prove that circuits are equivalent. For quantum-circuit simulation specifically, the typical use is to ask for a specific entry in the quantum state vector that the ZX diagram represents. The entry is found by attaching a ZX diagram for the input state to the diagram for the circuit, followed by a sequence of

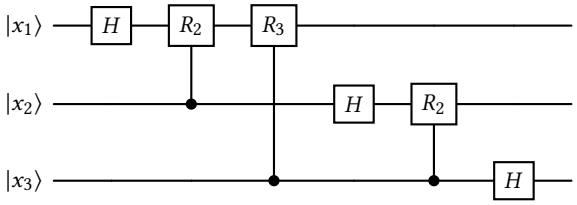


Figure 7: 3-qubit QFT circuit

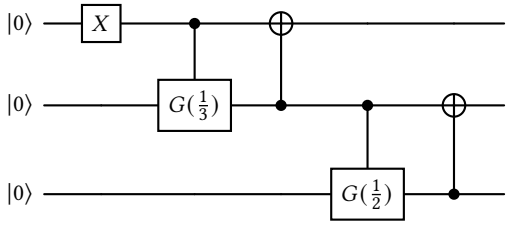


Figure 5: 3-qubit W state preparation circuit

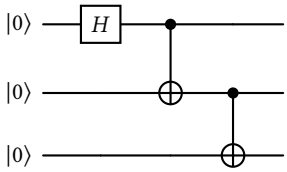


Figure 6: 3-qubit GHZ State preparation circuit

rewrite rules and a decomposition which expresses the entry as a linear combination of smaller diagrams which are easier to directly evaluate. **[Tim#13: todo add citations]**

A.7 Other representations

There are more representations of quantum state vectors other than the ones mentioned in the main text, such as one based on match-gates []. We specifically mention the extended stabilizer formalism, which is based on the stabilizer formalism that forms the basis for many techniques in quantum computing, such as measurement-based computation []. The stabilizer formalism enables polynomial-time simulation of Clifford circuits [], and has been extended to universal quantum circuits where the asymptotic runtime is fixed-parameter tractable in the number of non-Clifford gates []. **[Tim#14: todo add citations]**

A.8 W state and GHZ state

In this section, our aim is to include two simple quantum algorithms whose intermediate states are sparse: preparation circuits of W state and GHZ state, which are two different yet representative forms of tripartite entanglements [124]. The preparations of the W state and the GHZ state have been found in many application, including secure communications [103, 115, 116, 149] and quantum metrology [122]. **[Shih-Han#2: can you please add 2 sentences here why W and GHZ state worth our investigation, e.g., maybe they are widely used for xxx—see above]**

[Rihan#4: @Littau, can you please add explanation to W, GHZ, QFT circuits, please explain with Fig.4-6. mention gate count]

The W state. The 3-qubit W state is defined as

$$|W_3\rangle = \frac{1}{\sqrt{n}}(|001\rangle + |010\rangle + |100\rangle).$$

An n-qubit generalization would be a uniform superposition of n-bit strings with Hamming weight 1, i.e., $(|0000...1\rangle + |000...10\rangle + \ldots + |100...00\rangle)/\sqrt{n}$. Let's call the state $|W_n\rangle$. The state is sparse enough since the number of non-zero amplitudes in the standard basis is n but there are 2^n component. For our purposes, it seems sufficient to give some operator whose eigenvector is $|W_n\rangle$.

The W state preparation circuit. We implement the circuits following [42].

$$G(p) = \begin{pmatrix} \sqrt{p} & -\sqrt{1-p} \\ \sqrt{1-p} & \sqrt{p} \end{pmatrix}$$

$$G\left(\frac{1}{2}\right) = \sqrt{\frac{1}{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \quad G\left(\frac{1}{3}\right) = \begin{bmatrix} \sqrt{\frac{1}{3}} & -\sqrt{\frac{2}{3}} \\ \sqrt{\frac{2}{3}} & \sqrt{\frac{1}{3}} \end{bmatrix}$$

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

A.9 QFT

Appendix B ANALYSIS AND PRELIMINARY RESULTS

In this section, we first explain how to represent the quantum state with the relational model in Sec.B.1. In Sec. B.2, we analyze the space and time complexities of the relational model, comparing it with existing quantum data representations. In Sec. B.3 and B.4, we detail the experimental setup and report preliminary results. **[Floris#1: This section seems decoupled from the representations given earlier in appendix A?]**

B.1 Relational representation for quantum states

In Sec. A, we have introduced tensor networks. We can simply represent an n-qubit state ψ as an order-n tensor, as shown in Fig. 8a. **[Floris#2: Not sure what s_n, gray box, and lines are...]** With more specialization, tensor networks encompass a family of representations. One commonly used approach is the Matrix Product State (MPS), which employs smaller tensors, i.e., order-3 tensors and matrices, as shown in Fig. 8b. **[Floris#3: Again, not sure why that figure represents matrices. Perhaps this can be explained more formally, if that's needed. Sec. A we say nodes are tensors, edges are common indices. I think my confusion is the interpretation of the $s_1, \ldots, s_n$.]**

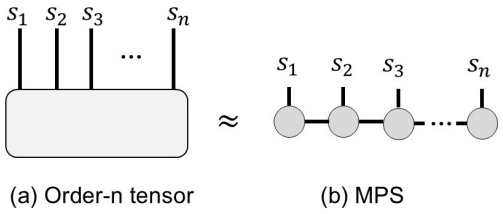


Figure 8: (a) N-qubit state ψ as a order-n tensor; (b) ψ represented in MPS

[Floris#4: I am confused: what has MPS to with relational representation?] MPS provides memory-efficient yet approximate simulation results, with limitations mainly to quantum circuits exhibiting low entanglement.

To address more general simulation scenarios, we design an approach using a relational model extended from order-n tensors. **[Floris#5: Confused again: what are comparing against representation-wise?]**

Relational representation of quantum state. A quantum state $|\psi\rangle$ is a superposition of basis states, with some of these basis states having non-zero coefficients. A coefficient is a complex number with the real part and imaginary part. To efficiently represent states using RDBMS, given an n-qubit state, we encode the non-zero coefficients and their corresponding basis state indices in a relation R with arity $n+2$. This can be expressed as follows:

$$R(s_1, s_2, \cdots, s_n, r, i) \mid s_k \in \{0,1\}, k \in [1,n], r \in \mathbb{R}, i \in \mathbb{R}, \quad (1)$$

where:

- n is the number of qubits.
- $s_1, s_2, \cdots, s_n$ represent the binary indices of the basis states in the computational basis ($|0\rangle$ or $|1\rangle$ for each qubit).
- r and i are the real and imaginary parts, respectively, of the complex coefficient associated with the k-th non-zero basis state. Both r and i are real-valued.
- A tuple $t \in R$ is a tuple representing one non-zero basis state and its coefficient. The total number of tuples is $\text{nnz}(|\psi\rangle)$, where $\text{nnz}(|\psi\rangle)$ denotes the number of non-zero coefficients in the quantum state $|\psi\rangle$.

Example B.1. In Fig. 9, we illustrate the relational representation of three quantum states:
1) The 3-qubit W state $|W_3\rangle = \frac{1}{\sqrt{3}}(|001\rangle + |010\rangle + |100\rangle)$ is a superposition of three basis states with non-zero coefficients, which are represented by three tuples in Fig. 9ca. For instance, the tuple $(0, 0, 1, \frac{1}{\sqrt{3}})$ corresponds to the basis state $|001\rangle$ with its coefficient $\frac{1}{\sqrt{3}}$.
2) The 3-qubit GHZ state $|\text{GHZ}_3\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ is a maximally entangled state involving only two basis states with non-zero coefficients, represented by two tuples in Fig. 9cb. Since the coefficients in the W and GHZ states are real values, the imaginary column i has only zero values and is therefore omitted in the relational representation.
3) The output of the 3-qubit QFT circuit in Fig. 7 is $|\psi_3\rangle = \frac{1}{4}\Big(2|000\rangle + (1+\frac{1}{\sqrt{2}}+\frac{i}{\sqrt{2}})|100\rangle + (1+i)|010\rangle + (1-\frac{1}{\sqrt{2}}+\frac{i}{\sqrt{2}})|110\rangle + (1+\frac{1}{\sqrt{2}}-\frac{i}{\sqrt{2}})|101\rangle + (1-i)|011\rangle + (1+\frac{1}{\sqrt{2}}-\frac{i}{\sqrt{2}})|111\rangle\Big)$. The QFT final state involves all eight basis states with only real coefficients given input state $|000\rangle$, which results in the relational table of 4 four columns and 8 rows in Fig. 9c.

s_1	s_2	s_3	r
0	0	1	$\frac{1}{\sqrt{3}}$
0	1	0	$\frac{1}{\sqrt{3}}$
1	0	0	$\frac{1}{\sqrt{3}}$

(a) 3-qubit W state

s_1	s_2	s_3	r
0	0	0	$\frac{1}{\sqrt{2}}$
1	1	1	$\frac{1}{\sqrt{2}}$

(b) 3-qubit GHZ state

s_1	s_2	s_3	r	i
0	0	0	$\frac{1}{2}$	0
0	0	1	$\frac{1}{4}+\frac{1}{4\sqrt{2}}$	$\frac{1}{4\sqrt{2}}$
0	1	0	$\frac{1}{4}$	$\frac{1}{4}$
0	1	1	$\frac{1}{4}-\frac{1}{4\sqrt{2}}$	$\frac{1}{4\sqrt{2}}$
1	0	1	$\frac{1}{4}-\frac{1}{4\sqrt{2}}$	$-\frac{1}{4\sqrt{2}}$
1	1	0	$\frac{1}{4}$	$-\frac{1}{4}$
1	1	1	$\frac{1}{4}+\frac{1}{4\sqrt{2}}$	$-\frac{1}{4\sqrt{2}}$

(c) Final state $|\psi_3\rangle$ of 3-qubit QFT circuit for GHZ state

Figure 9: Relational representations of different quantum states.

When to use the relational representation? The example above provides insight into scenarios where the relational representation is particularly advantageous for simulation. In sparse quantum states, such as the W state and GHZ state, the number of basis states with non-zero coefficients is significantly smaller than

Type	Exact representation		Approximate representation
Representation	Order-n tensor (Baseline I in NumPy)	Relational model (RDBMS implementation)	MPS (Baseline II in NumPy)
General	$O(2^n)$	$O(n \cdot \text{nnz(State)})$	$O(n\chi^2)$
W State	$O(2^n)$	$O(n^2)$	$O(n)$
GHZ State	$O(2^n)$	$O(n)$	$O(n)$
QFT	$O(2^n)$	$O(n \cdot 2^n)$	$O(n\chi^2)$

Table 1: Space complexity comparison of different representations of state $|\psi\rangle$. n is the number of qubits, $nnz(|\psi\rangle)$ denotes the number of non-zero coefficients in the quantum state $|\psi\rangle$, and χ is the bond dimension when $|\psi\rangle$ is represented in MPS.

the total number of basis states (2^n). In these cases, the relational model offers a compact and efficient representation.[5]

However, as illustrated in Fig. 9c, the benefit of the relational model diminishes for dense states like the QFT final state, where a majority of the basis states have non-zero coefficients. In the following sections, we formally analyze the space and time complexities of the relational representation to further evaluate its applicability.

B.2 Our analysis

B.2.1 Space complexity of order-n tensor and MPS. The space complexity of tensor networks is a well-studied topic [128]. We list the known space complexity of representing quantum states in order-n tensors and MPS Table 1, which correspond to baselines I and II in Sec. B.3 -Sec. B.4. When we represent an n-qubit quantum state as an order-n tensor, each dimension has size 2. The size of the tensor grows exponentially with the number of qubits, i.e., $O(n^2)$. As shown in Fig. 8, MPS employs smaller tensors: two matrices with the size of $2 \times \chi$ on the boundaries, and $n-2$ order-3 tensors with the size of $\chi \times 2 \times \chi$ in the middle. This is why MPS has the space complexity of $O(n\chi^2)$. In the case of W state and GHZ state circuits, the bond dimension χ is 2, which is a constant. This suffices because for both the W and GHZ states, the Schmidt rank for any bipartition is at most 2, allowing their entanglement to be fully captured with a bond dimension of 2. Thus the space complexity for these two cases are $O(n)$.

B.2.2 Space complexity of relational representation. In the following, we mainly analyze the space complexity of the relational representation. Besides the general cases, we discuss two sparse circuits, W state preparation, and GHZ state preparation, and a dense circuit, QFT algorithm. In all three of these circuits, the final states have the largest dimensions, as the gates are all small tensors (one-qubit gate can be presented as 2×2 matrices, and two-qubit gates are represented as order-4 tensors with size of $2 \times 2 \times 2 \times 2$). Thus, the space complexity of the whole circuits is the same as the space complexity of the final states.

General cases. The space complexity of dense representation is also $O(2^n)$. This applies to the W state, GHZ state, and the qubit states simulated in QFT algorithm.

W state. An n-qubit W state is represented as below:

$$|W_n\rangle = \frac{1}{\sqrt{n}} \left(|10\cdots0\rangle + |01\cdots0\rangle + \cdots + |00\cdots1\rangle\right)$$

[5] Further optimizations to enhance this representation remain a promising avenue for future research.

When we represent an n-qubit W state as a tensor, there are n non-zero elements. Thus, the table has n rows and $n+1$ columns. For instance, as shown in Fig. 9a, the table representing 3-qubit GHZ state has three rows and four columns. Thus, the space complexity is $O\left(n^2\right)$.

GHZ state. An n-qubit GHZ state is represented as below:

$$|\text{GHZ}_n\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle^{\otimes n} + |1\rangle^{\otimes n}\right)$$

It has a superposition of two basis states: either all qubits being $|0\rangle$ or all qubits being $|1\rangle$. In the database implementation, the corresponding relational representation has $n+1$ columns and only two rows representing $|00\cdots0\rangle$ and $|00\cdots0\rangle$. We illustrate an example of 3-qubit GHZ state in Fig. 9b. The space complexity is $O\left(n\right)$.

QFT final state. In the case of a dense circuit like QFT, in the RDBMS representation, the non-zero elements are close to 2^n. The table's dimension becomes $n+1$ columns and 2^n rows. The space complexity is $O(n2^n)$. Fig. 9c shows an example of the relational representation of 3-qubit QFT circuits. Compared to the cases of sparse circuits, there is less benefit of applying the relational representation here.

Sparse Circuits. In this work, we highlight the benefits of simulating sparse quantum circuits compared to dense circuits. A quantum circuit is considered sparse if the sparsity of the quantum state does behave monotonically throughout the circuit's evolution and the final state retains a high degree of sparsity. In the context of quantum computing, such circuits are commonly found in state preparation tasks [109], where the resulting quantum states occupy only a small fraction of the exponentially large state space , making them inherently sparse. The problem of evaluating the sparsity of a quantum circuit can also be seen as observing the branching factor for specific gates [178]. Operators with a branching factor higher than 1 would decrease the sparsity of a state. In quantum computing this is typically only the case for gates that apply a rotation to a given qubit, the most popular example of such a rotation would be the Hadamard gate.

Type	Exact representation		Approximate representation
Representation	Order-n tensor (Baseline I in NumPy)	Relational model (RDBMS implementation)	MPS (Baseline II in NumPy)
W State	$O(n \cdot 2^n)$	$O(n^3)$	$O(n)$
GHZ State	$O(n \cdot 2^n)$	$O(n^2)$	$O(n)$
QFT	$O(n^2 \cdot 2^n)$	$O(n^3 \cdot 2^n)$	$O(n^2\chi^3)$

Table 2: Time complexity comparison for different representations of state $|\psi\rangle$. n is the number of qubits, $nnz(|\psi\rangle)$ denotes the number of non-zero coefficients in the quantum state $|\psi\rangle$, and χ is the bond dimension when $|\psi\rangle$ is represented in MPS.

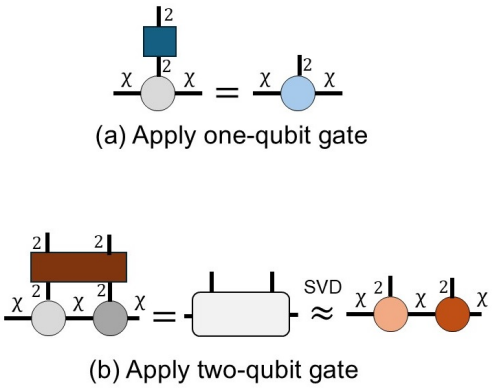


Figure 10: Applying gates to qubits in MPS

B.2.3 Time complexity. The time complexity of simulating general quantum circuits is a complicated problem. Relevant factors include the number of gates, circuit depth, type of gates, circuit length, and other factors depending on the specific simulation application. In practice, the actual implementation and hardware setting (e.g., GPU cache size) [166] can also significantly impact time efficiency.

As an early work, we aim to compare relational representation with tensor networks in Table 2. As a starting point, we focus on the three circuits: W state and GHZ state preparation courts, and QFT circuits. Consistent with the preliminary results in Sec. B.3, we primarily analyze the impact of the number of qubits, gates, and gate types. For simplicity, we assume that gates are applied sequentially.

W state. An n-qubit W state preparation circuit has one X gate, and $n-1$ controlled $G(p)$ gates and $n-1$ CNOT gates. As the number of qubits n increases, the computation cost of gates grows linearly, resulting in a complexity of $O(n)$.

1) Order-n tensor. The X gate is a single-qubit gate represented as 2×2 matrix. Both controlled $G(p)$ and CNOT are two-qubit gates represented as order-4 tensors of size $2 \times 2 \times 2 \times 2$. With a large value of n, the largest tensor is the order-n tensors representing the qubit states, which scales as 2^n. The time complexity for this representation is $O(n^2 \cdot 2^n)$.

2)Relational representation. Since the time complexity of the relational approach depends on the number of `JOIN`s executed. Due to the implementation the number of `JOIN`s is exactly the number of gates. Each `JOIN` itself will depend on the number of rows in each of the operands, since one operand will always be a gate tensor, we can consider its size constant as it's in the worst case for two-qubit gates $2^4 = 16$. The other operand is the stored state with its number of rows equal to the intermediate non-zero probability states. For the n-qubit W this leads to a time complexity of $O(n^3)$.

3) MPS. As shown in Fig. 10, a tensor in MPS has a maximum size of $\chi \times 2 \times \chi$, where χ is the bond dimension. Applying a single-qubit gate (2×2 matrix), involves a contraction step with a complexity of $O(\chi^2)$. Applying a two-qubit gate requires contracting tensors into an intermediate order-4 tensor, followed by a decomposition using Singular Value Decomposition (SVD) to reduce back to smaller tensors in the MPS format. The SVD step dominates the cost, and has a complexity of $O(\chi^3)$. Combining this with the gate count, the overall complexity is $O(n\chi^3)$. For the W state circuit, the bond dimension χ can be set to a constant value $\chi = 2$, reducing the final complexity to $O(n)$.

GHZ state. For GHZ state, the analysis for order-n tensor and MPS are similar to W state. Instead of two controlled gates in the GHZ state preparation circuit there are exactly n gates, starting with one Hadamard gate on the first qubit followed by $n-1$ CNOT gates. Therefore the time complexity only differs in a constant factor from the W state preparation circuit.

For relational representation, the reasoning is similar to the W state preparation circuit, which results in a linear time complexity.

QFT. The QFT circuit consists of n^2 gates. The analysis for order-n tensor is similar to W state, i.e., $O(n^2 \cdot 2^n)$. For MPS the time complexity depends on the initial state as this would influence the intermediate states. In our experiments, we start with the initial state of all qubits in state $|0\rangle$, hence no entanglement in the system. Therefore all intermediate states can be represented with maximum accuracy using a maximum bond dimension of 2. In general the bond dimension will influence the size of the matrices in the MPS and hence the time complexity. With the number of gates being n^2 this leads to a time complexity of $O(n^2 \cdot \chi^3)$.

For the relational implementation the time complexity scales similar to the order-n tensor approach except for an additional factor n with $O(n^3 \cdot 2^n)$.

B.3 Experiment setting

In this section, we explain the experiment set up for the preliminary results in Sec. B.4.

We conducted our experiments on on a machine equipped with an Intel Core i9-12900K CPU (16 cores, up to 5.2 GHz), 64 GB RAM, and running Ubuntu 22.04 LTS. Each experiment was executed on a single CPU core, and we report the average execution time across 100 runs.

Baselines and RDBMS implementation. We have implemented two NumPy-based baselines with different data representation:

Baseline I: tensor network representation and tensor contractions are performed over order-n tensor, where n is the number of qubits.
Baseline II: Matrix Product State (MPS) representation.
Both baselines were developed in Python 3.13.0 and NumPy 2.1.3, which are relative to widely used linear algebra libraries for classical simulation.
RDBMS-based solution: we have implemented and evaluated the relational representation in Sec. B.1 with three database systems: PostgreSQL 17.2, SQLite 2.6.0, DuckDB 1.1.3. The tested relational databases include a PostgreSQL database running in a docker container, DuckDB used in-memory via the Python API and an in-memory instance of a SQLite database.
Input circuit generation. The inputs of a classical simulator are quantum circuits. In preliminary experiments, we evaluated the baselines and RDBMS solutions over three widely used circuits: W state preparation circuit, GHZ state preparation circuit and QFT circuit. We have developed a circuit generator, which is publicly available online.[6] The input circuits are generated in JSON-format with a given number of qubits and a list of gates which are JSON-objects themselves containing an identifier like "H" for the Hadamard gate and a list of the qubits it acts on. The ideal contraction path is part of the input and generated by the Python library opt_einsum 3.3.0[7] providing the generated Einstein summation notation in the format "ij,jk->ik". This notation corresponds to a tensor contraction described by the Einstein summation notation $C_{ik} = \sum_j A_{ij}B_{jk}$, where i, j, k are the indices of the corresponding tensors A, B and C.

For RDBMS solutions, our circuit generator produces SQL queries representing the circuits, following the methodology in state-of-the-art for transforming quantum circuits into SQL queries [20]. The generated SQL queries include the representation of the initial quantum state — typically the zero state, represented as the vector $|0\rangle$ (see Fig. 5 for an example) — and the quantum gates as tensors. The tensor contraction paths of input circuit are specified by the Einstein summation notation, which are translated into SQL operations. Specifically, tensor contractions are expressed as `INNER JOIN` operations in SQL, with tensors as relational tables. The contraction order is optimized using Common Table Expressions (CTEs), enabling efficient computation and reuse of intermediate results. An example of applying a Hadamard gate to a one-qubit system would look as follows: **[Rihan#5: add TABLES/SQL of both H and CONT, change name to s1, s2, r; explain the swamp part]**

```
WITH T0(i, re, im) AS (
    VALUES (CAST(0 AS INTEGER),
            CAST(1.0 AS DOUBLE PRECISION),
            CAST(0.0 AS DOUBLE PRECISION))
    ),
H(i,j, re, im) AS (
    VALUES (CAST(0 AS INTEGER),
            CAST(0 AS INTEGER),
          CAST(0.7071067811865475 AS DOUBLE PRECISION),
            CAST(0.0 AS DOUBLE PRECISION)),
            (0, 1, 0.7071067811865475, 0.0),
            (1, 0, 0.7071067811865475, 0.0),
            (1, 1, -0.7071067811865475, 0.0)
    )
SELECT H.i AS i,
SUM(H.re * T0.re - H.im * T0.im) AS re,
SUM(H.re * T0.im + H.im * T0.re) AS im
FROM H, T0 WHERE H.j=T0.i GROUP BY H.i ORDER BY i
```

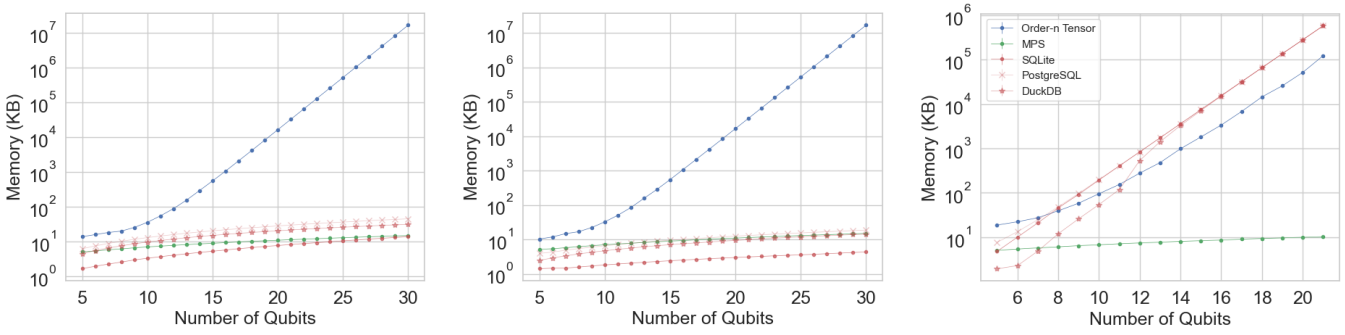
[6] https://github.com/InfiniData-Lab/QuantumProject
[7] https://pypi.org/project/opt-einsum/

B.4 Preliminary results

In Sec. 4.2, we have posed the first research question (Q1): Should simulation workloads be pushed to existing DBMSs? Based on the complexity analysis in Tables 1 and 2, RDBMS solutions appear to offer advantages over order-n tensor representation, when input circuits involve sparse tensor computation, such as W state and GHZ state preparation circuits. To investigate this, we compare RDBMS solutions with baselines in terms of memory usage (Sec. B.4.1) and execution time (Sec. B.4.2).

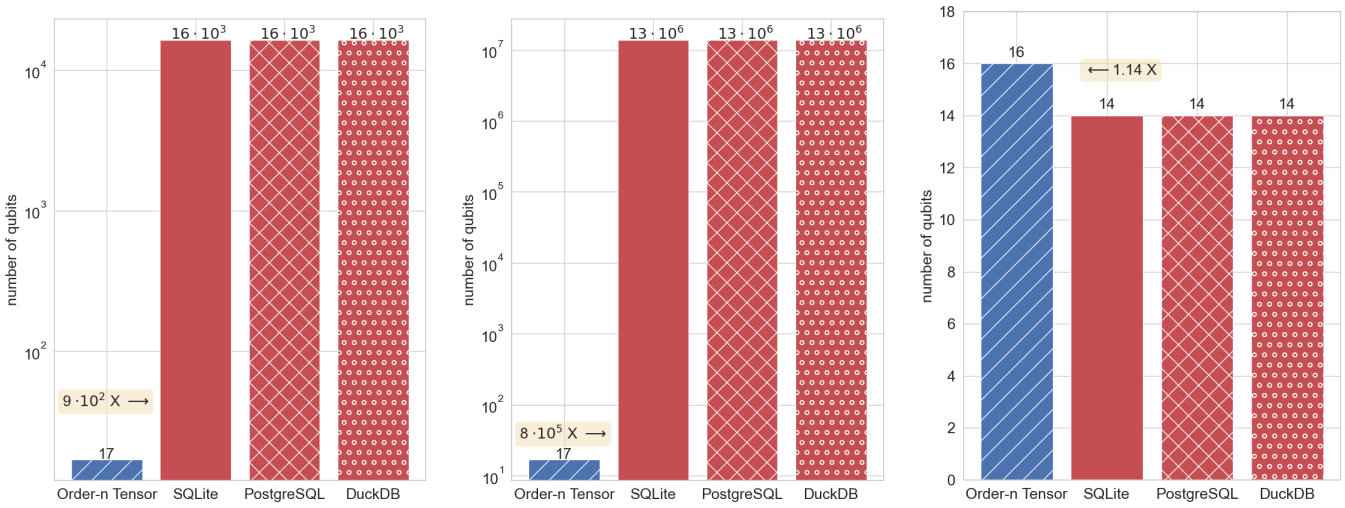
Furthermore, in Sec. B.4.3, through preliminary experiments, we have discovered a research gap in the state-of-the-art [20]: the inefficiency of SQL query generation for simulation workloads. Understanding this gap is crucial for future work toward developing an end-to-end simulation pipeline based on database systems.

Q1. *Should we push the simulation workload to existing DBMSs?*
(a) When to use the relational representation?



(a) W State (sparse circuit) (b) GHZ State (sparse circuit) (c) QFT (dense circuit)

Figure 11: Memory usage comparison of three RDBMS-based solutions (red) against general order-N tensor baseline (blue) and SotA MPS baseline (green). The x-axis shows increasing numbers of qubits, and the y-axis shows memory consumption (KB) in $\log_{10}$-scale.



(a) W State (sparse circuit) (b) GHZ State (sparse circuit) (c) QFT (dense circuit)

Figure 12: Simulated number of qubits comparison of RDBMS-based vs. NumPy-based solutions for a memory limit of 2.0GB. y-axis is in $\log_{10}$-scale for W and GHZ state preparation circuits.

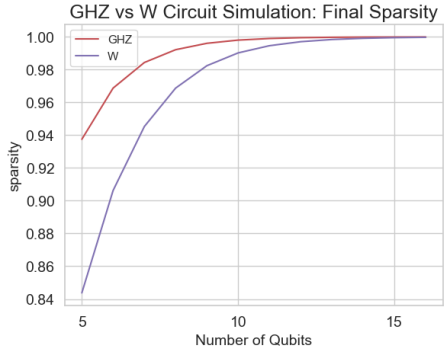
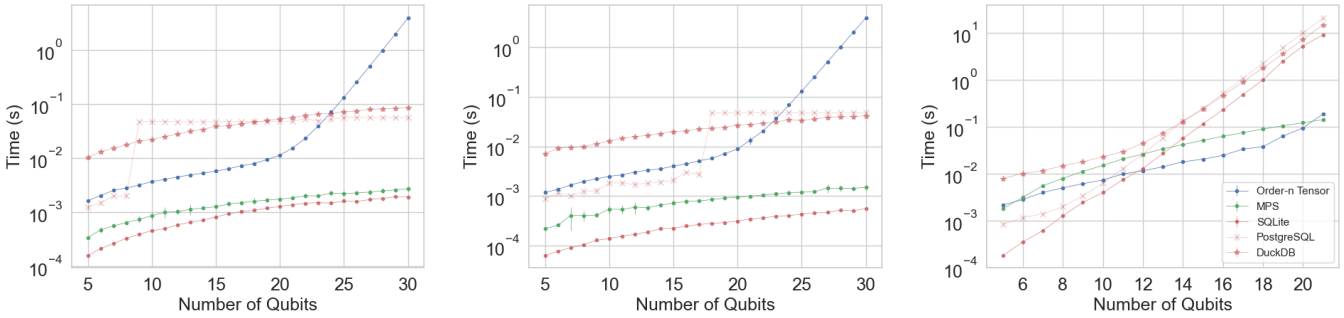


Figure 13: Sparsity GHZ vs W State preparation circuit

B.4.1 Memory Usage. Fig. 11a compares the memory usage of RDBMS solutions and two baselines. The memory usage of the order-n tensor baseline grows exponentially, consistent with its $O(2^n)$ memory complexity in Table 1. This rapid growth becomes evident as the number of qubits increases. In contrast, the RDBMS solutions, i.e., PostgreSQL, DuckDB, and SQLite, demonstrate significantly lower memory consumption in sparse scenarios (W state and GHZ state preparation circuits). RDBMS solutions have exhibited low memory usage, with small variations as the number of qubits increases. Memory usage for RDBMS solutions exhibits small variation with increasing qubits, aligning with the space complexity analysis in Table 1, which predicts a linear growth for GHZ states and a quadratic growth for W states. This memory efficiency arises from the RDBMS solutions' ability to compactly represent non-zero values, providing an effective form of sparse tensors in simulation.



(a) W State (sparse circuit) **(b) GHZ (sparse circuit)** **(c) QFT (dense circuit)**

Figure 14: Execution time comparison of RDBMS-based vs. NumPy-based solutions. The y-axis is in $\log_{10}$ scale. The line colors are based on the different data representations with blue being order-n tensor representation, red the RDBMS COO representation and green the MPS representation.

Another interesting discovery from Fig. 11a is that RDBMS solutions perform comparably to the SotA baseline using MPS representation for sparse circuits. In particular, the in-memory database SQLite demonstrates a slight advantage over the MPS baseline for smaller numbers of qubits. This advantage of RDBMS solutions is even more evident in the sparser GHZ state, as shown in Fig. 11b. Notably, as an early-stage study, we utilized the default configurations for all three RDBMS systems, leaving significant room for optimization to further enhance their performance.

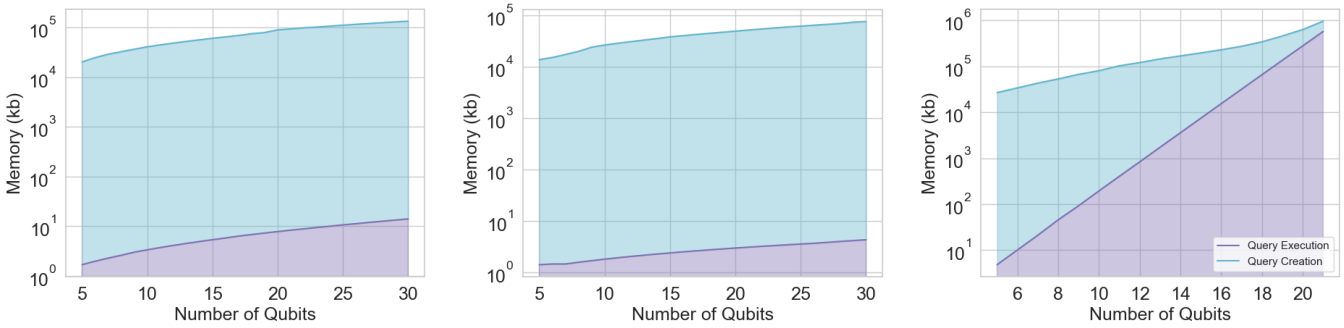
In contrast, in Fig. 11c, the memory advantage of the RDBMS solutions is not observed for the Quantum Fourier Transform circuit. The memory usage of the RDBMS solution increases at a rate comparable to order-n tensor baseline, sometimes even exceeding it. This aligns with our space complexity analysis in Table 1, where the relational representation scales as $O(n \cdot 2^n)$, while the order-n tensor representation scales as $O(2^n)$. The primary reason is that the QFT circuit mainly involves dense tensor computation. Fig. 13 compares the sparsity of the final state of QFT with W and GHZ states, which are much sparser. Besides the final state, dense states inherently require large intermediate tensors during tensor contraction.

We continue to explore the potential scalability advantages of RDBMS solutions compared to the general baseline using order-n tensor representation. Fig. 12 compares the maximum number of qubits that can be simulated by RDBMS-based solutions and baselines under a 2.0 GB memory limit. For W state preparation circuit in Fig. 12a, The RDBMS solutions (SQLite, PostgreSQL, DuckDB) outperform NumPy (Tensor) by a significant margin, supporting over $16 \cdot 10^3$ qubits compared to just 17 qubits for NumPy. This represents a 963× improvement in scalability. The GHZ state preparation circuit with even sparser tensors (Fig. 13), with RDBMS solutions simulating up to $13 \cdot 10^6$ qubits, compared to 17 for NumPy (Tensor), leading an $8 \cdot 10^5\times$ improvement. This showcases the significant memory-efficiency advantage of database systems in sparse scenarios, when the number of qubits can be simuluated scales up. For the dense QFT circuit, RDBMS solutions, simulating up to 14 qubits compared to 14 for order-N tensor- a reduction of 14%. This reflects the inefficiency of RDBMS solutions in handling dense computations, where large intermediate tensors and overheads negate their sparse storage advantage. In summary, RDBMS solutions demonstrate superior scalability for quantum circuits with sparse tensors (W and GHZ), while limited in the circuit with dense tensor computations (QFT). These results suggest that RDBMS-based approaches are highly effective for sparse quantum circuits but require further optimization to compete in dense circuit scenarios.

B.4.2 Execution Time. Figure 14a compares the time-wise efficiency of the RDBMS solutions with the baselines. The results and observation are similar to memory usage. The execution time results reveal a nuanced tradeoff between sparse and dense circuits. Sparse quantum circuits, such as the W State and GHZ state preparation circuits, RDBMS solutions showed significant runtime advantages compared to NumPy-based baselines. For smaller qubit counts (e.g., below 20 qubits), NumPy outperformed some of the database systems due to its lower constant overhead and direct handling of tensor operations.

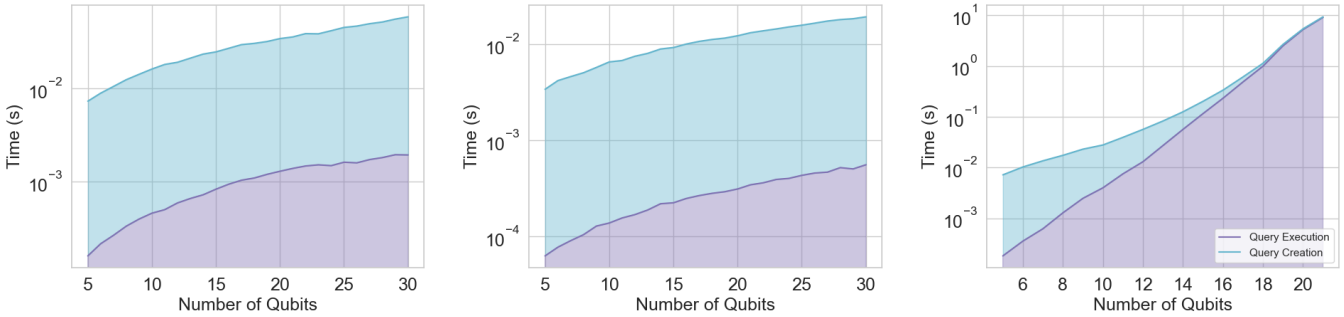
As shown in Figure 14a, the RDBMS implementation is comparable to state-of-the-art method MPS, especially for higher qubit counts. As shown in figure 14c, dense ciruits, such as the QFT circuit, exhibited a different trend. Numpy's tensor contraction implementation outperformed PostgreSQL, DuckDB and SQLite, primarily due to its efficient handling of dense tensor operations without additional overhead and the COO representation that is used in RDBMS. This highlights that while database systems offer scalability for sparse scenarios, their performance deteriorates for dense circuits compared to specialized numerical libraries. However, this also shows a potential for improvement on the database side, since at least some of the reasons for bad performance lie in the naive data representation used in this implementation. While both implementations show an exponential increase over the number of qubits, the increase for RDBMS implementations is even higher as previously shown in the theoretical time complexities in table 2.

To summarize, for $Q1$ now, we can say it is a promising solution to push simulation workload to RDBMS. For circuits with sparse tensors, opppunites remins for the RDBMS system optimization. For tensor circuits, we might need more advanced relational representation than Sec. B.1, system implementation, and optimization.



(a) W State (sparse circuit) **(b) GHZ (sparse circuit)** **(c) QFT (dense circuit)**

Figure 15: Memory comparison of SQL query creation vs. SQL query execution. The y-axis is in *logarithmic* scale.



(a) W State (sparse circuit) **(b) GHZ (sparse circuit)** **(c) QFT (dense circuit)**

Figure 16: Runtime comparison of query creation vs. query execution. The y-axis is in $\log_{10}$ scale.

Takeaways:
T1 RDBMS for Sparse Circuits:
When the input quantum circuit is mainly involved with sparse tensor computation, relational representation is efficient and scalable, even comparable, and may have an advantage compared to state-of-the-art representations.
T2 Future Optimizations:
Exciting research opportunities remain, to optimize existing RDBMSs for simulation workload.

B.4.3 Input query generation. Although not in our original submission, through preliminary experiments we discovered a research gap in existing works as below:

Q1. *Should we push the simulation workload to existing DBMSs?*
(b) How good are the state-of-the-art SQL query generation methods for quantum circuit simulation?

To generate the SQL queries for circuit simulation, state-of-the art approach [20] requires transforming the tensors — initially represented as NumPy arrays—into relational representation using a Coordinate (COO) tensor format[8]. Additionally, the contraction path for tensor operations must also be converted into a sequence of relational JOIN operations in SQL.

[8]https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.html

Figure 15 compares the memory overhead for creating the SQL query using the approach in [20]. For W and GHZ state preparation circuits, the memory overhead is significant, though it scales linearly with the number of qubits. This overhead arises primarily because to generate the SQL queries for a circuit, the state-of-the-art [20] requires transforming the tensors — initially represented as NumPy arrays—into relational representation using a Coordinate (COO) tensor format. Additionally, the contraction path for tensor operations must also be converted into a sequence of relational JOIN operations in SQL.

The memory cost of tensor translation depends on the variety of tensors involved. For circuits like the GHZ state preparation, this cost remains constant, as the tensor representation does not change with an increasing number of qubits. In contrast, for the W State preparation circuit, the memory usage scales linearly with the number of qubits, as each added qubit introduces a new variant of the $G(p)$ gate, which must also be translated.

The translational overhead for the contraction path—the sequence of `JOIN` operations—also scales linearly with the number of qubits. This is a consequence of the increasing circuit depth (i.e., the number of applied gates) as qubits are added. This translation effectively converts the tensor network or quantum circuit into its relational representation, which is necessary for simulation within an RDBMS framework.

Fig. 16 shows the query generation time and execution time comparison. The creation of SQL queries introduces an additional time overhead in the current implementation. This overhead results from constructing a string representation of the SQL query, where the string's length scales with the number of gates in the quantum circuit. The time cost is thus proportional to the number of gates, as reflected in Fig. 16.

The time overhead is also attributable to two aforementioned factors: the translation of tensors into COO-formatted database relations and the conversion of the contraction path into a sequence of `JOIN` operations. Each of these steps introduces computational effort that grows linearly with the number of gates or circuit depth.

Takeaways:
T3 SOTA SQL query generation methods for quantum circuit simulation is inefficient.
Possible improvement strategies: pre-stored in the database in COO format, eliminating the need for runtime translation and making the tensors directly accessible for quantum circuit operations. Future improvements, such as pre-storing tensors in the database or optimizing the contraction path generation process, could significantly reduce this overhead.

Appendix C MORE DATA MANAGEMENT OPPORTUNITIES

In this section, we continue with Sec. 4.3 and list more research opportunities.

NoSQL databases & Quantum data management. NoSQL databases, present a promising direction for managing unstructured data, offering scalability and flexible schemas that are well-suited for analyzing quantum experiments and simulations. For instance, TileDB [5] is an array database that supports HDF5 files, enabling the conversion of HDF5 datasets into TileDB arrays for more efficient queries and storage. JSON is a widely used file type for circuit descriptions and simulation configurations in quantum frameworks, e.g., Qiskit [4], Cirq [2], or Braket [1]. Document stores like MongoDB [3] can efficiently manage JSON. Exploring the recent effort of graph analytics [15, 108, 111, 176, 185] and graph databases [160] such as Neo4j[9], TigerGraph[10], might yield valuable insights to advance these error correction techniques. Continuing the discussing in Sec. 4.2, relational and NoSQL databases, each offer unique advantages, but no single database system is likely to meet the diverse requirements of all quantum applications; the optimal choice will depend on the specific needs of each quantum use case.

Quantum network & Distributed databases. Quantum networks hold the potential to offer advantages and capabilities beyond those of classical internet systems, e.g., secure communication [29] or distributed computing [123]. Developing quantum network requires simulators of various kinds, such as simulating timing behaviour accurately [40, 181] or for supporting the development of application software [46]. It is interesting to explore whether the quantum network applications and tools can be supported and advanced by distributed databases [129] and cloud-native databases [51], while also considering quantum-specific features like entanglement.

[9] https://neo4j.com/
[10] https://www.tigergraph.com/

Table 3: Comparison of simulation with and without databases technology

Feature	Without Databases	With Databases
Data Storage & Retrieval	Data, such as simulation configurations, intermediate and final results, is stored in flat files (e.g., HDF5, JSON). Retrieval requires custom scripts and manual searches, which hinders data discovery, mining, and analysis.	Data stored in structured, indexable formats, enabling efficient organization and querying; optimized retrieval through indexing, caching, and execution plans, and facilitating the use of existing tools for data discovery, mining, and analysis.
Query Languages	Accessing data relies on low-level programming, inefficient for querying data.	High-level query languages (e.g., SQL, relational algebra, or Tensor Languages) simplify retrieval, enabling efficient queries.
Consistency, Recovery, and Transactions	Manual checks or customized scripts, error-prone.	Well-developed theory and tools, reducing human effort and possible errors.
Scalability	Limited by hardware capacity.	Potential benefits of databases for memory-efficient simulations; extended by using secondary storage and out-of-core database technologies

Appendix D MORE DISCUSSION ON SIMULATION WITH DATABASE TECHNOLOGIES

Continuing with Sec. 4, we compare simulation without and with databases in Table 3.

1. Data Storage & Retrieval. A common need in many simulation applications is managing data effectively, particularly optimizing retrieval and supporting advanced data analytics. Existing simulation tools often rely on flat files to store configurations and results, requiring custom scripts (e.g., Python) or manual searches for access. This approach is inefficient, time-consuming, and limits data discovery, mining, and analysis. For instance, in quantum network simulations, distributed experiments generate data from multiple users, but lack efficient tools for data discovery and analytics. In contrast, databases store data in structured, indexable formats, enabling efficient organization and fast querying. Optimized retrieval through indexing, caching, and execution plans enhances performance. Another benefit is that database technologies are widely used and have a rich ecosystem of commercial and open-source tools (e.g., data analytics, discovery, integration, and cleaning platforms) that support advanced analysis and integration with existing simulation workflows.

2. Query Languages. Current simulation tools typically access data relying on programming languages (e.g., Python) requiring customized optimizations for different quantum applications. This approach significantly increases development overhead and limits flexibility. In contrast, database systems leverage high-level query languages like SQL, relational algebra, or tensor-specific languages [28, 65, 95] to simplify data retrieval. These languages allow users to perform complex queries in a declarative and efficient manner, facilitating more effective querying and more sophisticated analyses with less human effort.

3. Consistency, Recovery, and Transactions. In D2/R2 **[Tim#15: text seems to be copied directly from reviewer comments, including copying of references to their questions]**, we have discussed the potential of database recovery technologies for large-scale simulation. Another interesting future research direction is to explore consistency and transactions in distributed quantum computing applications such as quantum networks. We take *NetSquid*[11] for example. NetSquid is a widely used quantum network platform developed at TU Delft, which enables researchers to model, simulate, and analyze quantum communication protocols and hardware. NetSquid supports discrete-event simulation, where an event starts only after the previous event ends. Please refer to our previous work [40] for more technical details. A possible extension of NetSquid's discrete-event simulation is to allow concurrency simulation requests from multiple users. In this future direction, we foresee the potential to utilize and further develop database consistency and transaction management technologies.

4. Scalability. We have discussed the bottleneck of scalability without databases in Sec. 4.1. With Fig. 14 and **??**, we identified the potential benefits of database-based approaches for memory-efficient simulations. Another promising direction is out-of-core simulation using databases [162]. When data sizes exceed memory limits, most main-memory databases switch to out-of-core strategies to complete queries. However, this often results in performance that is significantly slower than in-memory processing. Recent advancements in out-of-core database techniques, such as sorting [97], aggregation [96], and join, have redesigned operators to ensure performance degrades significantly less when the data size exceeds memory limits. It is an intriguing research direction to explore whether these techniques can enhance out-of-core simulation when simulation workloads are integrated with database systems.

[11] https://netsquid.org/